PEOPELS'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

Blida 01 University

Institute of Aeronautics and Space Studies

Manuscript

In partial fulfillment of the requirements for

The Degree of Master in Aeronautics

Specialty: Avionics

UAV Aerial Image-Based Forest Fire Detection Using Deep Learning

A Thesis Submitted by

Akila Keddous

Under the Supervision of

Prof. Lagha Mohand

Dr. Choutri Kheireddine

Blida 2020-2021

Acknowledgments

This thesis is submitted in partial fulfillment of the requirements for the degree of Master at the University of Saad Dahleb Blida 01. Under the supervision of Prof. Lagha Mohand and Dr. Choutri Kheireddine, the presented thesis work is carried out at the laboratory of Aeronautical Sciences at the institute of Aeronautics and space studies (IAES).

My thanks go in the first place to ALLAH Almighty who has illuminated my path with the glow of knowledge and science and for the will, the health, and patience that he lavished on me during all these years of study.

Fortunately, I have been given such great opportunities to collaborate with and be significantly benefited by many outstanding people during my whole Final year project. In particular, my deepest gratitude should go to my supervisors, Dr. Choutri Kheireddine who has provided me with these valuable chances and resources allowing me to successfully handle various challenges and difficulties in different critical phases of my project. Without his consistent support and guidance, I cannot achieve such results. and Prof. Lagha Mohand for providing useful feedback, strong support and for his positive attitude to both work and life, it is really my great privilege being able to work under the supervision of such passionate people with great diligence to research works.

I am also very grateful to the Drone girls Group: Yasmine, Nihad, Nouha, Fairouz thank you for your great help, from the discussion of challenges in the project and paper writing, through experimental and algorithm validation, to the problems that I met in my daily life. It is my great pleasure to conduct research works with them, share opinions with them, and learn from them.

I would like to thank my best friends: Amina, Kaouter, Chaima, Merieme, Rayane, Souhila and Nesrine thank you for always being there for me and encouraging me through the tough times you really made my study years more joyful. My good friend Merouane, words cannot express how thankful I am for the help that you ‘ve provided.

Finally, I would like to dedicate this work to my parents Nacir and Cherifa, my two brothers Hamza and Mustapha for their continuous support, for always pushing me forward and for providing me with everything I need. Thank you for always believing in me, without your encouragement, I would not have been here today.

Abstract

Forest fires are very dangerous. Once they become widespread, it is very difficult to extinguish. In this work, an Unmanned aerial vehicle (UAV) image-based Real-time Forest fire detection approach is proposed. Where we took advantage of recent development in computer vision systems and the rapid maneuverability of Unmanned Aerial Vehicles to improve the performance of the Real time detection, we designed and implemented a YOLOv2 Convolutional Neural Network Model in MATLAB to train on an aerial dataset, Experimental results show that our proposed system has high detection performance, and its detection speed reaches 58 Frame Per Second with a mean average precision of 0.87, thereby satisfying the requirements of real-time detection (Speed and Accuracy).

Key Words: Real-Time Fire Detection, Deep Learning, Convolutional Neural Network, Computer Vision, Unmanned Aerial Vehicles, Fire Datasets, YOLOv2.

Résumé

Les feux de forêt sont très dangereux. Une fois qu'ils se sont répandus, il est très difficile de les éteindre. Dans ce travail, nous avons proposé une approche de détection des incendies de forêt en temps réel basée sur les images obtenues des véhicules aériens sans pilote (UAV). Pour améliorer les performances de la détection en temps réel, nous avons profité du développement récent des systèmes de vision par ordinateur et de la maniabilité rapide des véhicules aériens sans pilote où nous avons conçu et mis en œuvre un modèle de réseau neuronal convolutif basé sur YOLOv2 architecture dans MATLAB pour s'entrainer sur une base de données aériennes. Les résultats expérimentaux montrent que notre système proposé présente une performance de détection élevée et que sa vitesse de détection atteint 58 image par seconde avec une précision moyenne de 0,87, satisfaisant ainsi les exigences de détection en temps réel (Vitesse et Précision).

Mots Clés : Détection en temps réel, Apprentissage Profond, Réseaux de Neurones Convolutifs, Vision par Ordinateur, Ensemble de Données d'Incendie, Véhicules Aériens sans Pilote, YOLOv2.

ملخص

حرائق الغابات خطيرة للغاية. بمجرد أن تصبح واسعة الإنتشار، من الصعب جدًا إخمادها. في هذا العمل ، تم اقتراح نهج للكشف عن حرائق الغابات بشكل آني بواسطة الصورة المقدمة من قبل الطائرات بدون طيار .حيث استفدنا من التطورات الأخيرة في أنظمة رؤية الكمبيوتر والقدرة على المناورة السريعة للطائرات بدون طيار لتحسين أداء الكشف الآني ، قمنا بتصميم و تنفيذ نموذج الشبكة العصبية التلافيفية متركزة على هندسة الYOLOv2 في MATLAB للتدريب على مجموعة بيانات جوية ، تظهر النتائج التجريبية أن نظامنا المقترح يتمتع بأداء كشف عالي، وسرعة اكتشافه تصل إلى 58 إطارًا في الثانية بمتوسط دقة 0.87 ، والتي تلبي متطلبات الكشف الآني.

كلمات مفتاحية: الكشف الآني,التعلم العميق,الشبكات العصبية التلافيفية,الرؤية بواسطة الكمبيوتر,مجموعة بيانات الحرائق,طائرات بدون طيار,YOLOv2.

Contents

Acknowledgments i

Abstract ii

List of Figures vii

List of Tables ix

Acronyms x

General Introduction 1

Chapter I: Introduction to Forest Fire Detection 4

1. Introduction: 5
2. Statistics on Forest Fire 5
   - 2.1. Forest fire Worldwide: 5
   - 2.2. Forest fire in Algeria: 7
3. Forest Fire Detection Existing Methods: 10
   - 3.1. Using Sensors: 10
   - 3.2. Using Computer Vision 10
   - 3.3. Using Deep Learning: 11
4. Forest Fire Detections Existing Systems 11
   - 4.1. Terrestrial Systems 11
   - 4.2. Unmanned Aerial Vehicles Systems 11
   - 4.3. Space borne (Satellite) Systems 13
5. State of the Art 14
   - 5.1. Computer Vision-Based Traditional Approach 14
   - 5.2. Computer Vision-Based Deep Learning Approach 15
6. Conclusion: 16

Chapter II: Deep Learning for Computer Vision 17

1. Introduction 18
2. Artificial Intelligence ,Machine Learning and Deep Learning 18
   - 2.1. Artificial Intelligence 18
   - 2.2. Machine Learning 19
   - 2.3. Deep Learning : 21
3. Convolutional Neural Network 22
   - 3.1. Definition 22
   - 3.2. CNNs Principal of Operation 22

4. Computer vision: 24
4.1. Definition 24
4.2. The Evolution of Computer Vision 25
4.3. Computer Vision with Machine Learning Approach 26
4.4. Computer Vision with Deep Learning Approach 27
5. Object Recognition 27
5.1. Definition 27
5.2. Object Recognition Computer Vision Tasks 27
5.3. Comparison between Single Object Localization and Object Detection: 29
5.4. Popular families of object recognition deep learning Models 30
6. Conclusion: 35
Chapter III: Forest Fire Detection Model Using YOLOv2 36
1. Introduction: 37
2. Fire Detection Dataset 37
2.1 Popular Fire Detection Datasets: 37
2.2 Building our Personalized Dataset: 42
3. Fire Detection Model: 52
3.1 YOLOv2 Network Architecture: 53
3.2 YOLOv2 Proposed Network Design: 55
3.3 YOLOv2 Network Training Process: 58
3.4Training Evaluation: 61
4. Conclusion: 63
Chapter IV: Real-Time Model Evaluation and Testing 64
1. Introduction: 65
2. Testing the Trained Model: 65
2.1. Test Set: 65
2.2. Test Detection Scenarios: 66
2.3. Test Detection Results: 67
3. Model Evaluation: 71
3.1 Evaluation Metrics: 72
3.2 Evaluation Results: 73
3.3 Results Discussion : 75
4. Real Time Detection Testing 76
4.1 Conducted experiment: 76
4.2 Test Results: 77

4.3 Results Discussion: 78
5. Conclusion 79
Conclusion and Perspectives 80
References 83

List of Figures

Chapter1

Figure I. 1:Annual evolution in areas covered by fire in Algeria (period 1876 -2012) [20]...... 8
Figure I. 2:Average annual fire risk in Algeria [20]... 9
Figure I. 3:Generalized multispectral imaging systems for early fire detection [1]. 14
Figure I. 4:Schematic Illustration of the UAV Based Forest Fire Detection System [4]......... 14

Chapter2

Figure II. 1:Artificial Intelligence, Machine Learning, Deep learning [35]………………18

Figure II.2:Types of Machine Learning [33] 20
Figure II. 3:Comparing a machine learning approach to categorizing with deep learning [29]. 21
Figure II. 4:the relation between the performance of the algorithms and the amount of input Data 22
Figure II. 5:Neural networks, which are organized in layers consisting of a set of interconnected nodes. [33] 23
Figure II. 6:Example of a network with many convolutional layers. [32] 24
Figure II. 7:Overview of the relationship of AI and CV. [32] 25
Figure II. 8:An example of a machine learning approach [33] 27
Figure II. 9:Overview of Object Recognition Computer Vision Tasks [27]........................... 28
Figure II. 10:Overview of Object Recognition Computer Vision Tasks for fire scene example 29
Figure II. 11:Single Object Localisation VS Object Detection [27] 29
Figure II. 12:Summary of the R-CNN architecture.. 30
Figure II. 13:Summary of the Fast R-CNN Model Architecture. 31
Figure II. 14:Summary of the Faster R-CNN Model Architecture [27].................................. 32
Figure II. 15:Summary of Predictions made by YOLO Model. [1] 33
Figure II. 16:Example of the representation chosen when predicting Bounding box position and shape [27] 34

Chapter3

Figure III. 1:Frame samples of the FLAME Dataset [35]……………………………………39

Figure III. 2:Frame samples of the FiSmo Datasets [36]. 41
Figure III. 3:Frame samples of the All-New Dataset. 43
Figure III. 4:Images loading and Fire label definition in Image Labeler App. 45
Figure III. 5:Image annotation with green bounding boxes. 45

Figure III. 6:Lebel Summary Graph of Manual Labeled Datasets. 46
Figure III. 7:Color Thresholder App interface. 47
Figure III. 8:Binary Fire Mask. 48
Figure III. 9:Segmentation Fire Mask 48
Figure III. 10:Loading dataset to run the automatic custom algorithm.. 48
Figure III. 11:Sample frames of fire bounding boxes automated labeling results. 49
Figure III. 12:FLAME Dataset Lebel Summary Graph. 50
Figure III. 13:Ground Truth bounding boxes coordinates tables. 50
Figure III. 14:Ground Truth bounding boxes coordinates Sample. 51
Figure III. 15:Sample Frames from ALL New Dataset for Ground Truth testing. 51
Figure III. 16:Sample Frames from Binary Mask testing results. 52
Figure III. 17:Sample Frames from Bounding Box testing results. 52
Figure III. 18:The YOLO network architecture [25] 54
Figure III. 19:Network Architecture Layers. 56
.Figure III. 20:Network Layers Analysis 57
Figure III. 21:YOLOv2 Detector Properties 60
Figure III. 22:Training Progress's information for the first 10 epochs. 61
Figure III. 23:Training Progress's information for the last epochs. 61
Figure III. 24:Neural Network Loss Visualization [43]. 62
Figure III. 25:Training Loss Graph 63

Chapter4

Figure IV. 1:Project work Methodology 66
Figure IV. 2:Sample Frames from Testing the first Detector 68
Figure IV. 3:Sample Frames from Testing the second Detector 69
Figure IV. 4:Sample Frames from Testing the Third Detector 70
Figure IV. 5:Intersection over Union [46] 71
Figure IV. 6:Log Average Miss Rate of the first scenario 74
Figure IV. 7:Log Average Miss Rate of the first scenario 74
Figure IV. 8:Log Average precision of the seconde scenario/ Log Average Miss Rate of the seconde scenario 74
Figure IV. 9:Log Average Miss Precision of the Third scenario/ Log Average Miss Rate of the Third scenario 75
Figure IV. 10:Screenshot from the shooting phone 77
Figure IV. 11:MATLAB live processing window for detection with 27.28 FPS 78
Figure IV. 12:MATLAB live processing window for detection with 41.08 FPS 78

List of Tables

Chapter1

Table I.1:Total reported fire statistical Data by country, 1993-2018 [6]. 6
Table I. 2:Common indicators of fire statistics in the countries of the world in 2018 [6]. 7

Chapter3

Table III. 1:Flame Dataset Information Table [23]. 39
Table III. 2:Summarization of the FiSmo-Images Datasets [24]. 40
Table III. 3:All-New Dataset sources information. 42
Table III. 4:computational characteristics of the YOLOv2 [25] 55
Table III. 5:The dataset distribution Table 58

Chapter4

Table IV. 1:Models trained Dataset information Table 67
Table IV. 2:summary of the three secanrios results 76

Acronyms

UAV	Unmanned Aerial Vehicle
LIDAR	Light Detection and Ranging
IR	Infrared
CCD	Charge Coupled Device
NN	Neural Network
CNN	Convolutional Neural Network
R-CNN	Recurrent Convolutional Neural Network
RNN	Recurrent Neural Network
ANN	Artificial Neural Network
DNN	Deep Neural Network
LTSM	Long Term Semantic Memory
SVM	Support Vector Machines
RELU	Rectified Linear Unit
AI	Artificial Intelligent
ML	Machine Learning
DL	Deep Learning
FLAME	Fire Luminosity Airborne-Based Machine Learning Evaluation
YOLO	You Only Look Once
YOLOv2	You Only Look Once Version 2
VGG	Visual Geometry Goupe
RGB	Red, Green, Blue
ROI	Region of Interest
ADAM	Adaptive Moment Estimation
CPU	Central Processing Unit
GPU	Graphics Processing Unit
IOU	Intersection Over Union
TP	True Positive

FP	False Positive
FN	False Negative
TN	True Negative
mAP	Mean Average Precision
mr	Miss Rate
fppi	False Positive Per Image
FPS	Frame Per Second
GB	Gigabyte
GHZ	Gigahertz
LAMR	Log Average Miss Rate

General Introduction

Threat to people's lives and property caused by fires has become increasingly serious in particular, the increasing large-scale forest fire events around the world have made the development of automatic wildfire detection as an urgent need for fire warning systems. Traditionally, forest fires were mainly detected by human observation; however, this approach is inefficient, as it is prone to human error and fatigue. On the other hand, conventional sensors for the detection of heat, smoke, flame, and gas typically take time for the particles to reach the point of sensors and activate them., hence, a large number of sensors need to be installed to cover large areas, in order to tackle these disadvantages more advanced automatic forest fire detection methods are developed using satellites, ground-based equipment's, and manned/unmanned aerial vehicles (UAVs). In particular, recent advances in aerial monitoring systems can provide first responders and operational forces with more accurate data on fire behavior for enhanced fire management, the use of unmanned aerial vehicles UAVs for fire monitoring is gaining more attraction in recent years. moreover, UAVs offer new features and convenience including fast deployment, high maneuverability, wider and adjustable viewpoints, and less human intervention. Recent studies investigated the use of UAVs in disaster relief scenarios and operations such as wildfires and floods, particularly as a temporary solution when terrestrial networks fail due to damaged infrastructures, communication problems, spectrum scarcity, or coalition.

As the technology of fire detection by the camera has been gradually applied with the development of computer vision technology and artificial intelligence, deep learning vision-based detection techniques can supply intuitive and real-time data, detect wide range objects, and make record conveniently, it has become a crucial element in the UAV-based forest fire detection systems.

In this project we aim to respond to the following queries:

How can we build a UAV-based forest fire detector using deep learning technologies? How can we balance between the detection performance and real-time requirements?

In order to provide answers to these questions, we have structured our work in four chapters:

Chapter1: in this chapter we first introduce the Problem of forest fire detection and its statistics, we then expose the different methods and systems that are used for detection and discuss the previous conducted work in this field (State of the Art).

Chapter2: the second chapter is devoted to present generalities about deep learning approach in computer vision, as well as notions about object detection and popular convolutional neural networks models and architectures.

Chapter3: in the third chapter we discussed the conducted process of crafting a deep computer vision model, from dataset creation and labeling to YOLOv2 network design and training.

Chapter 4: in this chapter we try to test our built detector and evaluate its performance for real-time fire detection.

We will end this brief with a general conclusion which summarizes the main ideas that we have detailed and we will propose recommendations for the continuity of this work (perspectives).

Chapter I: Introduction to Forest Fire Detection

1. Introduction:

Forest fires can potentially result in a great number of environmental disasters, causing vast economic and ecological losses as well as endangering human lives. In order to preserve natural resources and protect human safety and properties, forest fire detection and accurate monitoring of the disturbance type, size, and impact over large areas is becoming increasingly important and attracting an enormous interest around the world.

Recent advances in computer vision, machine learning and deep learning offer new tools for detecting and monitoring forest fires, while the development of new materials and microelectronics have allowed sensors to be more efficient in identifying active forest fires.

In this chapter we will try to cover up some statistics about forest fires, and the utilized methods and systems for detection, we will also take an overview of the state of the art to get an idea on the work done on this detection problem.

2. Statistics on Forest Fire

2.1.Forest fire Worldwide:

The statitistics presneted by the Center of Fire Statistics (CFS) of the International Association of Fire and Rescue Services (CTIF) in its latest report №25, of many CTIF countries and their larger cities for 2018 and trends for 2014-2018 where Data from 46 nations and 41 cities was used to compile the 2018 fire and loss statistics. Data on calls, fires, and losses from 2014 to 2018 comes from 65 nations, each of which provided data for one or more of the five years. The number of countries whose data is displayed in each table fluctuates depending on the number of countries that supplied data for the given statistics at any point between 2014 and 2018. Statistics on fire service resources are offered for 59 nations.

Table I .1 below shows total fire statistics from 1993 to 2018 for 27-57 countries, collectively representing 0.9-3.8 billion inhabitants of the Earth, depending on the year of reporting. In these countries, 2.5-4.5million fires and 17-62 thousand fire deaths were reported to fire services, depending on the year. [6]

Year	Number of countries	Total population, bln.inh.	Number of fires, mln.	Number of fire deaths, thous.	Average numberof fires per 1000 inh.	Average number of fire deaths	
						per 100000 inh.	per 100 fires
1993	39	2,4	3,9	30,2	1,6	1,3	0,8
1994	27	1,1	4,0	29,5	3,6	2,7	0,7
1995	42	1,2	4,5	32,5	3,8	2,7	0,7
1996	43	0,9	4,0	29,1	4,4	3,2	0,7
1997	48	2,8	3,7	57,7	1,3	2,1	1,6
1998	47	3,0	3,6	51,7	1,2	1,7	1,4
1999	52	3,1	3,9	51,8	1,3	1,7	1,3
2000	57	3,3	4,5	56,2	1,4	1,7	1,2
2001	46	3,5	3,8	61,9	1,1	1,8	1,6
2002	41	3,5	4,3	62,3	1,2	1,8	1,4
2003	39	3,5	4,5	61,1	1,3	1,7	1,4
2004	44	3,5	4,1	60,1	1,2	1,7	1,5
2005	45	3,5	4,3	57,4	1,2	1,6	1,3
2006	37	3,6	4,1	52,2	1,1	1,5	1,3
2007	40	3,8	4,0	52,5	1,1	1,4	1,3
2008	31	3,5	3,6	48,3	1,0	1,4	1,3
2009	31	3,4	3,3	44,7	1,0	1,3	1,4
2010	33	2,2	3,2	46,1	1,5	2,1	1,4
2011	34	2,3	3,3	48,2	1,4	2,1	1,5
2012	35	1,1	3,1	23,7	2,8	2,2	0,8
2013	31	1,1	2,5	21,7	2,3	2,0	0,9
2014	32	1,1	2,7	20,7	2,5	1,9	0,8
2015	31	1,0	3,5	18,4	3,5	1,8	0,5
2016	39	1,1	3,0	18,0	2,7	1,6	0,6
2017	34	1,1	3,2	16,9	2,9	1,5	0,5
2018	46	2,7	4,5	30,8	1,7	1,1	0,7
Avrg	**39**	**2,4**	**3,7**	**41,7**	**1,5**	**1,7**	**1,1**
Total			**97,1**	**1083,7**			

Table I.1:*Total reported fire statistical Data by country, 1993-2018 [6].*

Table I .2 below shows that in 46 countries, representing 2.7 bln inhabitants, 36% of the World's population, 50 million calls (18.1 calls per 1000 inh.) 4,6 million fires(9.2% of all calls, 1.7 fires per 1000 inh.) 30.8 thousand civilian fire deaths (1.1 firedeaths per 100 thous inh.) and 51.3 thousand civilian fire injuries (1.9 fire injuries per 100 thous inh.) reported by fire services in 2018 [6]**.**

№	Country	Population, thous.inh.	Number of				Average number:					
			calls	fires	fire deaths	fire injuries	Per 1000 inh.:		fire deaths per:		fire injuries per:	
							calls	fires	100000 inh.	100 fires	100000 inh.	100 fires
1	India	1 359 000	-	1 600 000	12 747	-	-	1,2	0,9	0,8	-	-
2	USA	327 167	36 746 500	1 318 500	3 655	15 200	112,3	4,0	1,1	0,3	4,6	1,2
3	Bangladesh	162 951	-	19 642	130	664	-	0,1	0,1	0,7	0,4	3,4
4	Russia	146 781	760 653	144 199	7 913	9 650	5,2	1,0	5,4	5,5	6,6	6,7
5	Philippines	106 700	-	16 675	326	-	-	0,2	0,3	2,0	-	-
6	Vietnam	95 990	-	4 182	90	208	-	0,0	0,1	2,2	0,2	5,0
7	France	66 628	4 942 906	305 500	262	1 282	74,2	4,6	0,4	0,1	1,9	0,4
8	Great Britain	64 553	695 101	204 525	400	8 944	10,8	3,2	0,6	0,2	13,9	4,4
9	Italy	61 000	908 887	213 116	-	-	14,9	3,5	-	-	-	-
10	Republic of Korea	51 629	2 656 700	42 338	369	2 225	51,5	0,8	0,7	0,9	4,3	5,3
11	Ukraine	42 270	230 952	78 602	1 967	1 516	5,5	1,9	4,7	2,5	3,6	1,9
12	Poland	38 411	502 200	149 434	527	4 335	13,1	3,9	1,4	0,4	11,3	2,9
13	Peru	32 000	121 998	13 729	-	-	3,8	0,4	-	-	-	-
14	Kazakhstan	18 611	55 102	14 557	434	412	3,0	0,8	2,3	3,0	2,2	2,8
15	Netherlands	17 181	148 900	76 020	52	-	8,7	4,4	0,3	0,1	-	-
16	Greece	10 788	65 298	24 459	131	187	6,1	2,3	1,2	0,5	1,7	0,8
17	Czech Republic	10 650	-	20 720	100	1 466	-	1,9	0,9	0,5	13,8	7,1
18	Jordan	10 378	56 326	24 490	24	1 058	5,4	2,4	0,2	0,1	10,2	4,3
19	Sweden	10 230	133 955	31 376	73	390	6,7	1,9	1,0	0,2	3,8	1,2
20	Hungary	9 778	68 337	19 355	106	832	7,0	2,0	1,1	0,5	8,5	4,3
21	Belarus	9 475	52 974	6 435	525	311	5,6	0,7	5,5	8,2	3,3	4,8
22	Austria	8 837	278 672	43 554	-	-	31,5	4,9	-	-	-	-
23	Switzerland	8 500	77 304	13 178	-	-	9,1	1,6	-	-	-	-
24	Bulgaria	7 050	56 120	29 448	145	285	8,0	4,2	2,1	0,5	4,0	1,0
25	Denmark	5 786	42 876	15 081	71	-	7,4	2,6	1,2	0,5	-	-
26	Singapore	5 612	191 492	3 885	4	90	34,1	0,7	0,1	0,1	1,6	2,3
27	Kyrgyzstan	5 522	-	4 808	59	54	-	0,9	1,1	1,2	1,0	1,1
28	Finland	5 483	113 464	14 264	58	670	20,7	2,6	1,1	0,4	12,2	4,7
29	Slovakia	5 450	31 326	9 288	49	194	5,7	1,7	0,9	0,5	3,6	2,1
30	Costa Rica	4 973	41 881	23 862	30	73	8,4	4,8	0,6	0,1	1,5	0,3
31	Ireland	4 920	242 631	28 534	18	-	49,3	5,8	0,4	0,1	-	-
32	New Zealand	4 748	82 136	18 580	-	-	17,3	3,9	-	-	-	-
33	Oman	4 298	-	4 602	-	-	-	1,1	-	-	-	-
34	Croatia	4 087	22 927	9 968	23	117	5,6	2,4	0,6	0,2	2,9	1,2
35	Mongolia	3 238	-	3 612	64	-	-	1,1	2,0	1,8	-	-
36	Lithuania	2 848	22 142	11 848	315	387	7,8	4,2	11,1	2,7	13,6	3,3
37	Qatar	2 839	3 125	1 922	2	115	1,1	0,7	0,1	0,1	4,1	6,0
38	Slovenia	2 081	153 313	4 119	7	284	73,7	2,0	0,3	0,2	13,6	6,9
39	Latvia	1 950	-	9 134	81	301	-	4,7	4,2	0,9	15,4	3,3
40	Estonia	1 317	26 163	5 353	50	100	19,9	4,1	3,8	0,9	7,6	1,9
41	Mauritius	1 300	12 634	6 664	-	-	9,7	5,1	-	-	-	-
42	Bhutan	817	-	100	3	0	-	0,1	0,4	3,0	0,0	0,0
43	Luxemburg	602	61 157	2 228	0	-	101,6	3,7	0,0	0,0	-	-
44	Brunei	442	-	1 249	2	1	-	2,8	0,5	0,2	0,2	0,1
45	Barbados	277	-	1 925	-	-	-	6,9	-	-	-	-
46	Liechtenstein	38	-	42	0	0	-	1,1	-	-	-	-
Total		**2 745 186**	**49 606 152**	**4 595 102**	**30 812**	**51 351**	**18,1**	**1,7**	**1,1**	**0,7**	**1,9**	**1,1**

Table I. 2:*Common indicators of fire statistics in the countries of the world in 2018 [6].*

2.2.Forest fire in Algeria:

Algeria is one of the few countries with statistics on forest fires over a period of more than a century. *Figure* I. 1 represents a 137-year time series of areas covered.

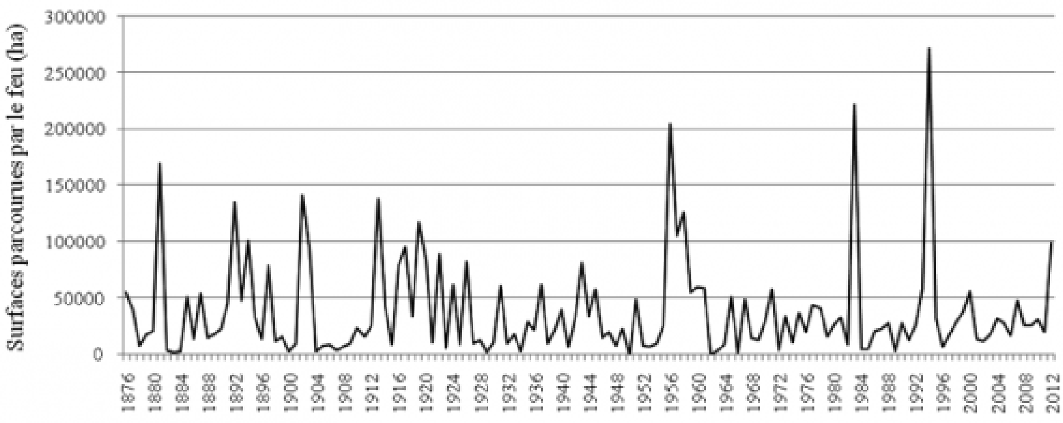


Figure I. 1:*Annual evolution in areas covered by fire in Algeria (period 1876 -2012) [20].*

From 1985 to 2012, the annual evolution reveals the variability of forest fires and especially the existence of catastrophic years in 1993, 1994, 2000, 2007 and 2012, While a large majority of fires (82.1%) are brought under control before the area covered exceeds 10 ha, 3.2% of fires affect areas of more than 100 ha, of which 0.6% are greater than 500 ha. This last category represents 272 fires (including 109 for 1994 alone) [20].

If we go back in time, we see that the colonial period was disastrous: a cumulative area of 3,506,942 ha was covered by fire, over a period of 87 years (1876-1962), i.e. an average of 41,258 ha / year. The catastrophic balances, of more than 100,000 ha / year (exceptionally more than 150,000, even 200,000 ha), in 1881, 1892, 1894, 1902, 1913, 1919, 1956, 1957 and 1958, mark dark years which generally coincide with troubled times [20].

After independence, the affected areas declined slightly, with an average of 35,315 ha/year over the period 1963-2012. This did not prevent the occurrence of new dark years in 1965, 1967, 1971, 1977, 1978, 1993, 2000, 2007 and 2012. Three of them were particularly catastrophic: 1983, 1994 and 2012 with 221,367 ha respectively. 271,598 ha and 99,061 ha covered. These three years alone amount to nearly 600,000 ha of burnt areas, or 34% of the total for the period 1963-2012. Such "out of the ordinary" surfaces can of course be favoured, at least in large part, by climatic conditions very favourable to the outbreak and propagation of fire, but they depend essentially on the human factor: prior disorder, poor management and above all instability [20].

It has long been known that, in times of political issues, Algerian forests always pay a heavy price for fires. Moreover, F. Ramade (1997) stigmatizes the political disorders which, as in

Algeria, have been "since 1992 at the origin of several fires which have devastated vast forests, in particular in Kabylia" [20].

Spatially, we note that the risk of forest fires is mainly concentrated in the coastal wilayas of northeast Algeria, from Tizi Ouzou to El Tarf (Figure I .2), corresponding to very wood and hilly wilayas, with a high population density and a lack of land for urbanization.

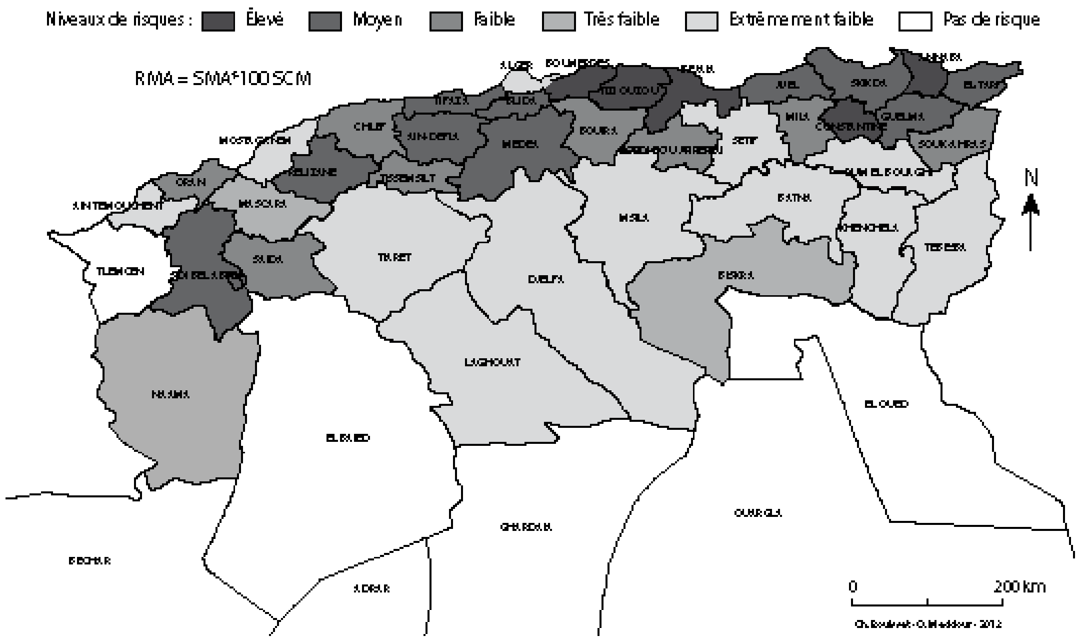


Figure I. 2:Average annual fire risk in Algeria [20].

We can say that Algeria is one of the countries where the problem of forest fires is relatively unknown by the Scientific community: if in absolute value the areas burned remain remotely modest compared to other countries around the Mediterranean, the scarcity of forests and threats of desertification mean that these fires have a particularly disastrous. Algeria has 4.1 million hectares of forests, or an afforestation rate of 1.76%. However, the close frequency of fires that follow one another with a return interval of less than 10 years has a catastrophic ecological impact.

The analysis of fires during the period 1985-2016, at the level of the 40 wilayas of Algeria from North (the most wooded part) shows that 42,555 fires covered a total forest area of more than 910,640 hectares [20].

3. Forest Fire Detection Existing Methods:

3.1. Using Sensors:

Data Sensors are now used in practically every fire alarm system. The system's improvement is determined by the sensor's accuracy, dependability, and spatial distributions. Large numbers of sensors are needed in open-air settings for high-accuracy fire detection systems. Sensors also require a continuous battery charge, which is impossible to do in a vast open environment. If there is a fire nearby, sensors will detect it. As a result, the sensor will be damaged. For fire detection, camera surveillance, and wireless sensor networks, two alternative types of sensor networks are now available. Different types of detection sensors can be used in terrestrial systems.

- video-camera, responsive to visible spectrum of smoke noticeable during the day and a fire recognizable at night.
- Infrared (IR), thermal imaging cameras based on the detection of heat flow of the fire,
- IR spectrometers to identify the spectral characteristics of smoke.
- Light detection and ranging systems— LIDAR (detection of light and range) that compute laser rays reflected from the smoke particles [2].

The several optical systems, which work according to different algorithms devised by the builders, all share the same general notion in terms of detecting smoke and fire glow. Simply put, the camera creates visual sequences. The processing unit tracks motion in photos and checks how many pixels contain smoke or fire light, then passes the results to another algorithm to decide whether or not to trigger an alarm for the operator.

3.2.Using Computer Vision

These replace conventional fire detection systems, due to the rapid development of digital camera technology and video processing. Computer vision-based systems use three stages.

- Flame pixel classification.
- Segmentation of moving object.
- Analysis of the candidate region.

The fire detection system's performance is determined by the fire pixel classifier, which generates main areas on which the rest of the system operates. As a result, a highly accurate fire pixel classifier with a low false detection rate is required. A video flame detection system that uses background subtraction and color analysis to find probable flame regions on the video frame and then uses a collection of fire extraction attributes, including color probability, to distinguish between fire and non-fire objects.

Although some methods directly deal with fire pixel categorization, there are some that deal with spatial variance, temporal variation, and contour variability of candidate blob regions.
In both greyscale and color video sequences, the fire pixel categorization can be evaluated.

3.3.Using Deep Learning:

Deep learning has recently been successfully used to a variety of disciplines, including image object detection and classification, audio recognition, and natural language processing. To increase performance, researchers have done a number of studies on fire detection using deep learning.
The deep learning approach has several differences from the conventional computer vision-based Fire detection. The first is that the features are not explored by an expert, but rather are automatically captured in the network after training with a large amount of diverse training data. Therefore, the effort to find the proper handcrafted features is shifted to designing a proper network and preparing the training data [3].
Another distinction is that the detector/classifier can be obtained by training features in the same neural network at the same time. As a result, with an efficient training method, the right network structure becomes even more essential.

4. Forest Fire Detections Existing Systems

The forest fire detection systems can be devised into 3 major group of systems: terrestrial systems, UAV systems, Satellite systems.

4.1.Terrestrial Systems

Terrestrial-based early detection systems consist of either individual sensors (fixed, PTZ, or 360_ cameras) or networks of ground sensors [1].to provide adequate visibility, these sensors must be carefully placed. As a result, they're generally found in watchtowers, which are structures built on high view points to monitor high-risk scenarios and can be utilized not just for fire detection but also for verification and localization. There are two types of cameras used for early fire detection: optical cameras and infrared cameras, which may gather data with resolutions ranging from low to ultra-high for various fire detection scenarios.
Early detection devices that combine the two types have lately been introduced. Computer-based solutions can process a large amount of data while retaining a high level of accuracy and a low false alarm rate.

4.2.Unmanned Aerial Vehicles Systems

Terrestrial imaging systems can detect both flame and smoke, but in many cases, it is almost Impossible to view, in a timely manner, the flames of a wildfire from a ground-based camera or a mounted camera on a forest watchtower. To this aim, autonomous unmanned aerial vehicles (UAVs) can provide a larger and more accurate view of the fire from above, even in regions that are inaccessible or too risky for firefighting crews to operate in. Fixed or rotary-wing UAVs cover a larger area and are more flexible, allowing for changes in monitoring area, although they are subject to weather and have a limited flying length. Unmanned aerial vehicles (UAVs) devoted to forest fire monitoring and detection are in high demand due to their rapid manoeuvrability and improved personnel safety. A typical UAV-based forest fire surveillance system is illustrated in Figure I. 2 Which is composed of a team of UAVs, different kinds of on-board sensors, and a central ground station, The goal is to use UAVs to identify and track flames, predict their spread, and provide real-time fire information to human firefighters, as well as to use UAVs to suppress fires. The system may perform fire monitoring (search for a prospective fire), detection (identify a potential fire and alert firefighting personnel), diagnosis (calculate parameters of the fire position, extent, and evolution), and prognosis (predict the outcome of the fire) (predict the fire propagation).

As can be observed, one of the most significant parts of the UAV-based forest fire monitoring system is the computer vision-based fire detection technique. This is due to its multiple advantages, including the ability to monitor a wide range of objects, provide intuitive and real-time images, and conveniently record information.More specifically, charge-coupled device (CCD) Cameras and infrared (IR) cameras are usually mounted on UAVs. Massive efforts have been dedicated to the development of more effective image processing scheme for fire detection.

Color and motion aspects in CCD camera visual images are typically used for fire detection. However, in some outdoor applications, the use of CCD cameras is commonly considered to be insufficiently durable and trustworthy.

Given highly complex, non-structured forest environments, the possibility of smoke covering the fire, or the circumstance for analogues of fire such as reddish leaves swinging in the wind and light reflections, the false fire alert rate is typically quite high.Due to the fact that IR images can be obtained in either weak or no light situations, or smoke can be seen as transparent in IR images, Even though IR cameras are more expensive than CCD cameras, they are extensively used to capture monochromatic images in both the day and the night. The use of this successful method is predicted to lower the rate of false fire alarms and improve the forest fire detection system's adaptive capabilities in various operating conditions.

4.3.Space borne (Satellite) Systems

Satellite detection is the most common detection techniques used by authorities across the Globe.

Recently, mainly due to the large number of satellites launched and the decrease of associated Costs, there are many research efforts to detect forest wildfires from satellite images. Specifically, a set of satellites were designed for Earth observation (EO, e.g., environmental monitoring or meteorology). Satellites are categorised into several kinds based on their orbit, each with its own set of benefits and drawbacks. The most important categories are: (a) geostationary orbit (GEO), which is a circular orbit with an altitude of 35,786 kilometers and zero inclination, so that the satellite does not move at all relative to the ground, providing a constant view of the same surface area; (b) low Earth orbit (LEO), which has an altitude of 2000 km or less, and requires the least amount of energy for satellites placement and provides high bandwidth and low communication latency, (c) the polar sun-synchronous orbit (SSO), which is a roughly polar orbit that crosses around the equator at the same local time each time. Most EO satellites are in low Earth polar SSO orbits, with exact altitude and inclination calculations to ensure that the satellite always observes the same scene with the same angle of light from the Sun, and that shadows seem the same on each pass.

Sun-synchronous satellite data has a high spatial resolution but a low temporal resolution, while geostationary satellite data has a high temporal resolution but a low spatial resolution. Some satellites in the first group, such as Landsat or Sentinel, have a long revisit period (LandSat-7/8 has an eight-day repeat cycle, while Sentinel 2A/2B has a two-to-three-day repeat cycle at mid-latitudes). As a result, they are not suited for real-time active forest fire detection, but only for tasks that are less time-sensitive, such as burnt area estimation.

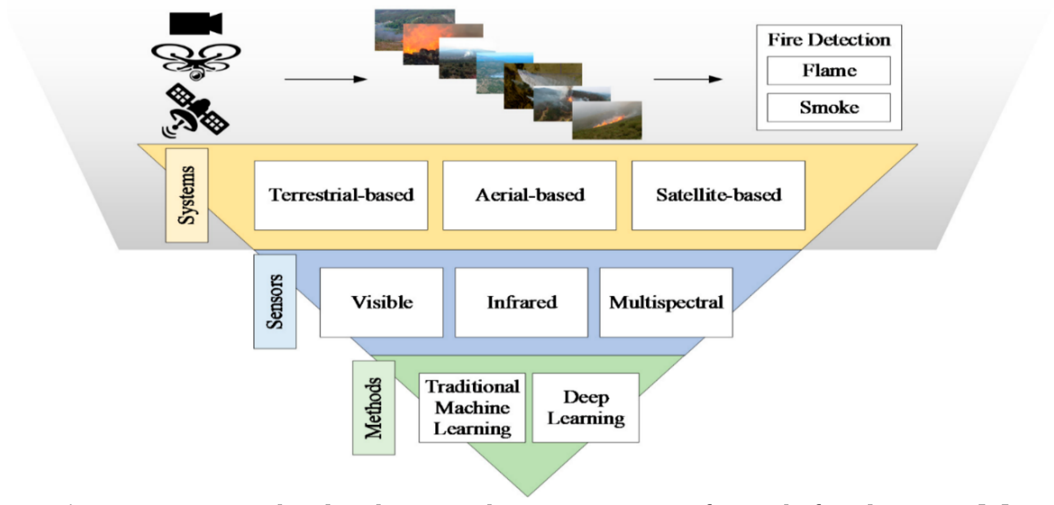


Figure I. 3:*Generalized multispectral imaging systems for early fire detection [1].*

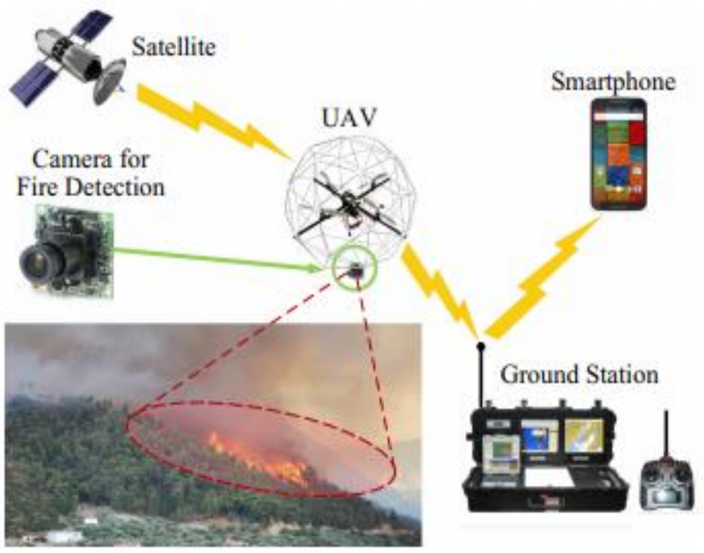


Figure I.4:*Schematic Illustration of the UAV Based Forest Fire Detection System [4].*

5. State of the Art

5.1.Computer Vision-Based Traditional Approach

With the development of computer vision technology, the technology of fire detection by the camera has gradually applied in our life. The original temperature and smoke sensors fall far short of some complex environments needs. Most of the researchers sensors detect the fire by using the colour characteristics and motion characteristics and the results got disappointing in the complex environment. It is significant in studying how to extract the characteristics of the weak fire, how to identify the weak fire accurately and how to prevent and control the fire alarm. Therefore, the researchers have proposed various methods of fire detection.

In conventional fire detection, much research has continuously focused on finding out the salient Features of fire images. Chen [8] analysed the changes of fire using an RGB and HSI colour model based on the difference between consecutive frames and proposed a rule-based approach for fire decision.

Celik and Demirel [7] proposed a generic rule-based flame pixel classification using the YCbCr colour Model to separate chrominance components from luminance ones. In addition, Wang [9] extracted the candidate fire area in an image using an HSI colour model and calculated the dispersion of the flame colour to determine the fire area. However, colour-based fire detection methods are generally vulnerable to a variety of environmental factors such as lighting and shadow.

Borges and Izquierdo [10] adopted the Bayes classifier to detect fires based on additional features Such as the area, surface, and boundary of the fire area to colour. In addition, Foggia [11] proposed a multi-expert system which combines the analysis results of a fire's colour, shape, and motion characteristics. Although insufficient, the supplementary features to colour, including texture, shape, and optical flow, can reduce the false detections.

Nevertheless, these approaches require domain knowledge of fires in captured images essential to Explore hand-crafted features and cannot reflect the information spatially and temporally involved in Fire environments well. In addition, almost all methods using the conventional approach only use a Still image or consecutive pairs of frames to detect fire. Therefore, they only consider the short-term Dynamic behaviour of fire, whereas a fire has a longer-term dynamic behaviour [3].

5.2.Computer Vision-Based Deep Learning Approach

In recent years, deep learning has been widely used in a myriad of computer vision applications because of its high recognition capability. To the best of our knowledge, Gunay et al. [12] is the first paper to use deep learning in dynamic texture recognition including wildfires. In early deep Learning based forest detection, researchers designed blank convolutional neural networks and trained them with collected or synthesized images. for example, Zhao et al [13]. Designed a 15-layer Convolutional neural network (CNN) to detect the forest fire. What is more, to locate the fire and Smoke in frames [5], Sebastien [14] proposed a fire detection network based on CNN where the features are simultaneously learned with a Multilayer Perceptron (MLP)-type neural net classifier by training.

Zhang et al. [15] also proposed a CNN-based fire detection method which is operated in a cascaded fashion. In their method, the full image is first tested by the global image-level classifier, and if a fire is detected, then a fine-grained patch classifier is used for precisely localizing the fire patches.

Muhammad et al. [16] proposed a fire surveillance system based on a fine-tuned CNN fire detector. This architecture is an efficient CNN architecture for fire detection, localization, and semantic understanding of the scene of the fire inspired by the Squeeze Net architecture. In the deep layer of CNN, a unit has a wide receptive field so that its activation can be treated as A feature that contains a large area of context information. This is another advantage of the learned Features with CNN for fire detection. Even though CNN showed over whelmingly superior classification performance against traditional computer vision methods, locating objects has been another problem. [3]

Although the CNN-based approaches provide excellent performance, it is hard to capture the Dynamic behaviour of fire, which can be obtained by recursive-type neural networks (RNN). LSTM Proposed by Hochreiter and Schmidhuber [17] is an RNN model that solves the vanishing gradient Problem of RNN. LSTM can accumulate the temporal features for decision making through the memory cells which preserve the internal states and the recurrent behaviour. However, the number of recursions is usually limited, which makes it difficult to capture the long-term dynamic behaviour necessary to make a decision. Therefore, special care must be taken to consider the decision based on long-term behaviour with LSTM.

Recently, Hu et al. [18] used LSTM for fire detection, where the CNN features are extracted from Optical flows of consecutive frames, and temporally accumulated in an LSTM network. The final Decision is made based on the fusion of successive temporal features. Their approach, however, computes the optical flow to prepare the input of CNN rather than directly using RGB frames.

6. Conclusion:

We have devoted this chapter to the presentation of some generalities on Forest fire detection systems and Methods while presenting a literature review on the most important work and research papers that was published during the last decade, as we discussed above the use of a deep learning method and a UAV based system has been considered as a powerful Approach solution in the recent years. In order to detect forest fire with great precision and a decreasing rate of false fire alarm, a model of neural networks is trained using a UAV Based Aerial Images dataset. The next chapter is dedicated to present generalities about deep learning approach in computer vision, as well as notions about object detection and CNNs.

Chapter II: Deep Learning for Computer Vision

1. Introduction

Deep Learning is a branch of artificial intelligence and Data Science. This field has undergone immense development, especially in recent years. Big companies like Google, Facebook, Microsoft and Amazon are working on this topic.

Our job is to find a solution for the detection of forest fire through the use of deep learning in computer vision on board of a UAV.

In this chapter we give general information about Artificial Intelligence, Machine Learning and Deep Learning technologies, then we present convolutional neurons, computer vision and object recognition, finally we get a closer look on the most popular Deep Learning model families.

2. Artificial Intelligence, Machine Learning and Deep Learning

2.1. Artificial Intelligence

2.1.1 Definition :

Artificial Intelligence is the science of making computers behave like humans in terms of making decisions, text processing, translation, etc. AI is a big umbrella that has Machine Learning and Deep Learning under it [31]. Figure II. 1 below illustrate clearly the correlation between the three.

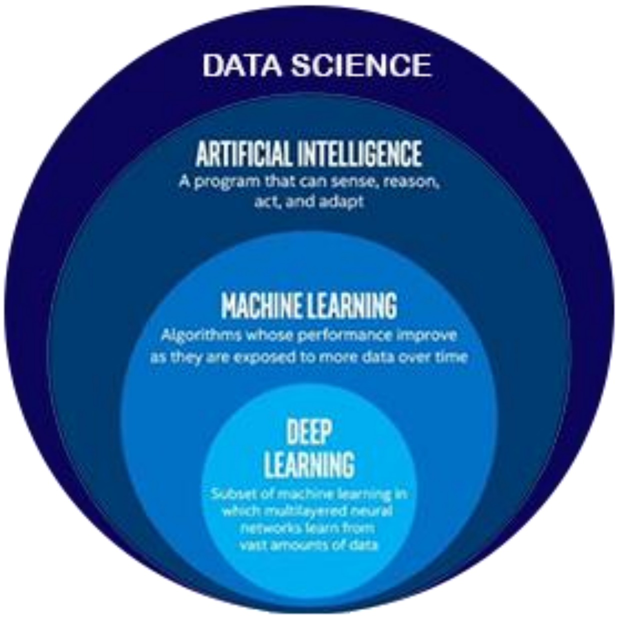

DATA SCIENCE
ARTIFICIAL INTELLIGENCE
A program that can sense, reason, act, and adapt
MACHINE LEARNING
Algorithms whose performance improve as they are exposed to more data over time
DEEP LEARNING
Subset of machine learning in which multilayered neural networks learn from vast amounts of data

Figure II. 2:*Artificial Intelligence, Machine Learning, Deep learning [35].*

2.1.2 Artificial Intelligence Approaches:

Historically, four approaches to AI have been followed, each by different people with different methods. a human-centered approach must in part be an empirical science, involving observation and hypothesis about human behavior. A rational approach involves combination of mathematics and engineering. The different groups described themselves as follows:

- Acting humanly**:** Turing Machine Approach [34].
- Thinking humanly**:** The Cognitive Modeling Approach [34].
- Thinking rationally**:** The Laws of Thought Approach [34].
- Acting rationally**:** The Rational Agent Approach [34].

2.2.Machine Learning

2.2.1. Definition :

Machine learning is a subset of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer program**s** that canaccess data and use it to learn for themselves [33].

2.2.2. Different types of Machine Learning :

Machine Learning can be divided into 3 categories:

A. ***Supervised Learning***: we give the algorithm labeled data and the algorithm has to learn from it and figure out how to solve future similar problems. It's like giving the algorithm problems and answers, the algorithm has to learn how these problems were solved in order to solve future problems in a similar manner.

B. ***Unsupervised Learning*****:** we give the algorithm a problem without any labeled data or any prior knowledge of what the answer could be. It's like giving the algorithm problems without any answers, the algorithm has to find the best answer by driving insights from the data.

C. ***Reinforcement Learning*****:** is the training of machine learning models to make a sequence of decisions. The agent learns to achieve a goal in an uncertain, potentially complex environment. In reinforcement learning, an artificial intelligence faces a game-like situation. The computer employs trial and error to come up with a solution to the problem. To get the machine to do what the programmer wants, the artificial intelligence gets either rewards or penalties for the actions it performs. Its goal is to maximize the total reward.

The three types can be summarized as it is shown in Figure II.3 below.

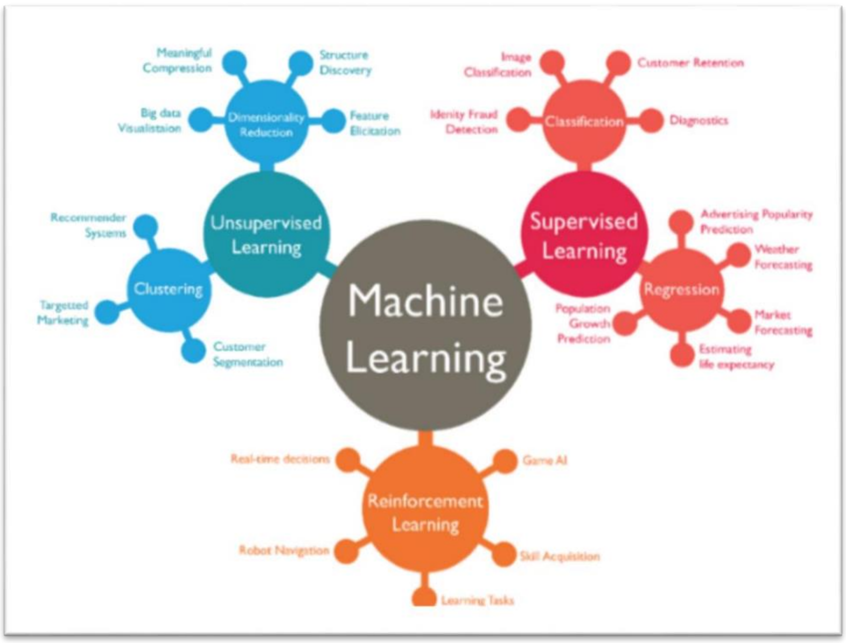


Figure II.4:*Types of Machine Learning [33].*

2.2.3. Elements of Machine Learning :

To use machine learning, two elements are essential: Model and Dataset.

A. Model:

- *Classification Model:* classification is a model of ML whose outputs belong to a finite set of values (example: good, average, bad). Some of the most common classification models: Logistic regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNN) and neural networks.
- *Regression model:* is a model of ML whose outputs are numbers (example: the temperature of tomorrow), some of the most common regression models: Linear regression, Decision Tree, Random Forest and neural networks.

B. Datasets:

Datasets correspond to one or more database tables, where each column of a table represents a particular variable and each row corresponds to a given record in the dataset. In question, the dataset lists the values of each of the variables, such as the height and weight of an object, for each member of the dataset. The dataset is often separated into three parts, namely:

- *Training data:* is that which is taken to refine the parameters of the model and thus allow it to generalize towards unknown data.
- *Validation data:* is used during training to estimate the performance of the modeland

help to select the hyper-parameters. In addition, it can be used to determine when it is most beneficial to stop training (early stopping).

- *Test data:* used to obtain the results and to evaluate the generalization capacities ofthe model.

2.3.Deep Learning :

2.3.1. Definition :

Deep learning is a machine learning technique that teaches computers to accomplish things that people do naturally. For example, a computer model can learn to execute classification tasks directly from images, text, or sound in deep learning. Deep learning models can attain state-of-the-art accuracy, even surpassing human performance in some cases. Models are trained utilizing a huge quantity of labeled data and multilayer neural network topologies.

2.3.2. Deep Learning VS Machine Learning:

Deep learning is a type of machine learning that is very specialized. The first step in a machine learning workflow is to manually extract useful characteristics from images. The features are then utilized to build a classification model for the items in the image. Relevant features are automatically retrieved from photos using a deep learning approach. Furthermore, deep learning accomplishes "end-to-end learning," in which a network is given raw data and a task to complete, such as classification, and it automatically learns how to do so, Figure II.5 below summaries perfectly the feature extraction difference.

Another significant distinction is that deep learning algorithms scale as data grows (Figure II.4), whereas shallow learning techniques converge. Machine learning algorithms that plateau at a given level of performance as more instances and training data are added to the network are referred to as shallow learning. Deep learning networks have the advantage of improving as the size of your data grows.

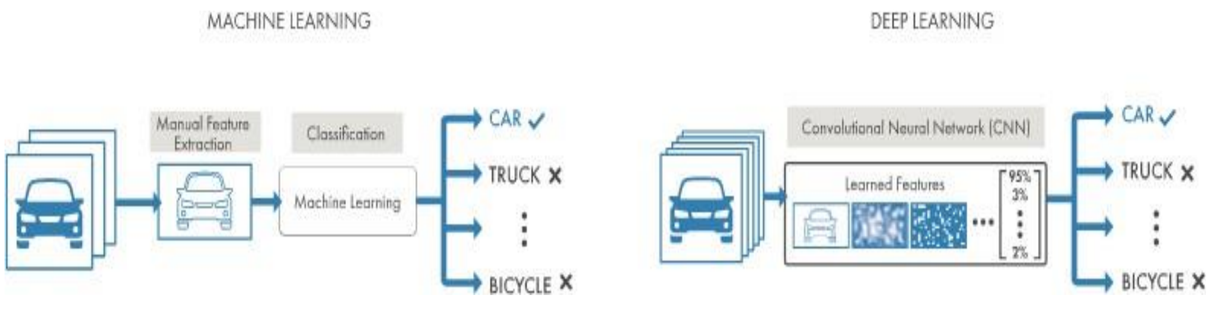


Figure II. 6:*Comparing a machine learning approach to categorizing with deep learning [29].*

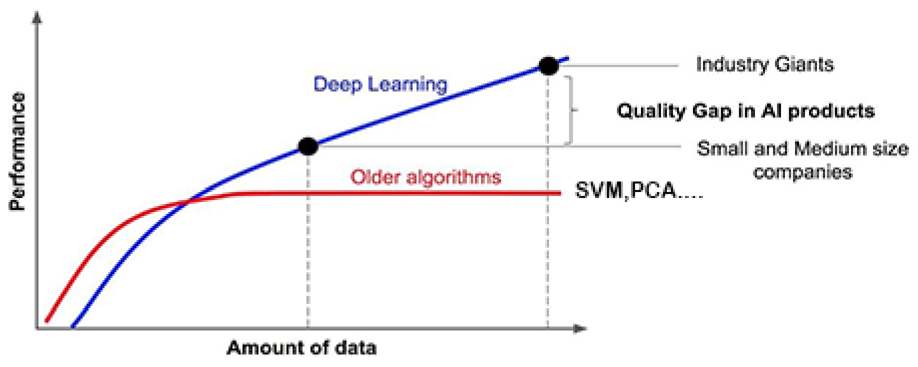


Figure II. 7*:the relation between the performance of the algorithms and the amount of input Data*

2.3.3. Deep Learning Principal of Operation

Deep learning models are sometimes referred to as deep neural networks because most deep learning approaches use neural network designs.

The number of hidden layers in a neural network is commonly referred to as "deep." Deep neural networks can have many hidden layers, whereas traditional neural networks only have 2-3.

Large sets of labeled data and neural network topologies that learn features directly from the data without the requirement for manual feature extraction are used to train deep learning models. Convolutional neural networks are one of the most popular types of deep neural networks (CNN or ConvNet). A CNN uses 2D convolutional layers to combine learnt features with input data, making it well suited to processing 2D data like images.

3. Convolutional Neural Network

3.1.Definition

A convolutional neural network (CNN or ConvNet) is a deep learning network design that learns from data without the requirement for human feature extraction. CNNs are particularly useful for recognizing objects, faces, and settings by looking for patterns in images. They're also useful for categorizing non-image data including audio, time series, and signal data. CNNs are frequently used in applications that need object recognition and computer vision, such as fire detection, self-driving vehicles, and face recognition.

3.2. CNNs Principal of Operation

A convolutional neural network can include tens or hundreds of layers, each of which learns to recognize distinct picture features. At various resolutions, filters are applied to each training image, and the result of each convolved image is utilized as the input to the next layer.

The filters can start with very basic qualities like brightness and edges and progress to more complicated attributes that describe the object uniquely.

A. Feature Learning, Layers, and Classification

A CNN, like other neural networks, has an input layer, an output layer, and a number of hidden layers in between.

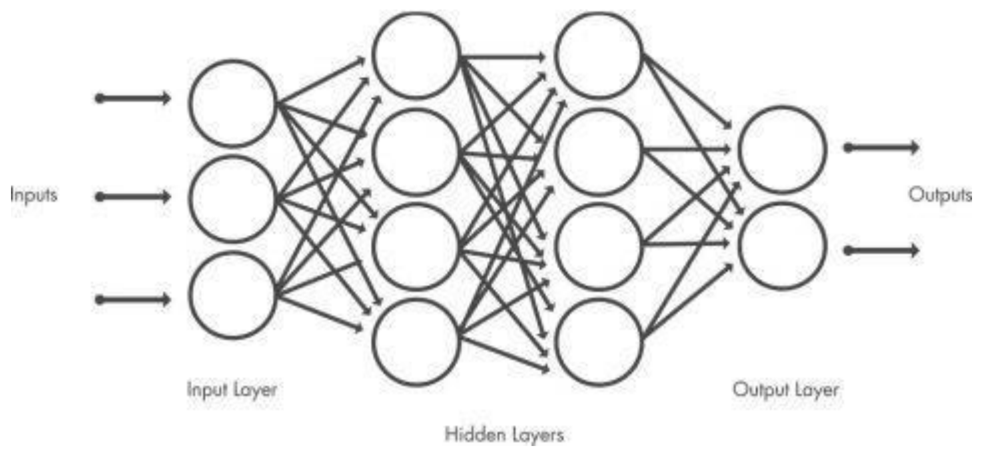


Figure II. 8*:Neural networks, which are organized in layers consisting of a set of interconnected nodes [33].*

These layers perform operations on the data with the goal of learning data-specific characteristics. Convolution, activation or ReLU, and pooling are three of the most popular layers.

- ***Convolution*** passes the input images through a series of convolutional filters, each of which activates different aspects of the images.
- ***Rectified linear unit (ReLU)*** By mapping negative values to zero and keeping positive values, it is possible to train faster and more effectively. Because only the activated characteristics are carried on into the next layer, this is frequently referred to as activation.
- ***Pooling*** reduces the amount of parameters that the network needs to learn by performing nonlinear down sampling on the output.

  Over tens or hundreds of layers, same actions are repeated, with each layer learning to recognize distinct features

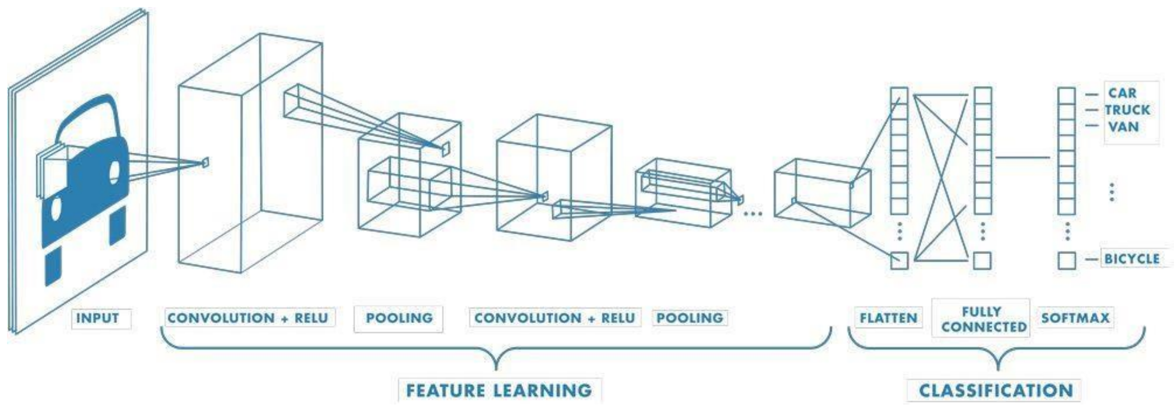


Figure II. 9:*Example of a network with many convolutional layers [32].*

B. Shared Weights and Biases

A CNN has neurons with weights and biases, just like a typical neural network. During the training phase, the model learns these values and updates them with each new training sample. In the case of CNNs, however, all hidden neurons in a layer have the identical weights and bias values.

This suggests that all hidden neurons are detecting the same feature in different parts of the image, such as an edge or a blob. As a result, the network is tolerant of image object translation. A network trained to recognize autos, for example, will be able to do so regardless of where the car appears in the image

C. Classification Layers

The architecture of a CNN turns to categorization after learning features in several layers.

The layer after that is a fully connected layer that outputs a vector with K dimensions, where K is the number of classes that the network can predict. The probabilities for each class of every image being classified are contained in this vector.

The classification output is provided by the final layer of the CNN design, which uses a classification layer such as softmax,

4. Computer vision:

4.1.Definition

Computer vision, abbreviated as CV, is a multidisciplinary field that focuses on replicating parts of the complexity of the human vision system and enabling computers to "see" and understand the content of digital images. It is a multidisciplinary field that could broadly be described as a subfield of artificial intelligence and machine learning (Figure II.10) that focuses on replicating parts of the complexity of the human vision system and enabling computers to "see" and understand the content of digital images. The purpose of computer vision is to comprehend the content of digital images by extracting a description from the image, which could be an object, a text description, or a three-dimensional model, among other things. Typically, this entails the creation of techniques that seek to replicate human eyesight figure.

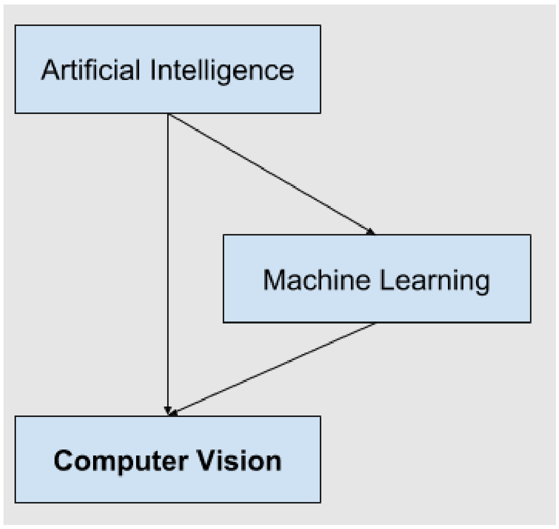


Figure II. 11:*Overview of the relationship of AI and CV [32].*

4.2.The Evolution of Computer Vision

Early computer vision investigations began in the 1950s, and by the 1970s, it was being used commercially to discern between typed and handwritten text. Today, the applications for computer vision have developed tremendously.

The tasks that computer vision could perform before the introduction of deep learning were relatively limited, and they needed a lot of manual coding and work from developers and human operators. For example, if we wanted to run an object recognition model, we would have to do the following:

- ***Create a database*:** Individual images of all the subjects we wish to track must be captured in a precise format.
- ***Annotate images*:** Then we'd have to enter numerous critical data points for each individual image.
- ***Capture new images*:** Then, whether from photographs or video content, we'd have to capture new images. After that, we had to repeat the measurement process, this time noting the image's main points. The angle from which the photo was shot has to be considered as well.

After all of this painstaking labour, the application would eventually be able to compare the measures in the new image to those in its database and inform you if it matched any of the profiles it was tracking. In actuality, the majority of the job was done manually, with very little mechanization. And the margin of error was still significant.

Artificial intelligence has made enormous strides in recent years, surpassing humans in several tasks linked to recognizing and labeling objects, thanks to breakthroughs in artificial intelligence and innovations in deep learning, neural networks, and the amount of data we generate today.

4.3.Computer Vision with Machine Learning Approach

When it came to solving computer vision difficulties, ML offered a fresh perspective. Developers no longer have to manually code every rule into their vision applications thanks to machine learning. Instead, they created "features," which were mini apps that could detect certain patterns in photographs. They then utilized a statistical learning technique to find patterns, classify photos, and detect objects in them, such as linear regression, logistic regression, decision trees, or support vector machines (SVM).

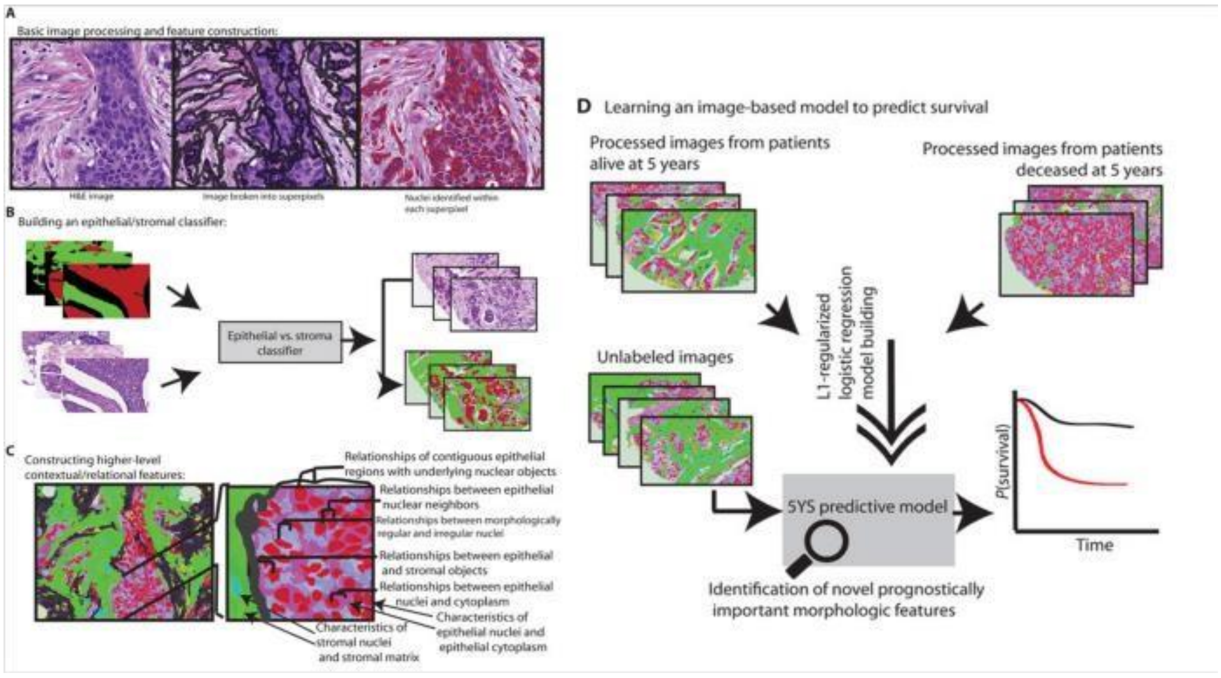

A
Basic image processing and feature construction:
H&E image
Image broken into superpixels
Nuclei identified within each superpixel
B
Building an epithelial/stromal classifier:
Epithelial vs. stroma classifier
C
Constructing higher-level contextual/relational features:
Relationships of contiguous epithelial regions with underlying nuclear objects
Relationships between epithelial nuclear neighbors
Relationships between morphologically regular and irregular nuclei
Relationships between epithelial and stromal objects
Relationships between epithelial nuclei and cytoplasm
Characteristics of epithelial nuclei and epithelial cytoplasm
Characteristics of stromal nuclei and stromal matrix
D Learning an image-based model to predict survival
Processed images from patients alive at 5 years
Processed images from patients deceased at 5 years
L1-regularized logistic regression model building
Unlabeled images
5YS predictive model
P(survival)
Time
Identification of novel prognostically important morphologic features

Figure II. 12:an example of a machine learning approach [33].

Many challenges that were previously difficult for traditional software development tools and methodologies were solved thanks to machine learning. Years ago, for example, machine learning programmers were able to develop algorithms that outperformed human specialists in predicting breast cancer survival windows. However, developing the software's functionality necessitated the collaboration of dozens of engineers and breast cancer experts and took a long period.

4.4.Computer Vision with Deep Learning Approach

Machine learning was approached in a fundamentally new way with DL. Deep learning is based on neural networks, which are a general-purpose function capable of solving any problem that can be represented by instances. When you provide a neural network a large number of labeled instances of a certain type of data, it may uncover common patterns between them and turn them into a mathematical equation that can be used to categorize future pieces of information.

DL is an extremely effective computer vision algorithm. Creating a good deep learning system usually boils down to accumulating a huge amount of labeled training data and fine-tuning parameters like the type and number of layers of n-layers.

Deep learning is easier and faster to design and deploy than prior types of machine learning. Deep learning is used in most contemporary computer vision applications, including fire detection, self-driving cars, and facial recognition.

Deep learning has progressed from a theoretical notion to a practical application because to breakthroughs in hardware and cloud computing resources.

5. Object Recognition

5.1.Definition

Object recognition is a broad word that refers to a group of related computer vision tasks that entail recognizing things in digital pictures. Object localization entails creating a bounding box around one or more objects in an image, whereas image classification entails providing a class label to an image. Object detection is a more difficult task that combines the two tasks by drawing a bounding box around each object of interest in the image and assigning a class label to it. Object recognition is the umbrella term for all of these issues.

5.2.Object Recognition Computer Vision Tasks

– ***Image Classification*:** Predict an object's type or class from an image.
  Input: A single-object image, such as a photograph.
  output: A class label (e.g. one or more integers that are mapped to class labels).
– ***Object Localization:*** Detect the existence of items in a picture and use a bounding box to denote their location.
  Input: *an image featuring one or more objects.*, such as a photograph.
  Output: One or more bounding boxes (e.g. defined by a point, width, and height).

- ***Object Detection:*** Locate the existence of things in a picture using a bounding box and the types or classes of the objects found.
- *Input*: *an image featuring one or more objects.*, such as a photograph.
- *Output*: One or more bounding boxes (e.g. defined by a point, width, and height), and a class label for each bounding box.
- ***Object segmentation,*** also known as "object instance segmentation" or "semantic segmentation," is a further development of this split of computer vision tasks, in which instances of recognized objects are shown by highlighting the exact pixels of the object rather than a crude bounding box.

  Object recognition, as we can see from this split, pertains to a group of difficult computer vision tasks that can be summarized in Figure II. 13 below.

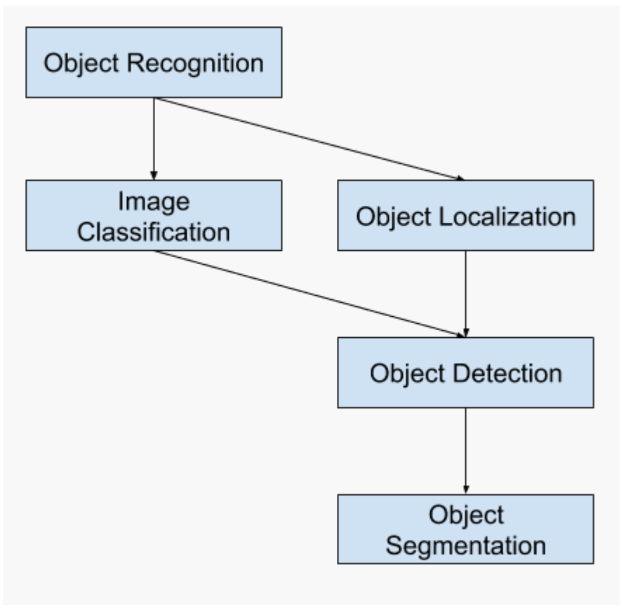


Figure II. 14:*Overview of Object Recognition Computer Vision Tasks [27].*

With the goal of encouraging autonomous and distinct innovations at each level that can be exploited more widely. The three equivalent task kinds, for example, are listed below:

- ***Image classification***: Algorithms generate a list of object categories that can be found in the image.
- ***Single-object localization:*** Algorithms generate a list of object categories that appear in the image, as well as an axis-aligned bounding box that indicates the position and scale of one instance of each object category.
- ***Object detection:*** Algorithms provide a list of item categories that appear in the image, as well as an axis-aligned bounding box that shows the position and scale of each instance of each object type.

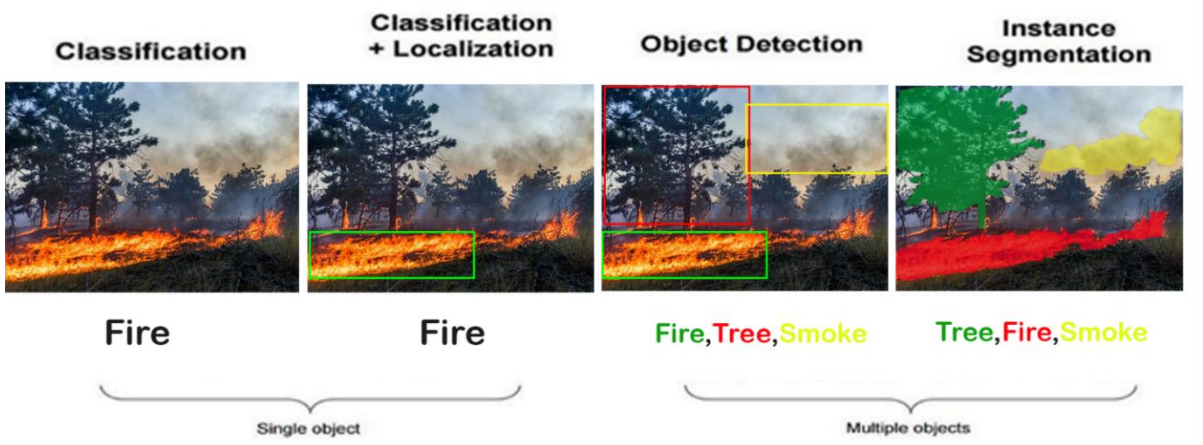


Figure II. 15:*Overview of Object Recognition Computer Vision Tasks for fire scene example.*

5.3.Comparison between Single Object Localization and Object Detection:

We can see that "Single-object localization" is a more constrained variant of the more generally described "Object Localization," limiting the localization tasks to objects of a single kind within an image, which we may infer is a simpler process.

The distance between the expected and predicted bounding boxes for the expected class is used to evaluate a model's performance for single-object localization. The precision and recall of each of the best matched bounding boxes for the known objects in the image are used to evaluate the performance of an object recognition model.

Figure II. 16 below summaries the difference between Single object localization and object detection.

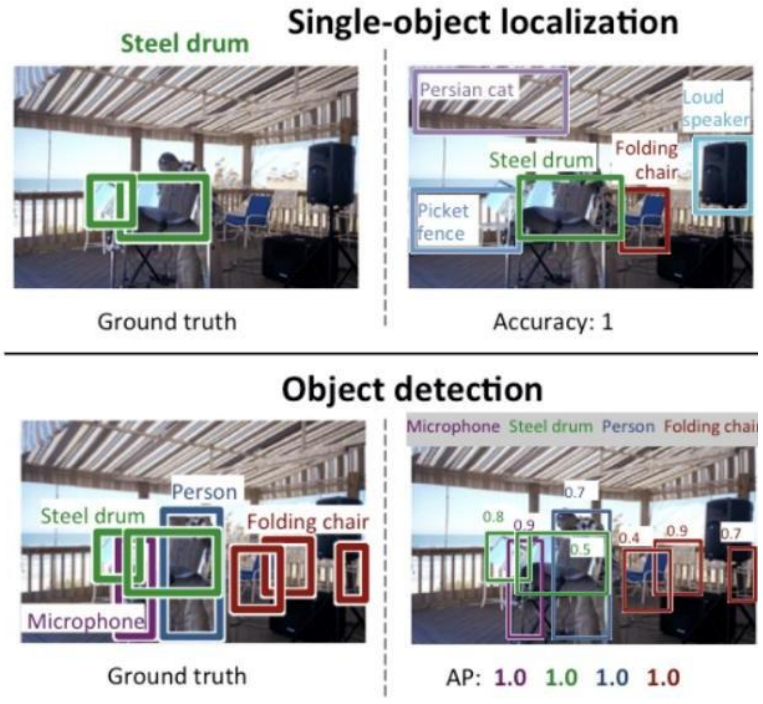


Figure II. 17:*Single Object Localisation VS Object Detection [27].*

5.4.Popular Families of Object Recognition Deep Learning Models

5.4.1 R-CNN Model Family

The R-CNN family of algorithms developed by Ross Girshick et alR-CNN,'s which stands for "Regions with CNN Features" or "Region-Based Convolutional Neural Network." This contains the R-CNN, Fast R-CNN, and Faster-RCNN object localization and recognition algorithms that have been proposed and demonstrated.

Let's take a deeper look at each of these techniques' highlights one by one.

a) R-CNN

The R-CNN was first described in the 2014 publication “Rich feature hierarchies for accurate object detection and semantic segmentation” by Ross Girshick et al. from UC Berkeley.

It's possible that it was one of the first large-scale and successful applications of convolutional neural networks to the problem of object detection, segmentation, and localization. On the VOC-2012 dataset and the 200-class ILSVRC-2013 object detection dataset, the technique was shown on benchmark datasets, yielding state-of-the-art results.

The R-CNN model they propose is made up of three modules:

– *Module 1: Region Proposal*. Generate and extract category independent region proposals, e.g. candidate bounding boxes.

– *Module 2: Feature Extractor*. Extract feature from each candidate region, e.g. using a deep convolutional neural network.

– *Module 3: Classifier*. Classify features as one of the known classes, e.g. linear SVM classifier model.

The architecture of the model is summarized in Figure II. 18 below.

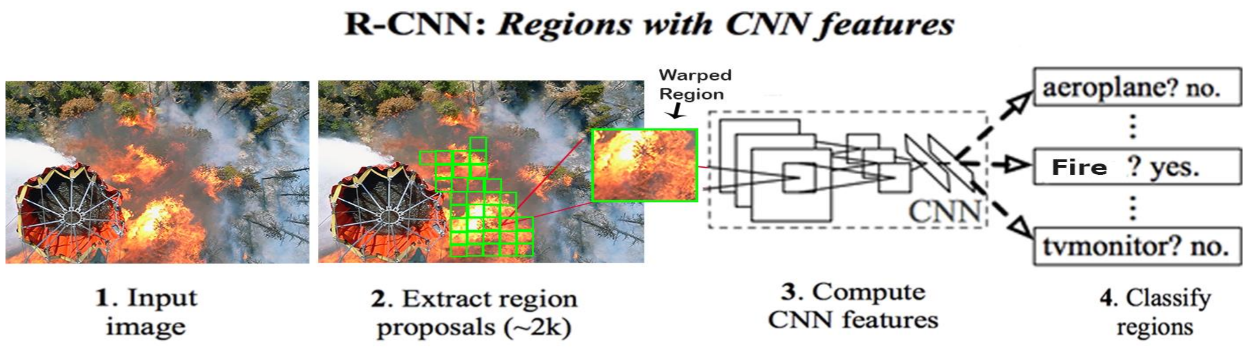


Figure II.12:Summary of the R-CNN architecture

To propose candidate regions or bounding boxes of prospective objects in the image, a computer vision approach known as "selective search" is utilized, while the design flexibility allows for different region proposal techniques to be applied.

The AlexNet deep CNN feature extractor was utilized by the model. The CNN produced a 4,096-element vector that characterizes the image's contents and is input into a linear SVM for classification, with one SVM trained for each known class.At test time, there were 2,000 potential regions per image.

b) *Fast R-CNN*

Given the great success of R-CNN, Ross Girshick, then at Microsoft Research, proposed an extension to address the speed issues of R-CNN in a 2015 paper titled "Fast R-CNN."

We can sit its limitation as follows:

– *Training is a multi-stage pipeline.* Involves the preparation and operation of three separate models.
– *Training is expensive in space and time*. Training a deep CNN on so many region proposals per image is very slow.
– *Object detection is slow*. Make predictions using a deep CNN on so many region proposals is very slow.

In the 2014 study "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," it was proposed to speed up the approach known as spatial pyramid pooling networks, or SPPnets. This did speed up the feature extraction process, but it was effectively a forward pass caching approach.

Fast R-CNN is proposed as a single model that learns and outputs regions and classifications without the use of a pipeline.This technique is then done for each region of interest in a given image many times. The architecture of the model is summarized in Figure II. 13 below.

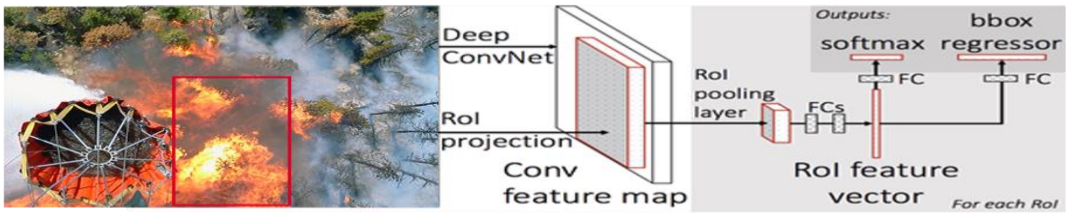


Figure II. 13:Summary of the Fast R-CNN Model Architecture.

Although the model is substantially faster to train and forecast, each input image still requires a collection of candidate regions to be offered.

c) Faster R-CNN

The model architecture was further improved for both speed of training and detection by Shaoqing Ren, et al. at Microsoft Research in the 2016 paper titled "Faster R-CNN: Towards

Real-Time Object Detection with Region Proposal Networks.” [27].

The architecture, known as a Region Proposal Network, or RPN, was created to both propose and refine region proposals as part of the training process. In a single model architecture, these areas are combined with a Fast R-CNN model. These enhancements minimize the number of region proposals while also accelerating the model's test-time operation to near real-time, resulting in state-of-the-art performance. Although it is a single unified model, the architecture is comprised of two modules:

– *Module 1: Region Proposal Network*. Convolutional neural network for proposing regions and the type of object to consider in the region.
– *Module 2: Fast R-CNN*. Convolutional neural network for extracting features from the proposed regions and outputting the bounding box and class labels.

Both modules work with the same deep CNN output. The Fast R-CNN network uses the region proposal network as an attention mechanism, directing the second network where to look or pay attention.

The architecture of the model is summarized in Figure II. 14 below.

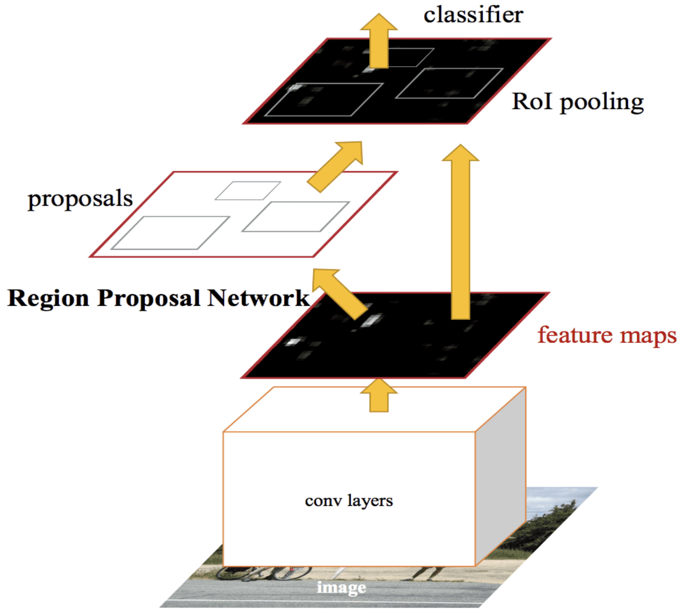


Figure II.14:*Summary of the Faster R-CNN Model Architecture [27].*

5.4.2 Time Object Detection with Region Proposal Networks

The RPN works by traversing a tiny network over the feature map with the output of a pre-trained deep CNN, such as VGG-16, and outputting numerous region proposals and a class prediction for each. Region proposals are bounding boxes based on so-called anchor boxes or pre-defined forms that are intended to speed up and improve region proposal. The class prediction is binary, indicating the presence of an object, or not, so-called “objectness” of the proposed region.

5.4.3 YOLO Model Family

Another popular family of object recognition models is referred to collectively as YOLO or“*You Only Look Once*,” develop by Joseph Redmon, et al.

Although R-CNN models are more accurate in general, the YOLO family of models is far faster, achieving real-time object detection.

a) YOLO:

The YOLO model was first described by Joseph Redmon, et al. in the 2015 paper titled “You Only Look Once: Unified, Real-Time Object Detection.” Note that Ross Girshick, developer of R-CNN, was also an author and contributor to this work, then at Facebook AI Research.

The approach uses a single end-to-end trained neural network that takes a photograph as input and immediately predicts bounding boxes and class labels for each bounding box. Although the technique operates at 45 frames per second and up to 155 frames per second for a speed-optimized version of the model, it has lower prediction accuracy (e.g., more localization errors).

The model divides the input image into a grid of cells, with each cell responsible for predicting a bounding box if the center of a bounding box falls within the cell. Each grid cell generates a bounding box based on the x, y coordinates, as well as the width, height, and confidence. Each cell also serves as a basis for a class prediction..

For instance, an image may be divided into a 77-cell grid, with each cell predicting two bounding boxes, yielding 94 recommended bounding box predictions. The bounding boxes with confidences and the class probability map are then combined to create a final set of bounding boxes and class labels. The model's two outputs are summarized in Figure II. 15.

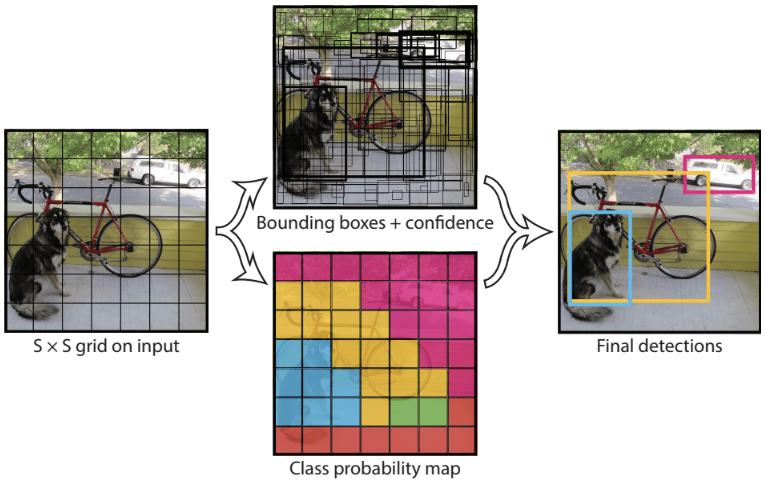


Figure II.15: Summary of Predictions made by YOLO Model [27].

b) *YOLOv2 (YOLO9000) and YOLOv3:*

The model was updated by Joseph Redmon and Ali Farhadi in an effort to further improve model performance in their 2016 paper titled "YOLO9000: Better, Faster, And Stronger." Although this variation of the model is referred to as YOLO v2, an instance of the model is described that was trained on two object recognition datasets in parallel, capable of predicting 9,000 object classes, hence given the name "*YOLO9000*." [27]

The model has undergone a variety of training and architectural improvements, including the use of batch normalization and high-resolution input images.

The YOLOv2 model, like the Faster R-CNN, employs anchor boxes, which are pre-defined bounding boxes with relevant shapes and sizes that are customized during training. On the training dataset, a k-means analysis is used to select the bounding boxes for the image.

Importantly, the bounding box predictions are adjusted to allow tiny changes to have a less dramatic influence on the predictions, resulting in a more stable model. Rather of explicitly forecasting location and size, offsets for moving and reshaping the pre-defined anchor boxes relative to a grid cell are anticipated and dampened by a logistic function.

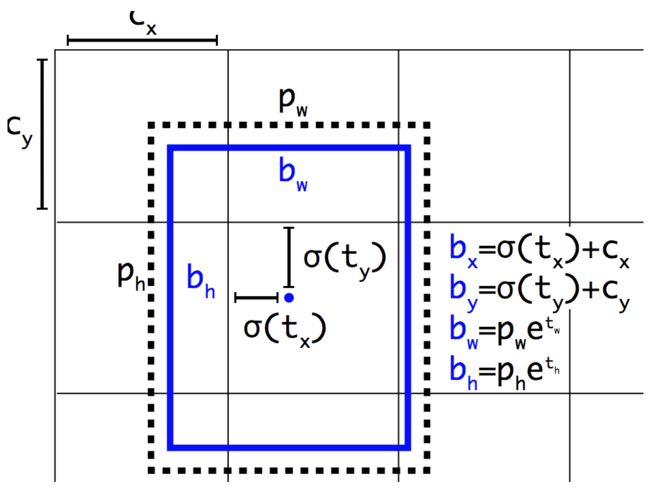


Figure II.16:*Example of the representation chosen when predicting Bounding box position and shape [27].*

Further improvements to the model were proposed by Joseph Redmon and Ali Farhadi in their 2018 paper titled "YOLOv3: An Incremental Improvement." The improvements were reasonably minor, including a deeper feature detector network and minor representational changes [22].

6. Conclusion:

We have devoted this chapter to the presentation of some generalities on Deep Learning approaches with computer vision while presenting the Convolutional Neural Network and its popular models family, as we discussed above the use of a deep Convolutional Neural Network in object recognition has been a popular solution in the recent years due to its impressive results and its evolution whether in its various architectures or its models family. and in order to detect forest fire with great precision in real time we choose to work with the YOLOv2 Model since it is faster and more accurate when it comes to real time detection.The next chapter is dedicated to expose and to dive deep in the YOLOv2 Model and the process of constructing the fire detection model in MATLAB from the data collection to labelling to training.

Chapter III: Forest Fire Detection Model Using YOLOv2

1. Introduction:

Recent advances in artificial intelligence (AI) and Deep learning have made image-based modeling and analysis (e.g., classification, real time prediction, and image segmentation) even more successful in different applications Also, with the advent of nanotechnology semiconductors, a new generation of Tensor Processing Units (TPUs) and Graphical Processing Units (GPUs) can provide an extraordinary computation capability for data-driven methods. Moreover, modern drones and UAVs can be equipped with tiny edge TPU/GPU platforms to perform on-board processing on the fly to facilitate early fire detection before a catastrophic event happens.

And to build such deep learning computer vision system for fire detection we need to have two essential elements: The Dataset and the Trained Model.

In this chapter, we will discuss the techniques and the approaches we have used to design and train the fire Detection model, it will be divided into two main processes:

First, we will talk about the fire detection Dataset Preparation process**,** then in the second part we will talk about the training process.

2. Fire Detection Dataset

Since our project is based on classifying and localizing a fire from an Arial input frame provided by a UAV's camera, the learning approach that we are going to adopt is a supervised deep learning approach but the challenge in utilizing such approaches is that they rely on large training datasets, without ignoring the fact that successful deep learning models are built on the shoulders of a large volumes of high-quality training data, considering these challenging facts we had to give more attention to the dataset that we will use and thus so much work has to be devoted in the creation and the collection of the necessary high quality data in order to train a reasonably accurate model, but before diving into the work process , we will take an overview of the popular available Fire Detection Datasets published Papers that are recently and frequently used.

2.1 Popular Fire Detection Datasets:

2.1.1 THE FLAME DATASET: Aerial Imagery Pile Burn Detection Using Drones (UAVS) [23].

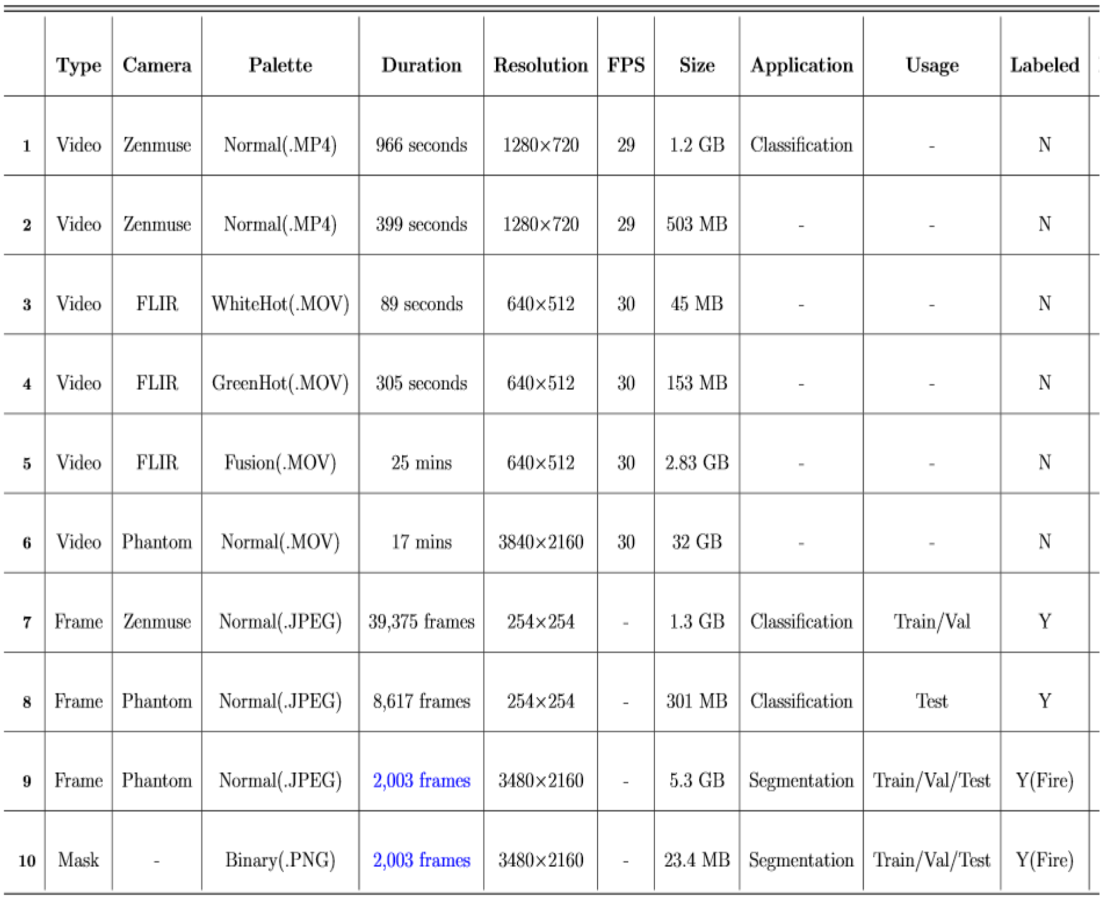
This study provides an aerial imagery *FLAME (Fire Luminosity Airborne-based Machine learning Evaluation)* dataset using drones during a prescribed pile burn in Northern Arizona, USA.

The test was conducted with fire managers from the Flag staff (Arizona) Fire Department who carried out a burn of piled slash on city-owned lands in a ponderosa pine forest on Observatory Mesa.

The prescribed fire took place on January 16th, 2020 with the temperature of 43◦F (∼ 6◦C) and partly cloudy conditions and no wind.

This dataset consists of different repositories including raw aerial videos recorded by drones' cameras and also raw heatmap footage recorded by an infrared thermal camera. To help researchers, two well-known studies; fire classification and fire segmentation are defined based on the dataset. For approaches such as Neural Networks (NNs) and fire classification, 39,375 frames are labeled ("Fire" vs "Non-Fire") for the training phase. Also, another 8,617 frames are labeled for the test data. 2,003 frames are considered for the fire segmentation and regarding that, 2,003 masks are generated for the purpose of Ground Truth data with pixel-wise annotation.

The FLAME dataset including all images, videos, and data are available on IEEE-Dataport [35].



	Type	Camera	Palette	Duration	Resolution	FPS	Size	Application	Usage	Labeled
1	Video	Zenmuse	Normal(.MP4)	966 seconds	1280×720	29	1.2 GB	Classification	-	N
2	Video	Zenmuse	Normal(.MP4)	399 seconds	1280×720	29	503 MB	-	-	N
3	Video	FLIR	WhiteHot(.MOV)	89 seconds	640×512	30	45 MB	-	-	N
4	Video	FLIR	GreenHot(.MOV)	305 seconds	640×512	30	153 MB	-	-	N
5	Video	FLIR	Fusion(.MOV)	25 mins	640×512	30	2.83 GB	-	-	N
6	Video	Phantom	Normal(.MOV)	17 mins	3840×2160	30	32 GB	-	-	N
7	Frame	Zenmuse	Normal(.JPEG)	39,375 frames	254×254	-	1.3 GB	Classification	Train/Val	Y
8	Frame	Phantom	Normal(.JPEG)	8,617 frames	254×254	-	301 MB	Classification	Test	Y
9	Frame	Phantom	Normal(.JPEG)	2,003 frames	3480×2160	-	5.3 GB	Segmentation	Train/Val/Test	Y(Fire)
10	Mask	-	Binary(.PNG)	2,003 frames	3480×2160	-	23.4 MB	Segmentation	Train/Val/Test	Y(Fire)

Table III. 1*:Flame Dataset Information Table [23].*

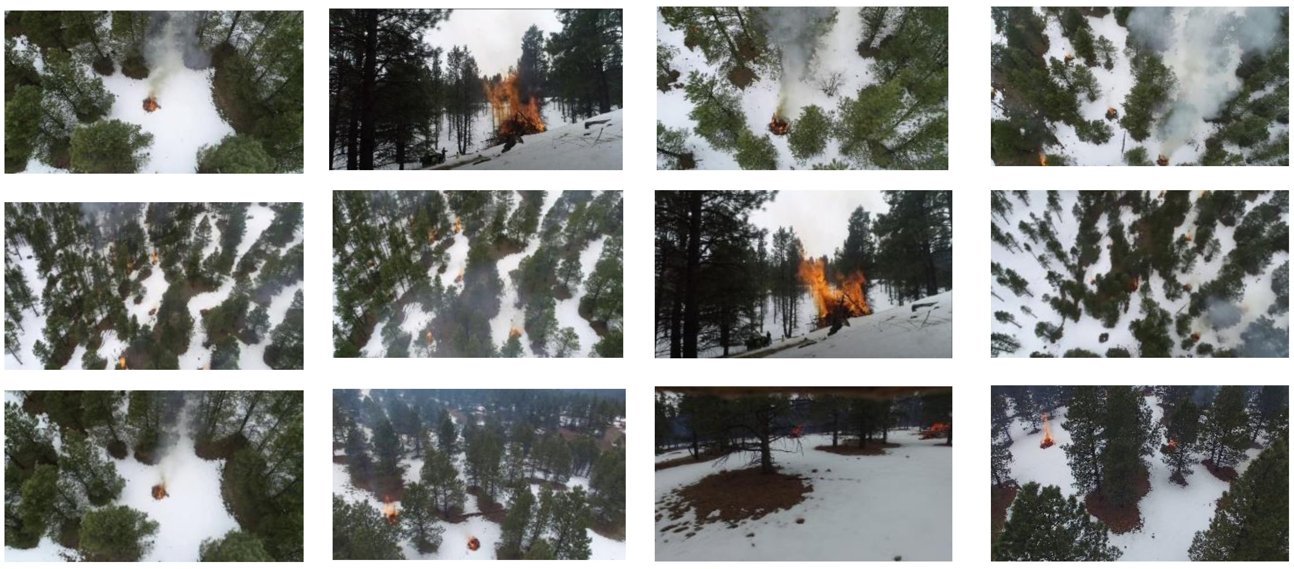


Figure III. 1*:Frame samples of the FLAME Dataset [35].*

2.1.2 FiSmo Dataset: A Compilation of Datasets from Emergency Situations for Fire and Smoke Analysis [24].

FiSmo, is a compilation of Datasets from emergency situations, composed of images, videos, regions of interest (ROIs), annotations, and features. these Datasets were employed in the context of the RESCUER project (RELIABLE AND SMART CROWDSOURCING SOLUTION FOR EMERGENCY AND CRISIS MANAGEMENT); The Databases and Images Group (GBDI) from the institute of mathematics and computer science of the university of SAO PAULO (ICMC-USP), is responsible for the image and video analysis functionalities of the rescuer architecture.

Considering the high cost of making simulations in emergency scenarios to take pictures and make videos, they assume that images and videos gathered from social media website were suitable to reflect the real case scenarioof emergency situations they divulge a compilation of datasets of images and videos that present emergency scenarios called FiSmo.

the Dataset is composed of images retrieved from the Flickr social media under the creative commons license; videos obtained from YouTube; and simulations carried out during the RESCUER project.

These works were focused on the analysis of images and videos regarding the presence of fire, smoke, and explosions in emergency situations. the available data is composed of four image and two video datasets: fire/smoke detection in images; fire segmentation in images; smoke segmentation in images; content-based image retrieval; temporal segmentation of fire segments in videos; and fire detection in videos. all datasets were preprocessed according to the involved context, including annotation steps carried out by a set of subjects, training images and ROIs. furthermore, the extracted feature vectors are also available providing features of color and texture.

These data were labeled according to the presence or absence of fire/smoke. therefore, the datasets provide a proper material for the validation of the algorithms developed for emergency image and video analysis.

FiSmo-Images				
Dataset Name	Purpose	# images	Features	ROIs
FireSmoke	Fire and Smoke detection	5,556	No	No
Flickr-Fire	Global fire detection and content-based image retrieval	2,000	Yes	No
BoWFire	Fire detection and segmentation	226	No	Yes
SmokeBlock	Smoke detection and segmentation	1,666	Yes	Yes

***Table III. 2**:Summarization of the FiSmo-Images Datasets [24].*

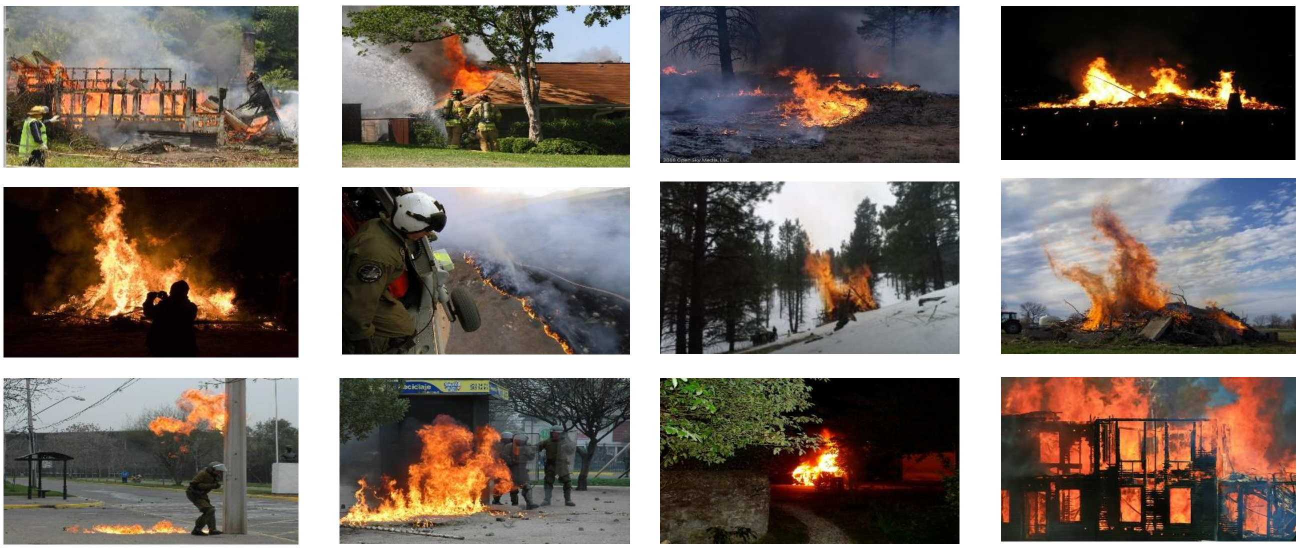


Figure III. 2:*Frame samples of the FiSmo Datasets [36].*

2.1.3 Datasets Discussion:

Doing our researches, we've noticed that although the amount of research papers and the work done recently in the field of fire detection yet currently there is a scarcity of a diverse fire dataset and a huge lack of benchmark Dataset for wildfire detection images thus it's very difficult to come up with one that simulate and counterfeit the nature and the wildfire scenarios in Algeria. One of the limitations of the FiSmo datasets [24], is that although it is vast it is not diverse enough to be solely utilized for training a network and expecting it to perform well in realistic fire detection scenarios.

Moreover, that dataset was based on terrestrial images of the fire and to the best of our knowledge, there exists no aerial imaging dataset for fire analysis besides the FLAME dataset [23], Note that aerial imagery exhibits different properties such as low resolutions, and top-view perspective, substantially different than images taken by ground cameras, but unfortunately this last does not appear to be diverse enough as it contains a large number of similar images basically from a restricted scenario that does not counterfeit completely the probable Algerian wildfire scenarios (pile burn in Northern Arizona with the temperature of 43◦F (∼ 6◦C) and partly cloudy conditions and no wind).

Facing these challenges regarding the dataset we tried to create a diverse one mainly using [23] and [24] plus the different online available resources, and our modest means to create a personalized dataset that we will rely on in the training phase.

2.2 Building our Personalized Dataset:

In order to build our personalized dataset, we needed to go through some essential steps that we will explore in the section below.

2.2.1 Data Collection:

We collect a fire detection dataset from various online resources. Mainly acquired from the FLAME Dataset [35] because our model needs to be trained largely on aerial images but we tried to augment it more to overcome the challenges we discussed in the section above and maintain the diversity in our dataset, so we mixed it with randomly chosen images from FiSmo Datasets [35], we‘ve also used some common open source Datasets which we found available on GitHub [36] and Kaggle [37] they were already augmented by mirroring and adding noise so there was no need for more augmentation, we further extracted images from the forest Fire videos of Foggia’s Dataset [38] with a sample rate of 5, i.e. sample one image every five frames. we also tried to accumulate as many realistic images as possible by searching images of the recent forest fires of Tizi Ouzou and Khenchla (on Google and social media); The final dataset that we named All New Dataset [41] consists of a total of 6,592 fire images, these images are all resized in canonical size of 416 × 416 in order to support evaluation of the YOLOv2 WorkFrame, more information about the Dataset sources are reported in Table III. 3, while some visual examples are shown in Figure III. 3.

All New Dataset	
Dataset Name	# images
FLAME Dataset [3]	4,911
FiSmo Dataset (Flickr-Fire) [4]	984
Gaisaid D-Fire (GitHub) [5]	402
Fire Dataset [6]	55
Foggia’s Dataset [7]	117
Handpicked Fire Images [8]	123
TOTAL	6,592

Table III. 3:*All-New Dataset sources information.*

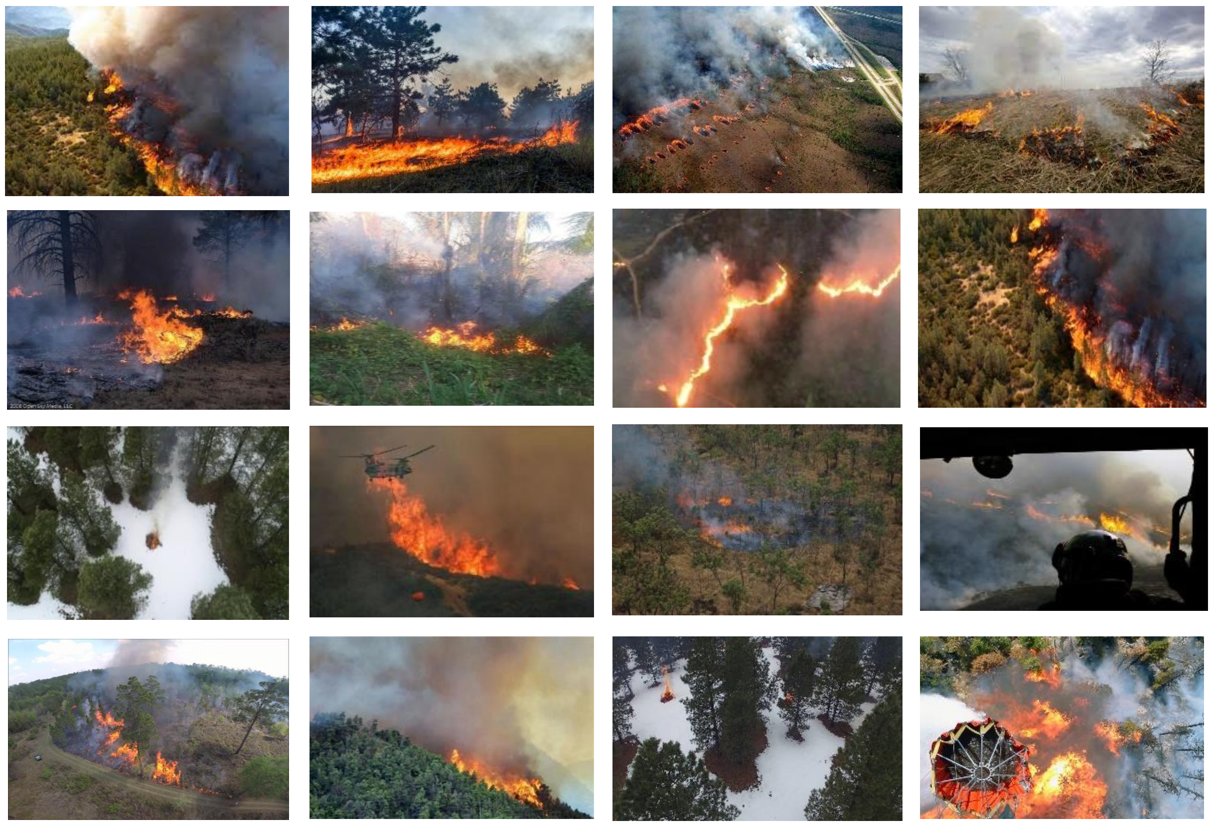


Figure III. 3:*Frame samples of the All-New Dataset.*

After collecting the necessary amount of Raw Data, we need to move to the next essential step which its success represent 80% of the model's success.

2.2.2 Data Labeling:

Today, most practical Deep learning models utilize supervised learning, which applies an algorithm to map one input to one output. For supervised learning to work, we need a labeled set of data that the model can learn from to make correct decisions. Data labeling typically starts by asking humans to make judgments about a given piece of unlabeled data.it is considered as a central part of the data preprocessing workflow for Deep learning it structures data to make it meaningful. This labeled data is then used to train a Deep learning model to find "meaning" in new, relevantly similar data. Throughout this process, Deep learning practitioners strive for both quality and quantity. More accurately labeled coupled with a larger quantity of labeled data creates more useful deep learning models, that uses human-provided labels to learn the underlying patterns in a process called "model training." The result is a trained model that can be used to make predictions on new data. But, the process to create the training data necessary to build these modes is often expensive, complicated, and time-consuming.

to build our computer vision model that can be used to automatically categorize images whether it contains Fire or not and detect the ROI which is the location of the Fire in the Frame, we first need to label images and to do that we utilized two different Methods of labeling:

- The Manual labeling Method.
- The Automatic labeling Method.

Before diving into the Fire labeling process, we need to clarify 3 main points:

The Data labeling tool: Since our input Data is all consisted of images, we opted to use the image labeler APP from the Computer Vision Toolbox available in MATLAB to do the Labeling Task.

The Label Class: For this project we choose to detect a single region of interest (ROI) which means having a single label class that we will annotate “Fire”.

The computer vision Data Annotation Type: since Fire doesn’t have a predefine shape and our main aim is to detect and localize the fire in a 2D frame for real time object detection, we will use the bounding boxes type of annotation instead of the polygons and cuboids annotation to minimize the operation time and maximize the accuracy.

2.2.2.1 Manual Labeling Method

We used the Manual labeling Method to annotate FiSmo Datase (Flicker-Fire) [10], Gaisaid D-Fire[11],Fire Dataset from Kaggle [12],Foggia’s Dataset [13] and handpicked fire images from the internet [14] , that’s because these datasets are very different from each other with different fire scenarios so we cannot apply a single automation algorithm to annotate the fire in each image Frame, automating the labeling process would be definitely a gain of time but certainly with a very low annotation precision which results a low accuracy trained model ,To overcome this challenge we need a human intervention, we had to annotate manually a total of 1,681 image frame , it was a very time-consuming process where a labeling mistake or inaccuracies of the input can lead to wrong predictions and a wrong output.

For this task we need to:

- open the image labeler App from the Image Processing and Computer Vision tab and load the data directly from the App, as shown in Figure III. 4.

define the labels we intend to draw directly within the app by creating a ROI label that corresponds to our region of interest and create a 2D rectangle label named “Fire”, as shown in Figure III. 4.

- use the mouse to draw a rectangular ROI around the Fire spot and we re-apply the process for the rest of the datasets images, as shown in Figure III. 5.

– After labeling all the frames we export the labeled ground truth to the MATLAB workspace. the labeled ground truth is stored as a ground Truth object so we can use this object to train our deep-learning-based computer vision algorithm.

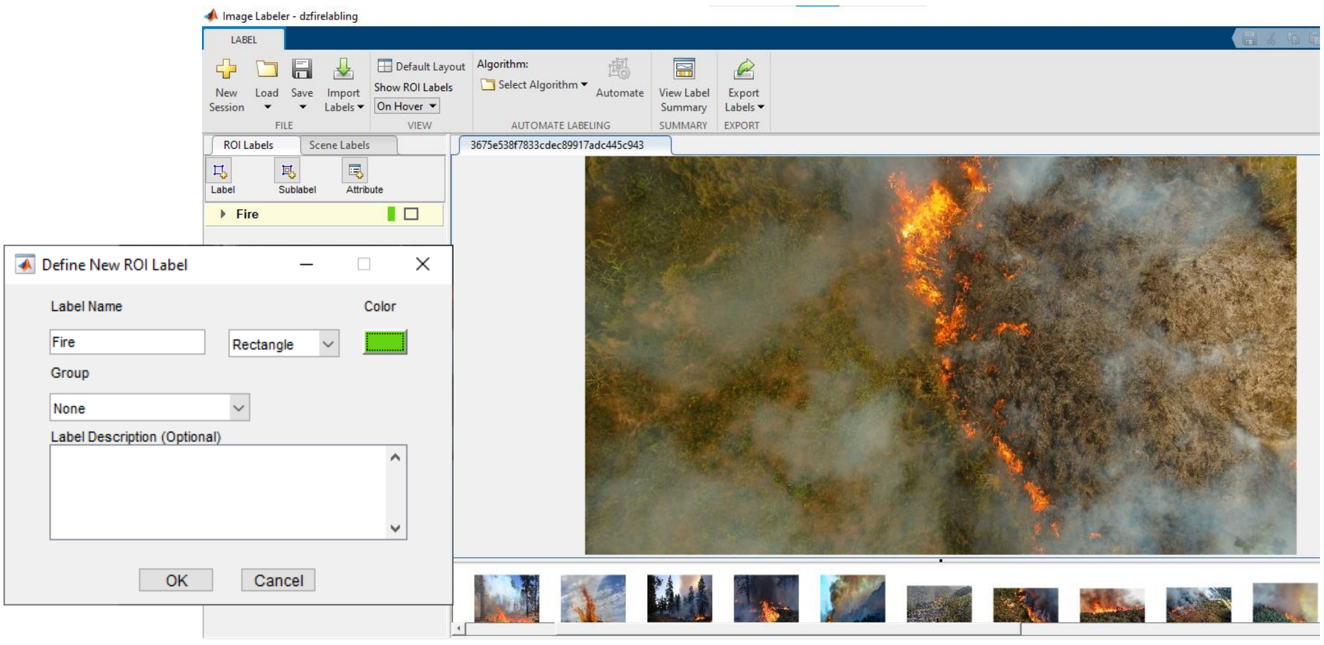


Figure III.4:*Images loading and Fire label definition in Image Labeler App.*

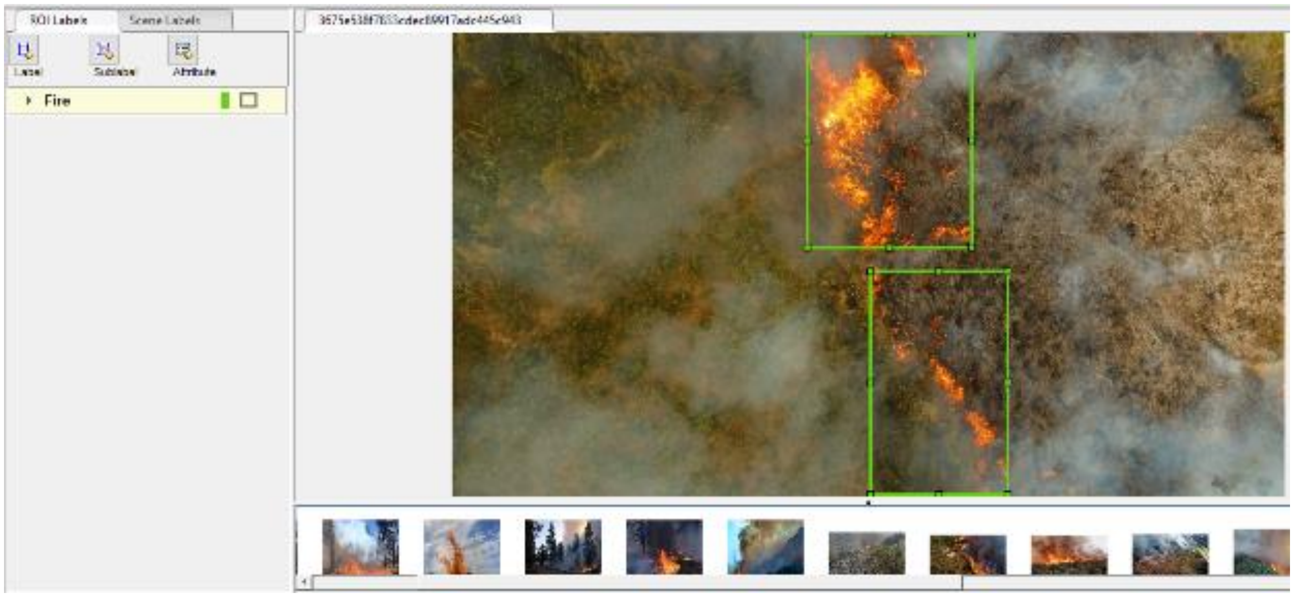


Figure III. 5:*Image annotation with green bounding boxes.*

- **Manual Labeling Summary Discussion:**

To further evaluate our labels, we can view a visual summary of the labeled ground truth, the label summary graphs for the ROI distribution of the manual labeled dataset [36],[37],[38],[39] and [40] Datasets are represented in Figure III. 6.

We used the graph to examine the occurrence of labels over time and compare the frames, the frequency of labels and the distribution of ROIs where:

– The x-axis of the graph displays the numeric ID of each image in the dataset image collection, the total number of images loaded and manually labeled from the dataset is 1,681 image frame.

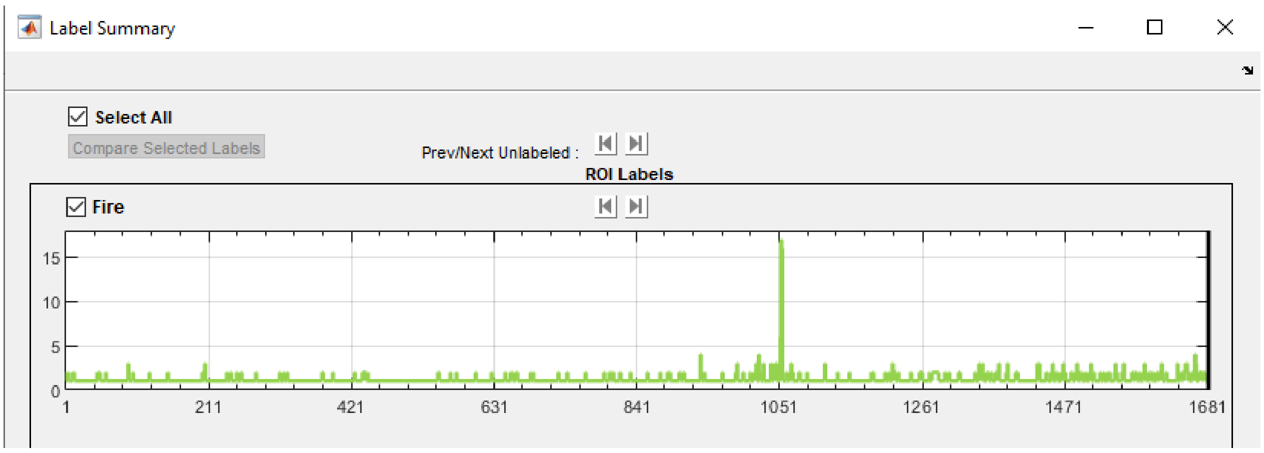


Figure III. 6:*Lebel Summary Graph of Manual Labeled Datasets.*

– The y-axis displays the number of ROIs for each image in the dataset image collection, which means how many fire bounding boxes are in each single image frame.as we can see in the Figure III. 5 above, most of the frames are labeled with the average of 1 to 4 bonding boxes per frame with an exception in some rare situations where there were frames that were labeled with up to 19 Bounding boxes.

2.2.2.2 Automatic Labeling Method:

Since the FLAME dataset [9] consists of Fire images that represent the same scenario plus it is the biggest dataset that we have, its manually labeling process will be more time-consuming, so we tried to speed it up by using an automation algorithm to label the remainder of images, for fire labeling there is no suitable built-in automation algorithm that are provided by MATLAB toolstrip for that we need to create our own custom label automation algorithm.

To create this custom automation algorithm, we first require to create a segmented Fire Mask.

a) Binary Mask creation:

The mask is the main variable that we stand in need of for custom automation algorithm, using the Color Thresholder APP from the Computer Vision Toolbox available in MATLAB, we created the necessary binary segmentation Fire mask that we used to create the automation algorithm for the FLAME dataset labeling.

In order to crate this mask, we loaded a good quality and net fire image from the Flame dataset to the color Thresholder App we choose to segment in the RGB color space because it isolates the fire colors better than the rest of the color spaces.

To separate fire from the rest elements of the image we used the point cloud approach where the App converts the 3-D point cloud into a 2-D representation and activates the polygon ROI tool, then we Draw a ROI around the Fire we want to segment and fine tune the segmentation using the color controls R, G, B.

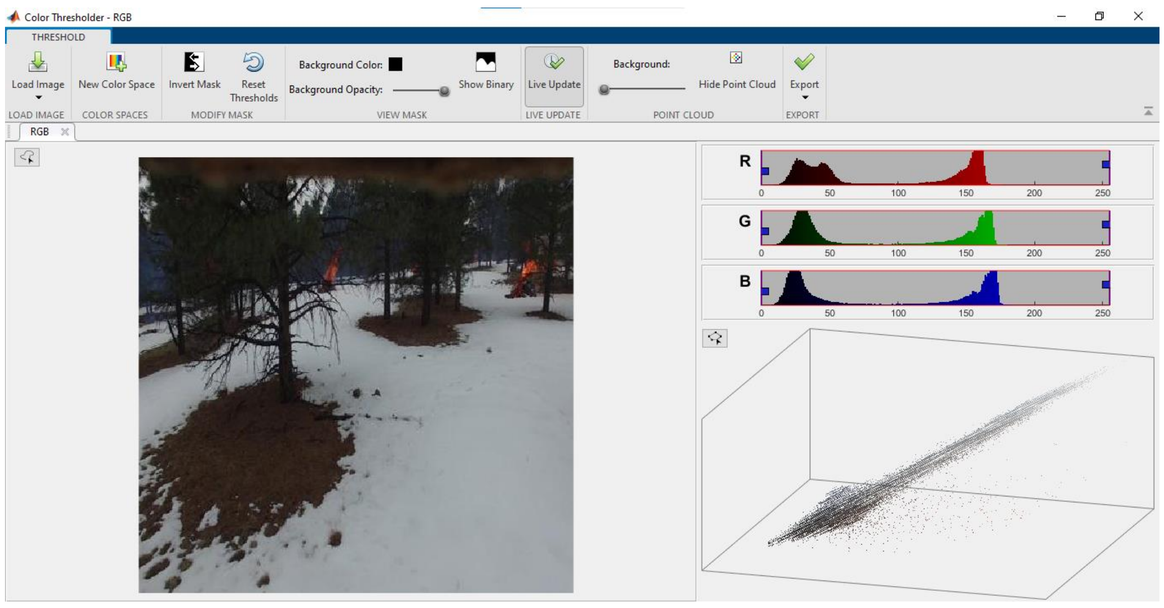
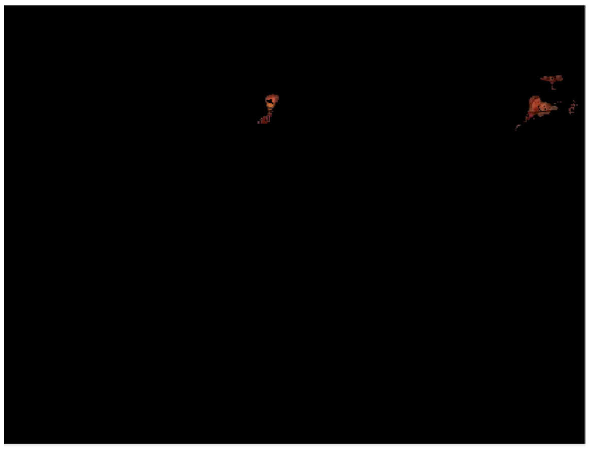


Figure III. 7:Color Thresholder App interface.

After checking the binary Fire mask, we export the function which we named “create mask” with the code that creates the segmentation so we can use it to create the custom algorithm.



. ***Figure III.8****:Segmentation Fire Mask*

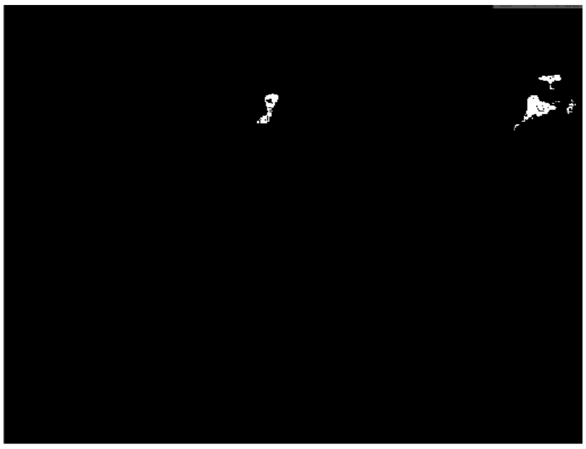


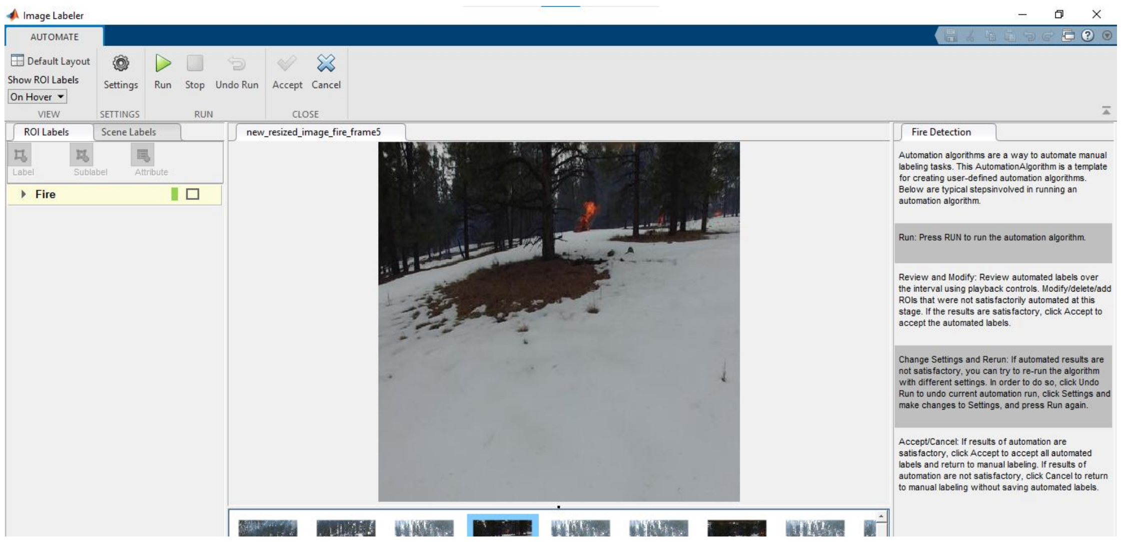
Figure III. 4:*Binary Fire Mask.*

b) ***Custom Algorithm Creation and Automation:***

For the Creation of the custom algorithm, we used two functions: the first is the binary mask function that we discussed above and the second is the “object label” function that the algorithm needs in order to move from a black and white binary mask input into a 2-D bounding box output, we completed coding the custom algorithm and we name It” Fire Detection”.

To perform the algorithm on the whole FLAME dataset [35] we need to:

– Open the image labeler App from the Image Processing and Computer Vision tab and load all the images directly from the App as shown is Figure III. 10 below.
– Define the labels we intend to draw directly within the app by creating a ROI label that corresponds to our region of interest and create a 2D rectangle label named “Fire”.
– Import the custom automation algorithm” Fire Detection” from the MAT files, select all the images and run the automation on.



Automation algorithms are a way to automate manual labeling tasks. This AutomationAlgorithm is a template for creating user-defined automation algorithms. Below are typical stepsinvolved in running an automation algorithm.

Run: Press RUN to run the automation algorithm.

Review and Modify: Review automated labels over the interval using playback controls. Modify/delete/add ROIs that were not satisfactorily automated at this stage. If the results are satisfactory, click Accept to accept the automated labels.

Change Settings and Rerun: If automated results are not satisfactory, you can try to re-run the algorithm with different settings. In order to do so, click Undo Run to undo current automation run, click Settings and make changes to Settings, and press Run again.

Accept/Cancel: If results of automation are satisfactory, click Accept to accept all automated labels and return to manual labeling. If results of automation are not satisfactory, click Cancel to return to manual labeling without saving automated labels.

Figure III.10*:Loading dataset to run the automatic custom algorithm.*

- After labeling all the selected frames we accept the annotation and we export the labeled ground truth to the MATLAB workspace.

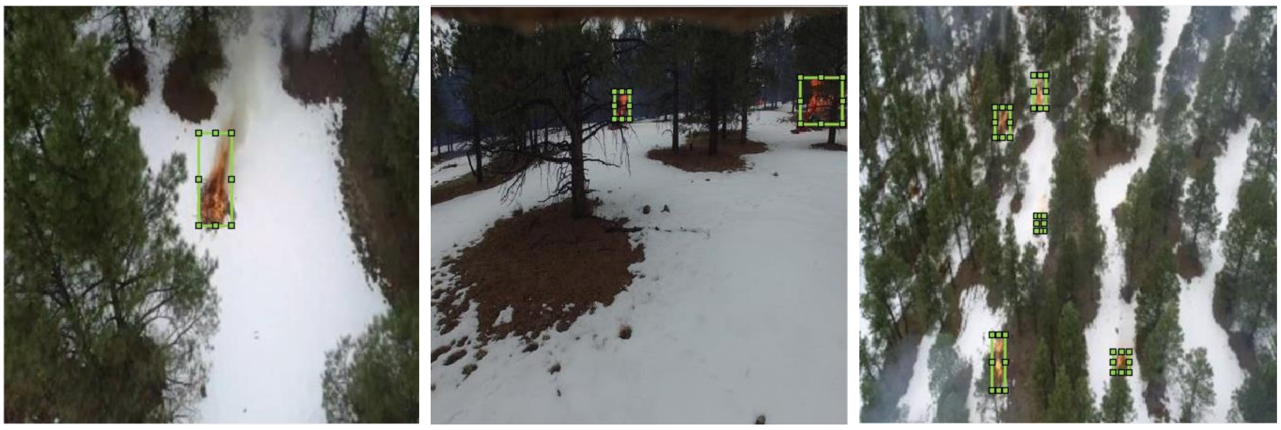


Figure III. 11*:Sample frames of fire bounding boxes automated labeling results.*

- **Automated Label Summary Discussion:**

We've also evaluated our automatically generated labels through the visual summary of the labeled ground truth, which is represented in Figure III. 11**,** by the label summary graphs for the ROI distribution of FLAME Dataset [35].

We used the graph to examine the occurrence of labels over time and compare the frames, the frequency of labels and the distribution of ROIs generated by the fire detection custom algorithm where:

- The *x*-axis of the graph displays the numeric ID of each image in the dataset image collection, the total number of images loaded and manually labeled from the dataset is 4911 image frame.
- The *y*-axis displays the number of ROIs for each image in the dataset image collection, which means how many fire bounding boxes are in each single image frame.as we can see in the Figure III. 12 most of the frames are labeled with the average of 5 to 10 bonding boxes per frame with an exception in some rare situations where there were frames that were labeled with up to 18 Bounding boxes.

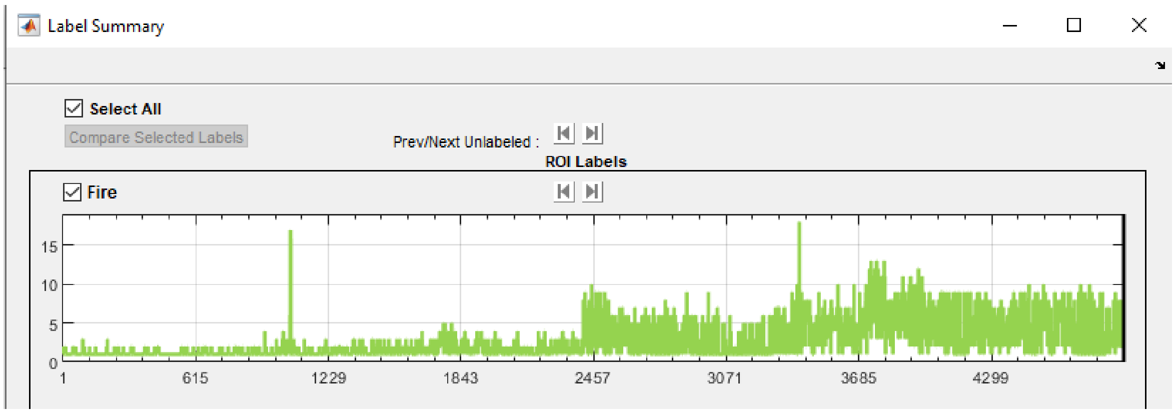


Figure III. 12:FLAME Dataset Lebel Summary Graph.

2.2.2.3 Labels Exported Format:

The Bounding boxes generated from the labeling are exported into the MATLAB workspace and specified as an *M*-by-4 nonsparse numeric matrix of *M* bounding boxes. Each row, *M*, of the matrix defines a bounding box as an axis-aligned rectangle.

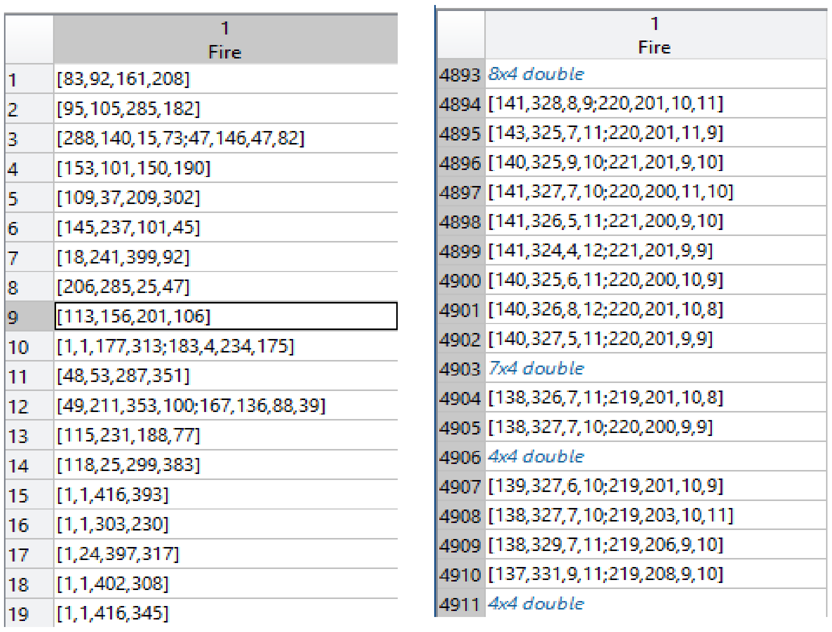
The format of the bounding boxes defined in pixel coordinates as an *M*-by-4 numeric matrix with rows of the form [*x y w h*], where [42]:

M is the number of axis-aligned rectangles.

x and y specify the upper-left corner of the rectangle.

w specifies the width of the rectangle, which is its length along the *x*-axis.

h specifies the height of the rectangle, which is its length along the *y*-axis.



	1 Fire
1	[83,92,161,208]
2	[95,105,285,182]
3	[288,140,15,73;47,146,47,82]
4	[153,101,150,190]
5	[109,37,209,302]
6	[145,237,101,45]
7	[18,241,399,92]
8	[206,285,25,47]
9	[113,156,201,106]
10	[1,1,177,313;183,4,234,175]
11	[48,53,287,351]
12	[49,211,353,100;167,136,88,39]
13	[115,231,188,77]
14	[118,25,299,383]
15	[1,1,416,393]
16	[1,1,303,230]
17	[1,24,397,317]
18	[1,1,402,308]
19	[1,1,416,345]

	1 Fire
4893	8x4 double
4894	[141,328,8,9;220,201,10,11]
4895	[143,325,7,11;220,201,11,9]
4896	[140,325,9,10;221,201,9,10]
4897	[141,327,7,10;220,200,11,10]
4898	[141,326,5,11;221,200,9,10]
4899	[141,324,4,12;221,201,9,9]
4900	[140,325,6,11;220,200,10,9]
4901	[140,326,8,12;220,201,10,8]
4902	[140,327,5,11;220,201,9,9]
4903	7x4 double
4904	[138,326,7,11;219,201,10,8]
4905	[138,327,7,10;220,200,9,9]
4906	4x4 double
4907	[139,327,6,10;219,201,10,9]
4908	[138,327,7,10;219,203,10,11]
4909	[138,329,7,11;219,206,9,10]
4910	[137,331,9,11;219,208,9,10]
4911	4x4 double

Figure III.13*: Ground Truth bounding boxes coordinates tables.*

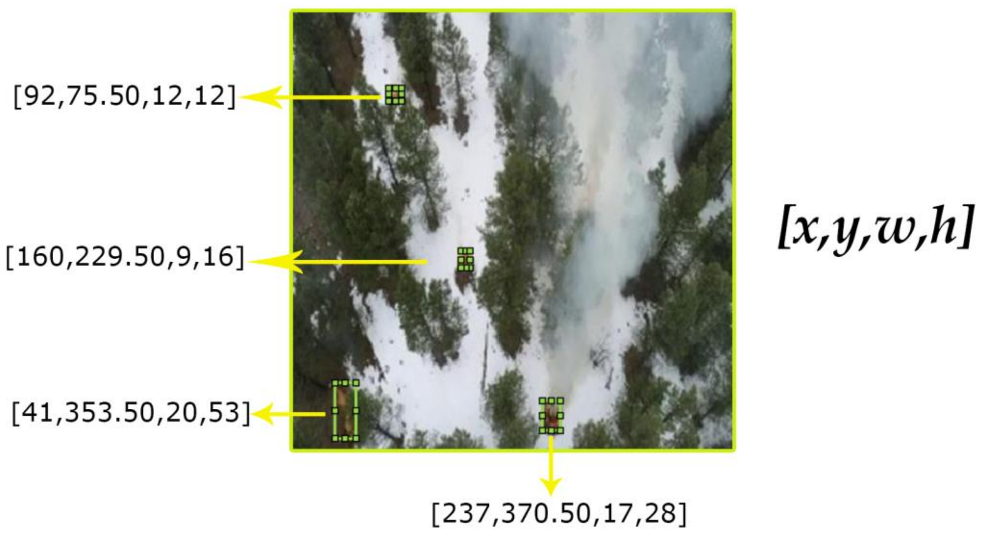


Figure* III.14:Ground Truth bounding boxes coordinates Sample.*

2.2.3 Ground Truth Testing

In machine learning, a properly labeled dataset that we use as the objective standard to train and assess a given model is often called “ground truth.”

Definition: Ground truth is a term used in statistics and machine learning that means checking the results of machine learning for accuracy against the real world. The term is borrowed from meteorology, where "ground truth" refers to information obtained on site.

The accuracy of our trained model will depend on the accuracy of the used ground truth for that we need to test it by inputting a raw Image from the dataset and see the annotation output.

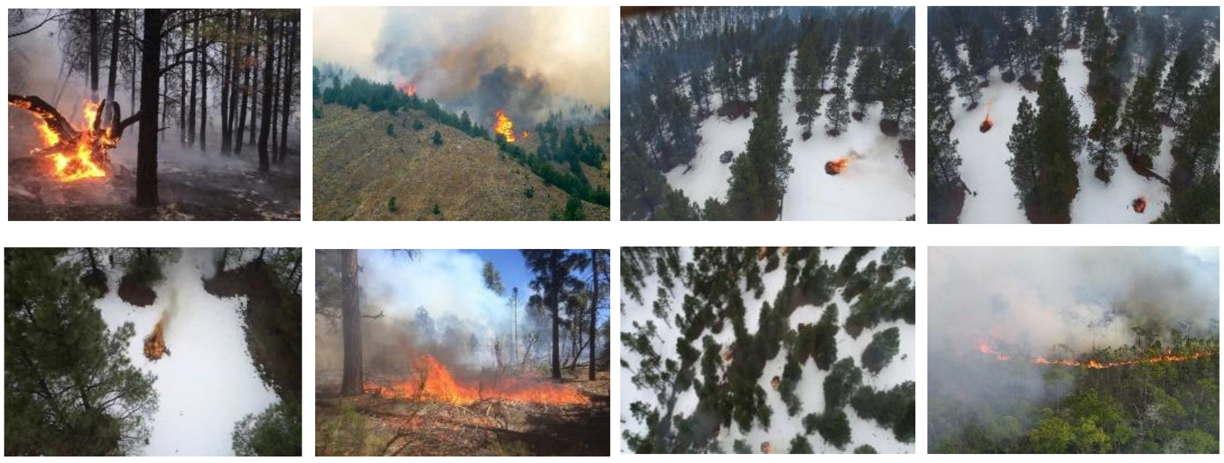


Figure III.15*:Sample Frames from ALL New Dataset for Ground Truth testing.*

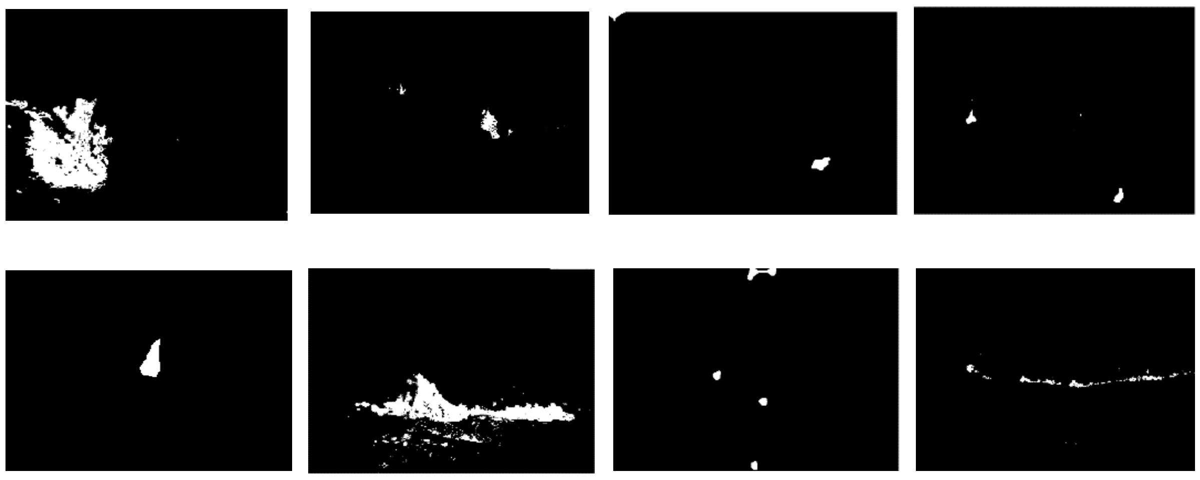


Figure III.16:*Sample Frames from Binary Mask testing results.*

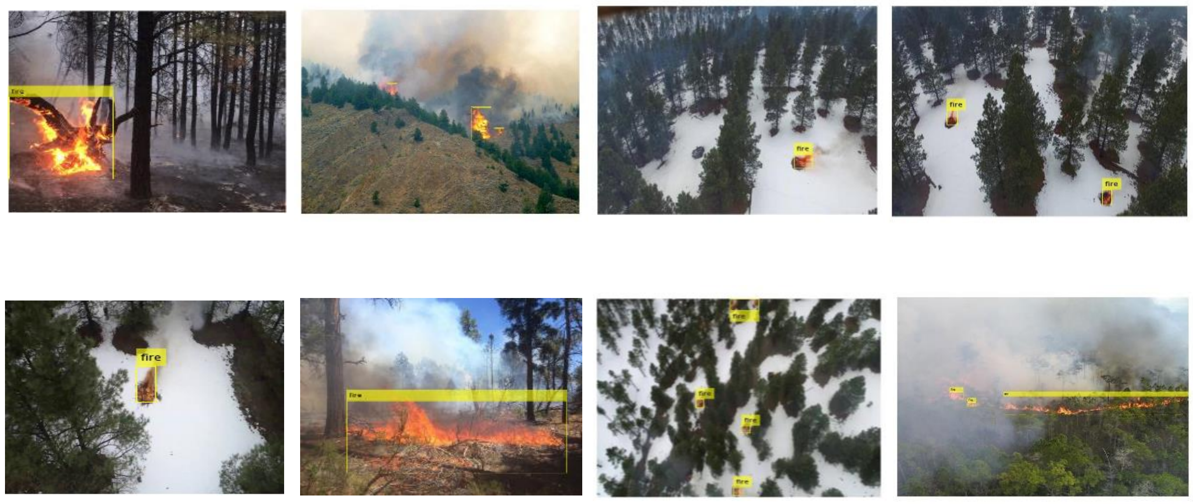


Figure III.17: Sample Frames from Bounding Box testing results.

“Garbage In Garbage Out” is commonly used phrase in the machine learning community, which means that the quality of the training data determines the quality of the model so

After investing 65% of our time and effort to ensure the quality and the quantity of the Data it is time to ensure the success of the Detection Model.

3. Fire Detection Model:

After the data are collected and annotated, we can proceed to model a computer vision model for which we would build, feed, train, test, and deploy and since machine learning algorithms are a base over which we enable deep learning and computer vision models, the first step is to have a framework built for the computer vision model for that we will use the YOLOv2 framework.

3.1 YOLOv2 Network Architecture:

YOLOv2 is a classic one-stage object detection algorithm targeted for real-time processing The YOLOv2 model runs a deep learning CNN on an input image to produce network predictions. The object detector decodes the predictions and generates bounding boxes.

Compared to traditional two-stage detection algorithms (such as R-CNN, Fast R-CNN, Faster-RCNN, etc.), YOLOv2 directly converts the problem of bounding box positioning into an end-to-end regression solution. Since YOLOv2 avoids the process of generating hundreds of candidate boxes, the execution speed of the algorithm is significantly improved over the two-stage detection schemes, which makes YOLOv2 an excellent choice for implementing real-world applications. The whole detection flow of the YOLOv2 algorithm can be generally divided into three basic procedures:

preprocessing of the input image: It involves resizing images of different input resolutions to a uniform size by using bilinear interpolation, and then subtracting the average brightness of all the images in the dataset to avoid distortion effect, such as over brightness.

the main CNN forward computation: in the YOLOv2 algorithm, the input image is divided into several grids according to the frame size, e.g., 13 × 13 grids for a 416×416 input image, and each grid is responsible for predicting five anchor boxes with different aspect ratio. Corresponding to each anchor box, the CNN outputs a 25- dimensional vector: one number for the probability the box contains an object, four numbers to represent the bounding box coordinates in relation to the anchor box, and 20-dimensional probability for each of the categories in the training dataset.

post-processing: which extracts the coordinates of the bounding box and the category information from the output of the CNN, and then filters them by performing non-maximum suppression (NMS) to get the best detection result. The bounding boxes of the objects are finally drawn and displayed for the user [26].

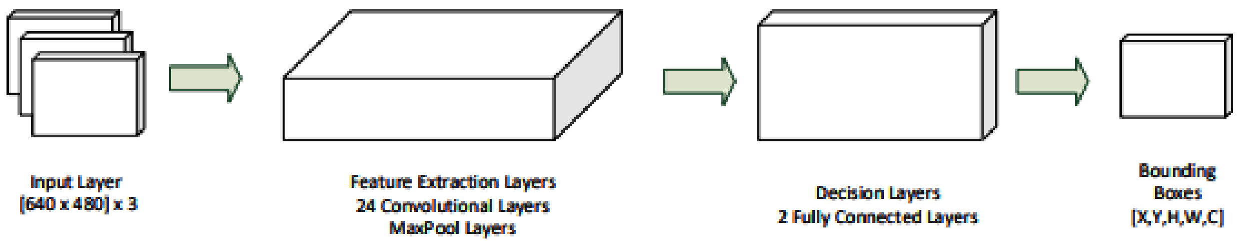


Figure III.18*:The YOLO network architecture [25]*

Each frame is divided into a grid of $\boldsymbol{s_x \cdot s_y}$ in each grid cell, a maximum number of $\boldsymbol{B}$ bounding boxes are searched; for every bounding box, a certain confidence score is assigned. The $\boldsymbol{Confidence}$ score is the probability that inside a bounding box an object is contained and how accurate the decision is. This $\boldsymbol{Confidence}$ is expressed as $\boldsymbol{p(Object)}$.Thus, each bounding box consists in five values: $\boldsymbol{X, Y, W, H}$ and, $\boldsymbol{Confidence}$. Furthermore, for each grid cell, the probability of the $\boldsymbol{C}$ conditional class is computed. In this way, it is possible to obtain for the entire image N bounding boxes, divided into grids. This leads to [25]:

$$\boldsymbol{p(Class_i|Object).p(Object).IOU_{pred}^{trut} = \; p(Class_i).IOU_{pred}^{truth} \quad (1)}$$

for a total of $\boldsymbol{s_x \cdot s_y \cdot (B \cdot 5 + C)}$ numbers of tensors for iteration, where:

$\boldsymbol{p(Class_i|Object)}$: Given a grid cell with at least one object, it represents the conditional probability for one of them to belong to the $\boldsymbol{i}$th class.

$\boldsymbol{p(Object).IOU_{pred}^{truth}}$: It represents the $\boldsymbol{Confidence}$ of each bounding box inside a cell grid. In each cell, only the classes with the highest Confidence are taken [25].

We summarize some of the important computational characteristics of the YOLOv2 algorithm in Table III. 4 where #Param represents the total number of parameters of the CNN model and #FLOP represents the total amount of floating-point operations (each MAC operation contains one floating point addition and one multiplication), which changes according to the input resolutions. Detection accuracy is measured in terms of mean Average Precision (mAP), and the scores shown in Table III. 4 are based on the PASCAL VOC 2007 dataset [26]

Input Resolution	#Param($\mathbf{10^6}$)	#Flop($\mathbf{10^9}$)	mAP
288×288	50.6	14.1	69
352×352		21.0	73.7
416×416		29.4	76.8
480×480		39.1	77.8

Table III. 4:*computational characteristics of the YOLOv2 [25].*

3.2 YOLOv2 Proposed Network Design:

In this work we built using MATLAB20 a YOLOv2 Model layer by layer from scratch based on the original YOLOv2 Model, starting with an input layer, followed by the detection subnetwork containing a series of Convolutional, Batch normalization, and ReLu layers. These layers are then connected by the MATLAB's inbuilt yolov2TransformLayer and yolov2OutputLayer.

yolov2TransformLayer transforms the raw CNN output into a form required to produce object detections. Yolov2OutputLayer defines the anchor box parameters and implements the loss function used to train the detector. Using the network analyzer app, we can visualize the layers graph of the built Model (Figure III.19).

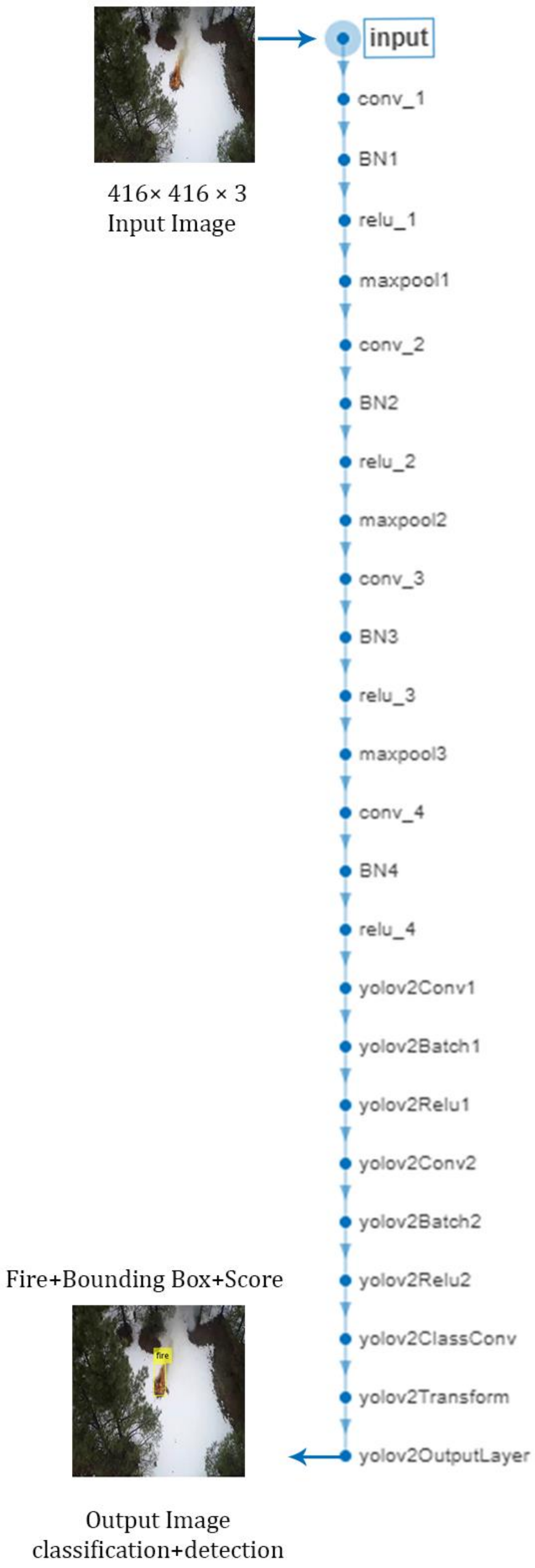
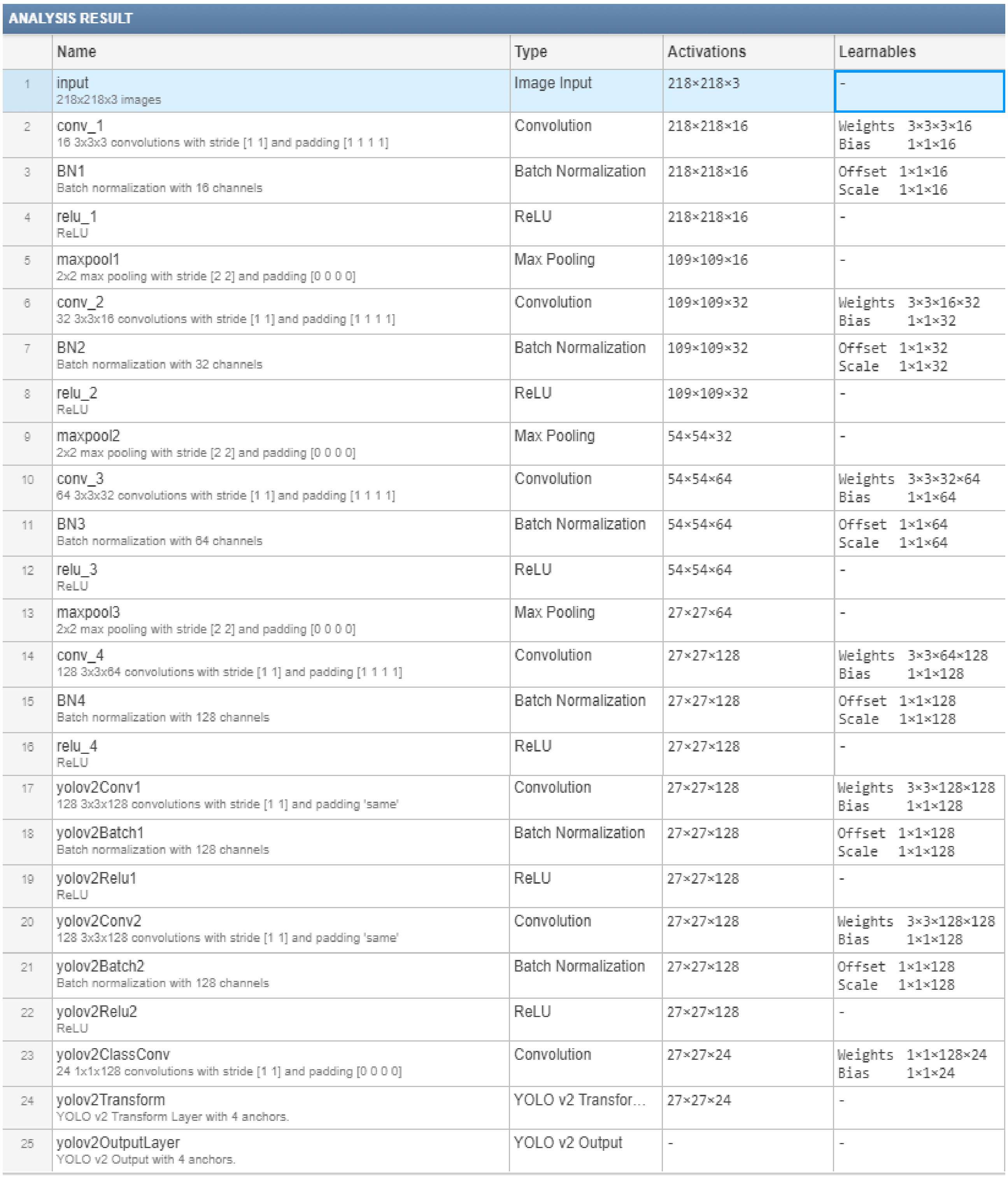


Figure III.19:*Network Architecture Layers.*



ANALYSIS RESULT

	Name	Type	Activations	Learnables
1	input 218x218x3 images	Image Input	218×218×3	-
2	conv_1 16 3x3x3 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	218×218×16	Weights 3×3×3×16 Bias 1×1×16
3	BN1 Batch normalization with 16 channels	Batch Normalization	218×218×16	Offset 1×1×16 Scale 1×1×16
4	relu_1 ReLU	ReLU	218×218×16	-
5	maxpool1 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	109×109×16	-
6	conv_2 32 3x3x16 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	109×109×32	Weights 3×3×16×32 Bias 1×1×32
7	BN2 Batch normalization with 32 channels	Batch Normalization	109×109×32	Offset 1×1×32 Scale 1×1×32
8	relu_2 ReLU	ReLU	109×109×32	-
9	maxpool2 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	54×54×32	-
10	conv_3 64 3x3x32 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	54×54×64	Weights 3×3×32×64 Bias 1×1×64
11	BN3 Batch normalization with 64 channels	Batch Normalization	54×54×64	Offset 1×1×64 Scale 1×1×64
12	relu_3 ReLU	ReLU	54×54×64	-
13	maxpool3 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	27×27×64	-
14	conv_4 128 3x3x64 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	27×27×128	Weights 3×3×64×128 Bias 1×1×128
15	BN4 Batch normalization with 128 channels	Batch Normalization	27×27×128	Offset 1×1×128 Scale 1×1×128
16	relu_4 ReLU	ReLU	27×27×128	-
17	yolov2Conv1 128 3x3x128 convolutions with stride [1 1] and padding 'same'	Convolution	27×27×128	Weights 3×3×128×128 Bias 1×1×128
18	yolov2Batch1 Batch normalization with 128 channels	Batch Normalization	27×27×128	Offset 1×1×128 Scale 1×1×128
19	yolov2Relu1 ReLU	ReLU	27×27×128	-
20	yolov2Conv2 128 3x3x128 convolutions with stride [1 1] and padding 'same'	Convolution	27×27×128	Weights 3×3×128×128 Bias 1×1×128
21	yolov2Batch2 Batch normalization with 128 channels	Batch Normalization	27×27×128	Offset 1×1×128 Scale 1×1×128
22	yolov2Relu2 ReLU	ReLU	27×27×128	-
23	yolov2ClassConv 24 1x1x128 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	27×27×24	Weights 1×1×128×24 Bias 1×1×24
24	yolov2Transform YOLO v2 Transform Layer with 4 anchors.	YOLO v2 Transfor...	27×27×24	-
25	yolov2OutputLayer YOLO v2 Output with 4 anchors.	YOLO v2 Output	-	-

.***Figure III.20:****Network Layers Analysis*

Once the network layers are ready and fully built, we moved to the training phase where we train our model on precedent-built Dataset.

for the training and testing process we first need to feed our network with the annotated fire detection datasets that we split the dataset into 75% Training Set and 25% Testing Set as shown in the table below.

	#Images
▪ Training Set	4,944
▪ Testing Set	1,648

Table III. 5:*The dataset distribution Table*

3.3 YOLOv2 Network Training Process:

3.3.1 Overview:

a) Network Training Workflow :

The training stage is the stage where the Deep learning algorithm is trained by feeding datasets, that’s where the learning takes place.

Deep learning neural network models learn to map inputs to outputs given a training dataset of examples, it consists of the sample output data and the corresponding sets of input data that have an influence on the output. all neurons of a given layer are generating an output, but they don’t have the same weight for the next neurons layer. This means that if a neuron on a layer observes a given pattern it might mean less for the overall picture and will be partially or completely muted. This is what is called Weighting: a big weight means that the Input is important and of course a small weight means that we should ignore it. Every Neural Connection between neurons will have an associated weight. And this is the magic of neural network adaptability: weights will be adjusted over the training to fit the objectives we have set (recognizing fire in an image Frame). In simple terms: Training a NN means finding the appropriate weights of the neural Connections thanks to a feedback loop called Gradient Backward propagation.

This training process is iterative, meaning that it progresses step by step with small updates to the model weights each iteration and, in turn, a change in the performance of the model each iteration. This iterative process which is called “model fitting” solves an optimization problem that finds for parameters (model weights) that result in a minimum error or loss when evaluating the examples in the training dataset.

Supervised learning is possible when the training data contains both the input and output values. Each set of data that has the inputs and the expected output is called a supervisory signal. The training is done based on the deviation of the processed result from the documented result when the inputs are fed into the model.

b) Network Training Options:

For training a network we always provide the algorithm with some training options. These options guide the network through how the network should learn.

- ***solver:*** it orchestrates model optimization by coordinating the network's forward inference and backward gradients to form parameter updates that attempt to improve the loss. The responsibilities of learning are divided between the Solver for overseeing the optimization and generating parameter updates and the Net for yielding loss and gradients.
- ***Batch Size***. *is* the number of examples used to estimate the error gradient before updating the model parameters.

Once an error gradient has been estimated, the derivative of the error can be calculated and used to update each parameter. There may be statistical noise in the training dataset and in the estimate of the error gradient. Also, the depth of the model (number of layers) and the fact that model parameters are updated separately means that it is hard to calculate exactly how much to change each model parameter to best move down the whole model down the error gradient. Instead, a small portion of the update to the weights is performed each iteration. A hyperparameter called the "*learning rate*" controls how much to update model weights and, in turn, controls how fast a model learns on the training dataset.

- ***Learning Rate****:* The amount that each model parameter is updated per cycle of the learning algorithm.

The training process must be repeated many times until a good or good enough set of model parameters is discovered. The total number of iterations of the process is bounded by the number of complete passes through the training dataset after which the training process is terminated. This is referred to as the number of trainings "*epochs*."

– ***Epochs****:* The number of complete passes through the training dataset before the training process is terminated.

There are many extensions to the learning algorithm, although these five hyperparameters generally control the learning algorithm for deep learning neural networks.

3.3.2 Configuration of the YOLOv2 Network Training Options:

Once we have the Network layers ready, we moved to work on our model's training options. The solver, Learning *Rate, mini batch size, and no of epochs* that we've discussed above are some of the important training options to consider in order to decide what should be the learning speed of the network and how much data sample the network should train on, in each round of training.

For the YOLOv2 Network Training we set some options based on the size of the dataset where we've:

– Set the solver to use Adam (adaptive moment estimation) solver.
– Set the initial learning rate to 0.001 Here we performed lower learning rate to give more time for training considering the size of data.
– Set the size of mini-batch to use for each training iteration to16, we performed lower size of mini-batch to reduce memory usage during training.
– Set the maximum number of epochs for training to 30.
– Specify the frequency of verbose printing to 30.

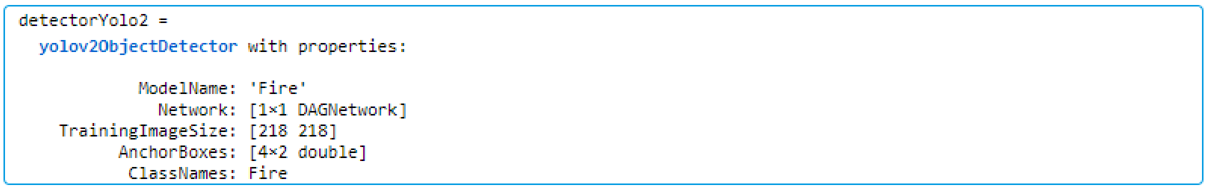
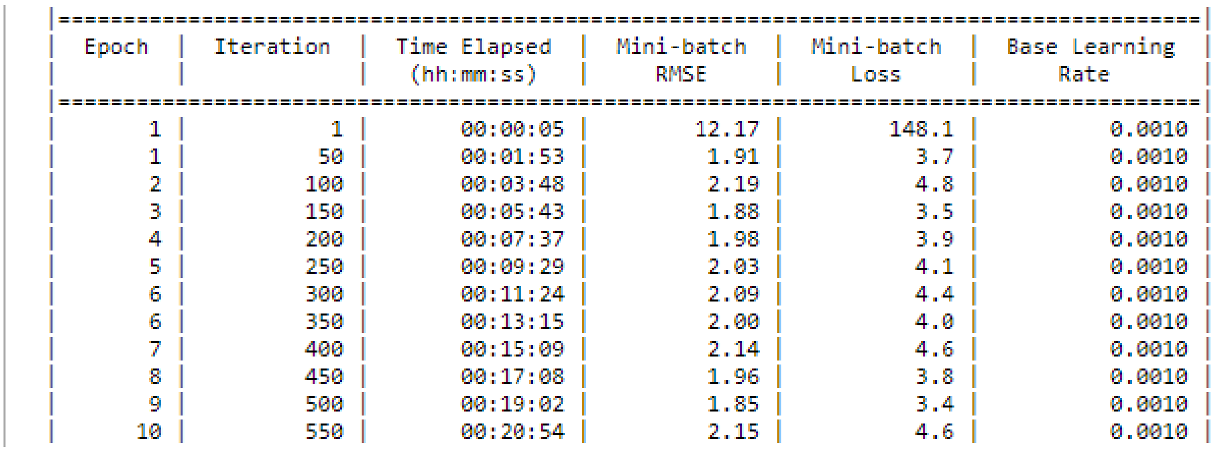

```
detectorYolo2 =
  yolov2ObjectDetector with properties:

            ModelName: 'Fire'
              Network: [1×1 DAGNetwork]
    TrainingImageSize: [218 218]
          AnchorBoxes: [4×2 double]
           ClassNames: Fire
```

Figure III.21:*YOLOv2 Detector Properties*

The training was conducted using a system with the following specifications: Intel®Xeon®CPU E5-1650 v2 @ 3.50 GHz with 64 GB onboard memory; The required environments and tools (MATLAB 2020). The weight in the network is initialized randomly and the other training options have been set and discussed above.

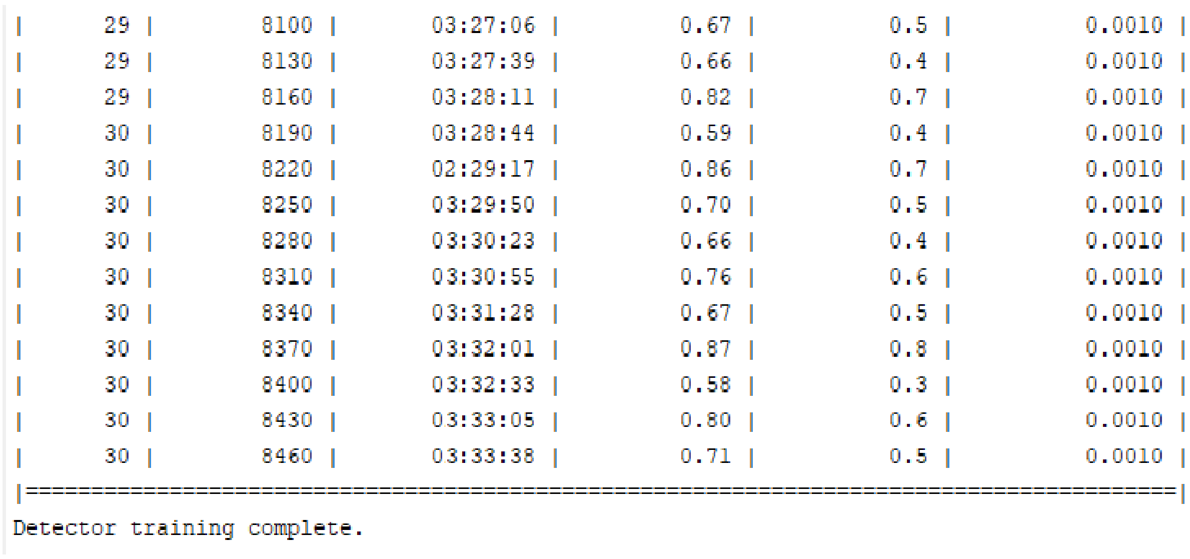
YOLOv2 training stages are carried out to get the best model which will then be used to recognize and detect fire patterns. The training process in this study takes about 2 to 5 minutes per epoch. The length of training time is affected by the batch size value, and the number of epochs, for 50 epochs the computing took around 3 hours and 33 minutes to complete the model training.

Figure III.22 and Figure Figure III.23 below show the training progress through each iteration with mini batch and base learning rate information.



```
|========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |     RMSE     |     Loss     |      Rate       |
|========================================================================================|
|       1 |           1 |       00:00:05 |        12.17 |        148.1 |          0.0010 |
|       1 |          50 |       00:01:53 |         1.91 |          3.7 |          0.0010 |
|       2 |         100 |       00:03:48 |         2.19 |          4.8 |          0.0010 |
|       3 |         150 |       00:05:43 |         1.88 |          3.5 |          0.0010 |
|       4 |         200 |       00:07:37 |         1.98 |          3.9 |          0.0010 |
|       5 |         250 |       00:09:29 |         2.03 |          4.1 |          0.0010 |
|       6 |         300 |       00:11:24 |         2.09 |          4.4 |          0.0010 |
|       6 |         350 |       00:13:15 |         2.00 |          4.0 |          0.0010 |
|       7 |         400 |       00:15:09 |         2.14 |          4.6 |          0.0010 |
|       8 |         450 |       00:17:08 |         1.96 |          3.8 |          0.0010 |
|       9 |         500 |       00:19:02 |         1.85 |          3.4 |          0.0010 |
|      10 |         550 |       00:20:54 |         2.15 |          4.6 |          0.0010 |
```

Figure III.22*:Training Progress's information for the first 10 epochs.*



```
|      29 |        8100 |       03:27:06 |         0.67 |          0.5 |          0.0010 |
|      29 |        8130 |       03:27:39 |         0.66 |          0.4 |          0.0010 |
|      29 |        8160 |       03:28:11 |         0.82 |          0.7 |          0.0010 |
|      30 |        8190 |       03:28:44 |         0.59 |          0.4 |          0.0010 |
|      30 |        8220 |       02:29:17 |         0.86 |          0.7 |          0.0010 |
|      30 |        8250 |       03:29:50 |         0.70 |          0.5 |          0.0010 |
|      30 |        8280 |       03:30:23 |         0.66 |          0.4 |          0.0010 |
|      30 |        8310 |       03:30:55 |         0.76 |          0.6 |          0.0010 |
|      30 |        8340 |       03:31:28 |         0.67 |          0.5 |          0.0010 |
|      30 |        8370 |       03:32:01 |         0.87 |          0.8 |          0.0010 |
|      30 |        8400 |       03:32:33 |         0.58 |          0.3 |          0.0010 |
|      30 |        8430 |       03:33:05 |         0.80 |          0.6 |          0.0010 |
|      30 |        8460 |       03:33:38 |         0.71 |          0.5 |          0.0010 |
|========================================================================================|
Detector training complete.
```

Figure III. 23*:Training Progress's information for the last epochs.*

3.4Training Evaluation:

In order to evaluate the training process an error function must be chosen, often called the objective function, cost function, or the loss function. Typically, a specific probabilistic framework for inference is chosen called Maximum Likelihood. Under this framework, the commonly chosen loss functions are cross entropy for classification problems and mean squared error for regression problems [43].

a) ***Loss Function:*** The function used to estimate the performance of a model with a specific set of weights on examples from the training dataset.

The search or optimization process requires a starting point from which to begin model updates. The starting point is defined by the initial model parameters or weights. Because the error surface is non-convex, the optimization algorithm is sensitive to the initial starting point. As such, small random values are chosen as the initial model weights [25].

From a very simplified perspective, the loss function (J) can be defined as a function which takes in two parameters [43]:

- Predicted Output.
- True Output.

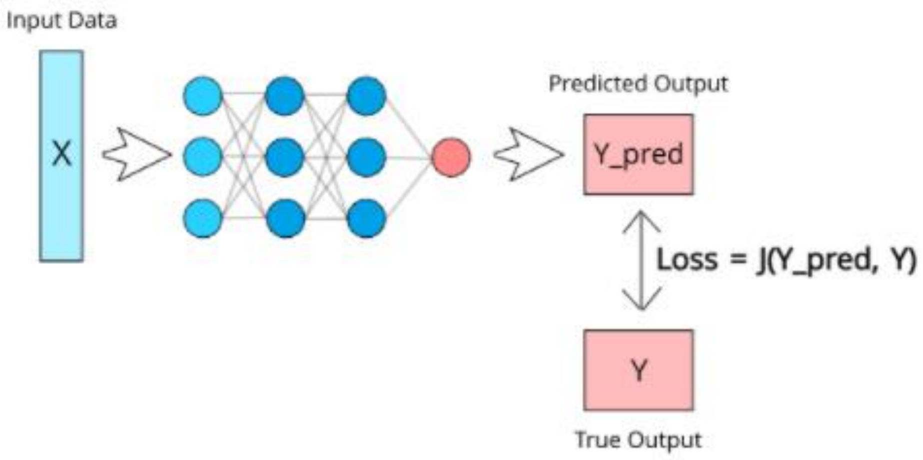


Figure III.24:*Neural Network Loss Visualization [43].*

This function will essentially calculate how poorly our model is performing by comparing what the model is predicting with the actual value it is supposed to output which is our ground truth.

If Y pred is very far off from Y, the Loss value will be very high. However, if both values are almost similar, the Loss value will be very low.

If the loss is very high, this huge value will propagate through the network while it's training and the weights will be changed a little more than usual. If it's small then the weights won't change that much since the network is already doing a good job.

The generated training loss graph is shown in Figure III. 25 below.

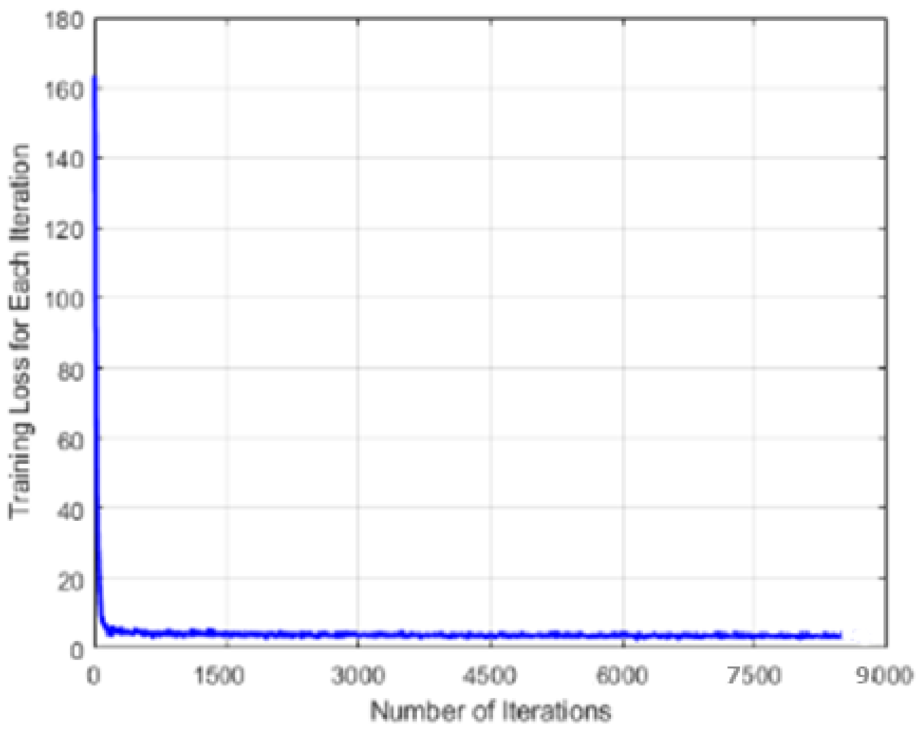


Figure III.25*:Training Loss Graph*

b) Training Loss Graph discussion:

It was inferred from the graph that while the total loss value was 164 at the starting of the training it was decreased to level of 0.00 at the end of the training at the steps of 8460. As the loss function was lower than the level of 0.02, the inference graph was frozen and exported to detect the fire source, which proves that the network is efficiently trained and ready to be tested.

4. Conclusion:

This chapter was devoted to discuss the process of building the fire detection deep learning model from collecting and labeling the necessary high quality data to designing the YOLOv2 Network Layers and training a reasonably accurate model ,after the evaluation of the training we need to move to the next step which is the testing phase, The next chapter is dedicated to expose this important phase where we will test our trained model on the test set ,evaluate it , finally we expose the test and evaluation of the model for Real-Time Detection.

Chapter IV: Real-Time Model Evaluation and Testing

1.Introduction:

Fire Classification and localization is a supervised learning approach in which the target variable “fire” is discrete (or categorical) for that evaluating a deep learning model is considered as important as building and training it, in this chapter we will proceed to the next phase of our project which is the Testing phase where we will perform our built in Fire Detection Model on new, previously unseen realistic data and try to evaluate the results based on three created model scenarios ,finally we will cover up Real-time detection where the model should be able to sense the environment parse the scene to balance between detection performance and real time requirements.

2. Testing the Trained Model:

Deep learning Model Test is an important and critical step in the detection process that we can neither neglect nor over pass, this part of the process enables us to quickly identify any shortcomings or edge cases that the model does not perform well on and thus evaluate it based on the detection test performance.

According to this test results we can re-train our model using annotations from images depicting these edge cases, or at least be aware of the specific environments in which the model performs best.

therefore, once we have made the YOLOv2 model using the prepared training data, we can test it by feeding in new, real wildfire scenarios images (test test) and seeing whether the model classifies and localize the fire accurately, while this isn’t an automated or quantified test it was conducted using the same environment specifications that we used for the training: Intel®Xeon®CPU E5-1650 v2@ 3.50 GHz with 64 GB onboard memory; The required environments and tools (MATLAB 2020).

2.1. Test Set:

Our test set is composed of 1,648 Aerial Fire and Non Fire images that were extracted from a fire video that was filmed in north Florida, these test set highly simulate and counterfeit the environment and the fire scenarios that our model will perform on, note that we applied the same preprocessing transform (image resizing and labeling) with no augmentation to the test data as for the training data which we’ve discussed its process in chapter N°3.

The primary reason for feeding an already labeled data to the trained model is to be able to evaluate the model later according to the comparison between the exiting ground truth and the predicted outputs and other range of different metrics, thus test data should be representative of the original data and be left unmodified for unbiased evaluation.

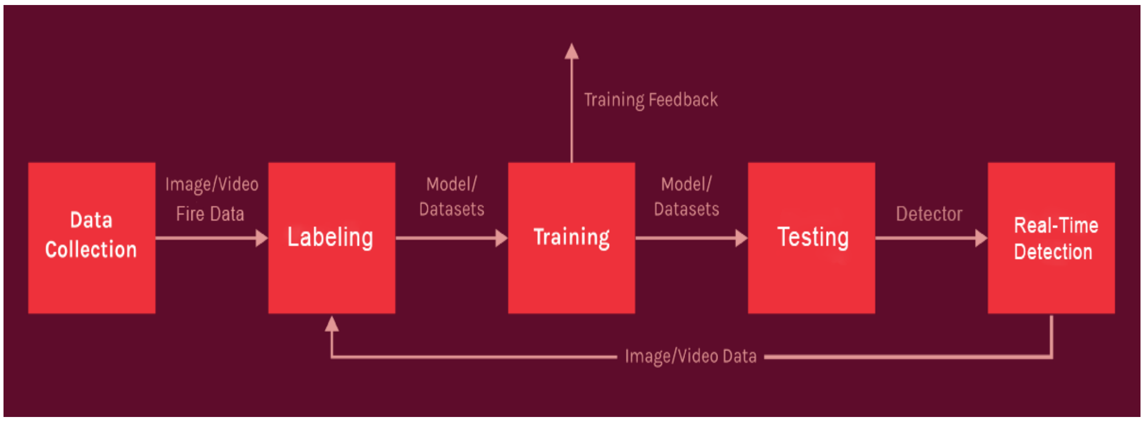


Figure IV. 1:*Project work Methodology*

2.2. Test Detection Scenarios:

For the test detection a total of three scenarios are made for experimental validation where we ran three different trained Fire Detectors (Models) on the same test set that we discussed above.

a) ***Scenario 1****:* the detector of the first scenario was trained on only aerial images obtained from the FLAME Dataset [35] that we discussed in chapter N°3, were we fed the model with 1,126 fire annotated images.

b) ***Scenario 2****:* the detector of the second scenario was trained on only terrestrial images obtained from the FiSmo [36] and Gaisaid [37] Datasets that we discussed in chapter N°3, were we fed the model with 850 fire annotated images.

c) ***Scenario 3:*** the last detector is the same detector that we discussed its mixed dataset (All New Dataset [41]), training process and training evaluation in chapter N°3.

Note that all the above scenarios were experimented using the same system environment and its training dataset was also preprocessed in the same way and have been fed to the same YOLOv2 network designed in chapter N°3

	Dataset Source	**Train set**
Scenario 1	FLAME Dataset [35]	1,126
Scenario 2	FiSmo [36] and Gaisaid [37] Datasets	850
Scenario 3	All New Dataset [41]	4,944

Table IV. 1:*Models trained Dataset information Table*

We adopted this approach of testing to demonstrate the efficacy and simulate the performance of each Trained detector once it would be deployed on UAV-based forest fire monitoring and detection missions for real situations.

2.3. Test Detection Results:

After running the three detectors on the same test set, we obtained the below Detection results:

a) Results Detection of Scenario 1:

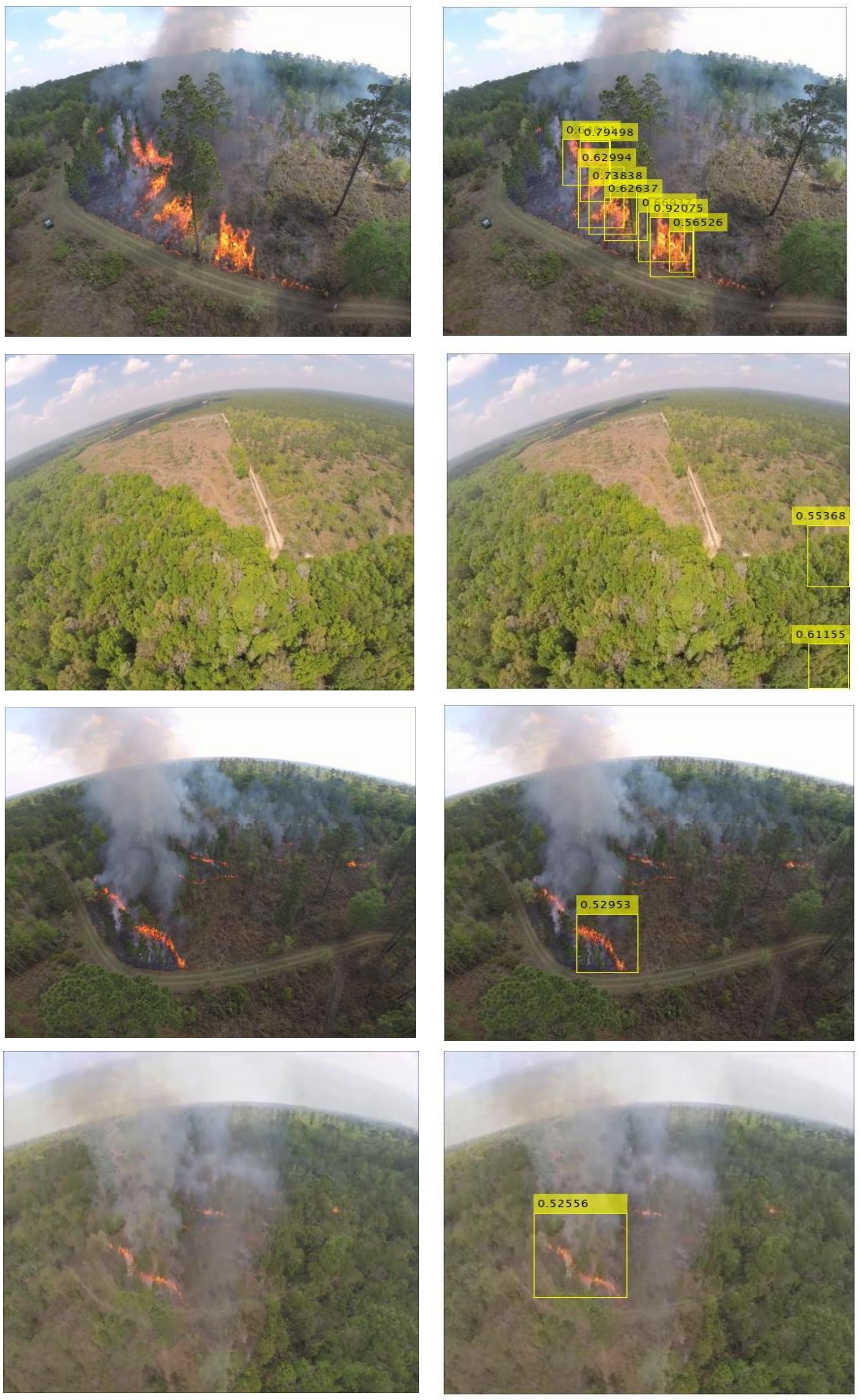


***Figure IV.** 2:Sample Frames from Testing the first Detector*

b) Results detection of Scenario 2:

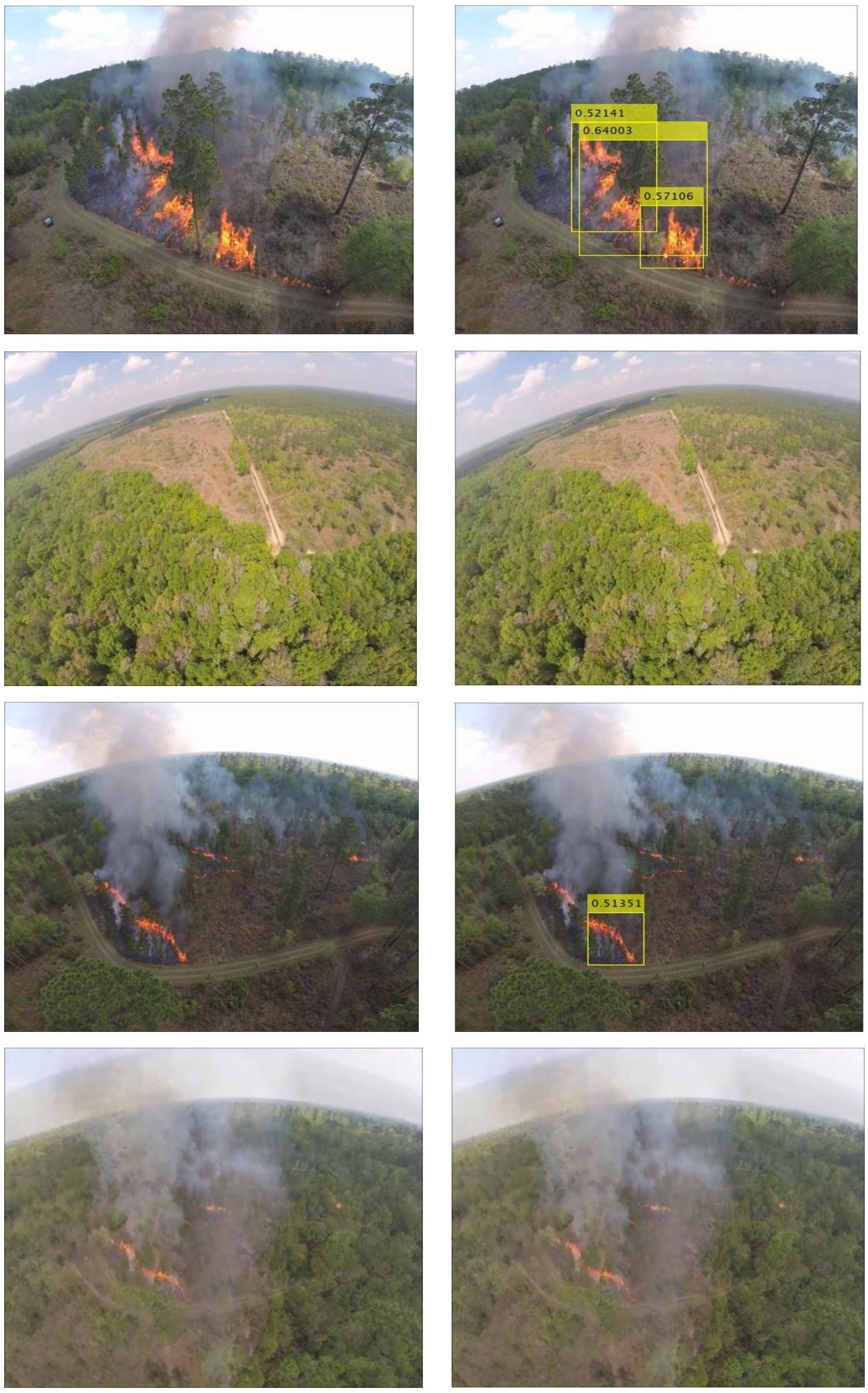


Figure IV. 3:*Sample Frames from Testing the second Detector*

c) *Results detection of Scenario 3:*

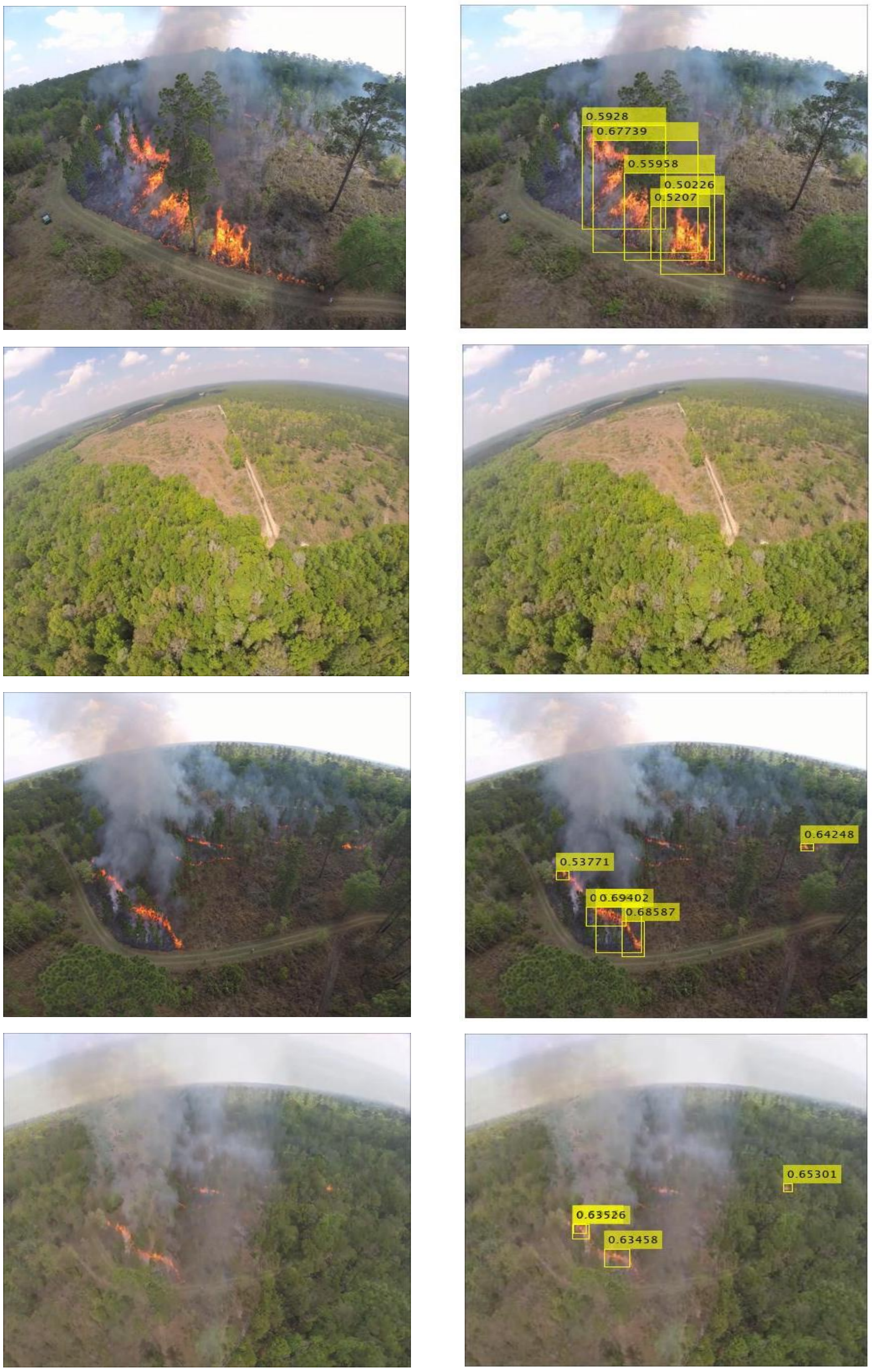


Figure IV. 4:*Sample Frames from Testing the Third Detector*

3. Model Evaluation:

An object detection model produces the output in three components:

1. The *bounding boxes* — x1, y1, width, height.
2. The *class* of the bounding box.
3. The *probability score* for that prediction— how certain the model is that the class is actually the predicted class [45].

Evaluating object detection model means that we need to measure if the model found all the objects, and also a way to verify if the found objects belong to the correct class. This implies that in our case our Fire detection model needs to accomplish two things:

1. *Classification:* Identify if a Fire spot is present in the image and its class.
2. *Localization:* Predict the coordinates of the bounding box around the Fire when it is present in the image. Here we compare the coordinates of ground truth and predicted bounding boxes.

To evaluate our model, we need to evaluate the performance of both classification as well as localization of using bounding boxes in the image using evaluation metrics, but before exploring the metrics we need to cover an important parameter —IoU.

– ***Intersection over Union (IoU):*** we use the concept of Intersection over Union (IoU) or so-called threshold as a similarity measure, it computes intersection (overlap) over the union of the two bounding boxes; the bounding box for the ground truth and the predicted bounding box as shown in figure where Red represents ground truth bounding box and green represent predicted bounding box.

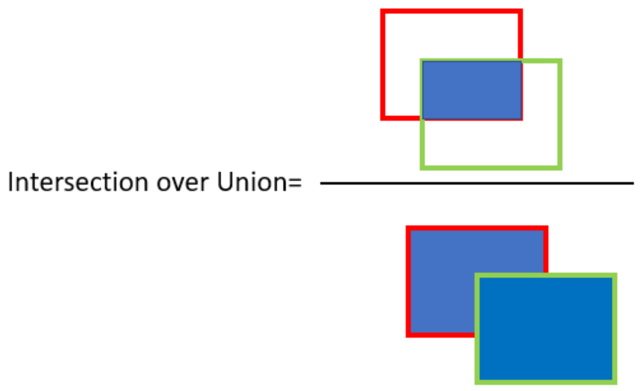


Figure IV. 5:*Intersection over Union [46].*

An IoU of 1 implies that predicted and the ground-truth bounding boxes perfectly overlap. Threshold is a key parameter for various evaluation Metrics, changing the value of the threshold parameter drastically changes the value of the evaluation metric. We can set a threshold value for the IoU to determine if the object detection is valid or not.

in our case we set the threshold value to 0.5 which means:

- if IoU **≥0.5,** classify the Fire detection as ***True Positive (TP*****).**
- if IoU **<0.5** then it is a wrong detection and classify it as ***False Positive (FP****)*.
- When a ground truth is present in the image and model failed to detect the Fire**,** classify it as ***False Negative (FN)***.
- ***True Negative (TN)****:* TN is every part of the image where we did not predict a Fire. This last metrics is not useful for object detection in general, hence we ignore TN.

3.1 Evaluation Metrics:

To evaluate our YOLOv2 Fire detector, we will use three main evaluation metrics that we will discuss below:

1) Mean Average Precision (mPA):

it's the most used metric to evaluate an object detection model it's calculated using three essential elements:

- ***Precision*** measures how good our model is when the prediction is positive [46].

$$pecision = \frac{TP}{TP+FP} \quad (1)$$

- ***Recall*** measures how good our model is at correctly predicting positive classes [46].

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

The focus of precision is positive predictions. It indicates how many positive predictions are true, while the focus of recall is actual positive classes**.** It indicates how many of the positive classes the model is able to predict correctly.

- ***Average precision*****:** for a multiclass detector, the average precision is a vector of average precision scores for each object class.

we count the accumulated TP and the accumulated FP and compute the precision/recall at each line, the Average Precision is computed as the average precision at 11 equally spaced recall levels. The Mean Average Precision (mAP) i**s** the averaged AP over all the object categories and since in our case we have only one object category which is" Fire" then we can imply that.

$$mAP = AP \quad (4)$$

2) **Log Average Miss-Rate (LAMR):**

it's calculated using two essential elements:

- ***Miss Rate*** **— Log miss rate:** [46]

$$mr(c) = \frac{FN}{TP+FN} \qquad (4)$$

- ***fppi*** **— False Positives Per Image:** [46]

$$fppi(c) = \frac{FP}{\#img} \qquad (5)$$

we first plot the *miss-rate the* number of *false positives per image* in *log-log* plots. all for a given confidence value c such that only detections are taken into account with a confidence value greater or equal than c. As commonly applied in object detection evaluation the confidence threshold c is used as a control variable. By decreasing c, more detections are taken into account for evaluation resulting in more possible true or false positives, and possible less false negatives. We define the *log average miss-rate (LAMR)* as shown in (equation (6) [46].

$$LAMR = exp(\frac{1}{9}\sum_{f} log(mr(\underset{fppi(c)\leq f}{\text{armax}} fppi\,(c)))) \quad (6)$$

3.2 Evaluation Results:

We applied the evaluation metric discussed above and we've obtained these results:

Note that due to the high non-rigidness of pedestrians we follow the common choice of an IoU threshold of 0.5.

a) Scenario 1 Results:

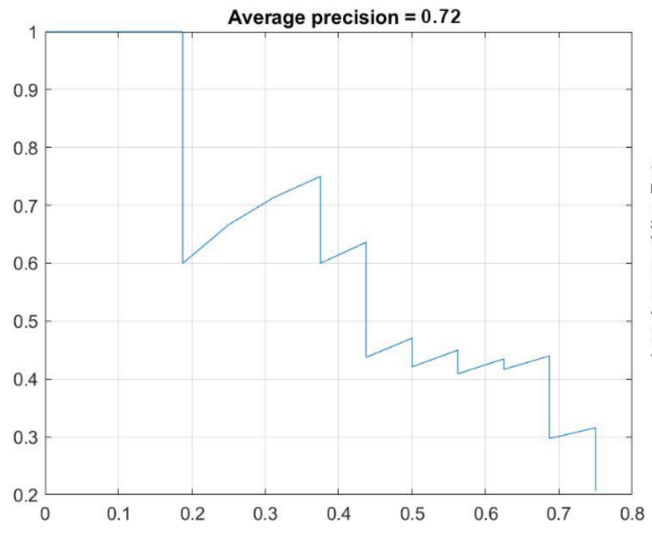


Figure IV. 6:Log Average Miss Rate of the first scenario

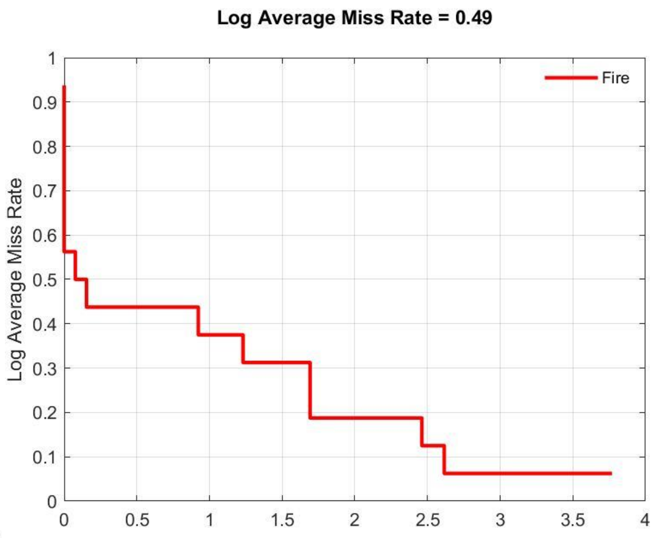
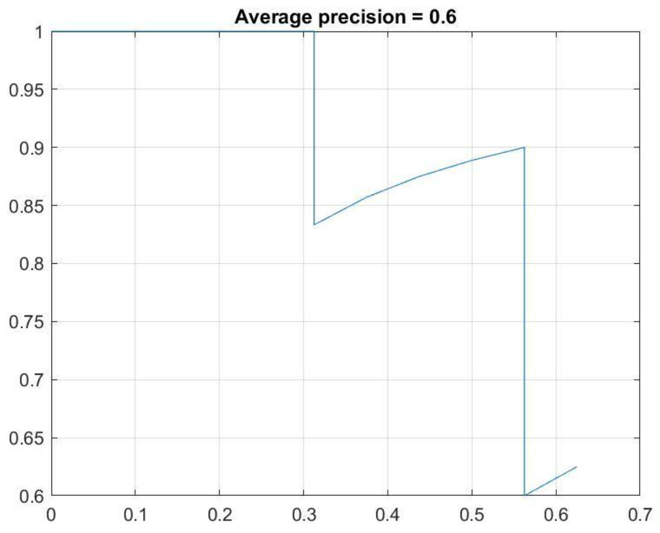


Figure IV.7:Log Average Miss Rate of the first scenario

b) Scenario 2 Results:



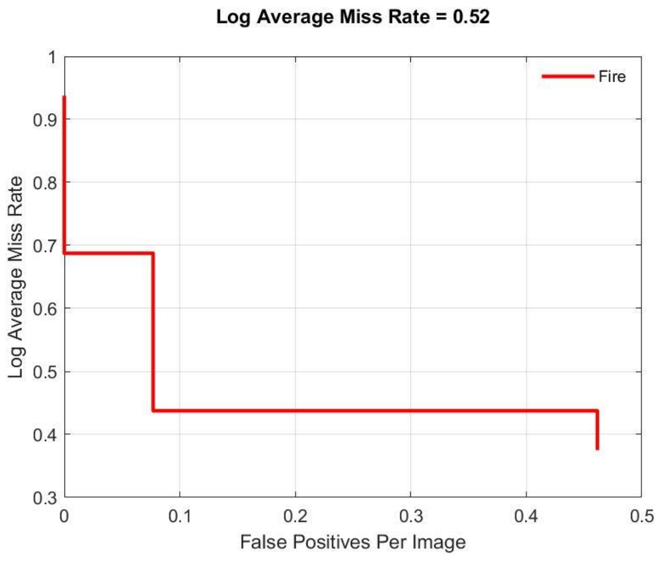
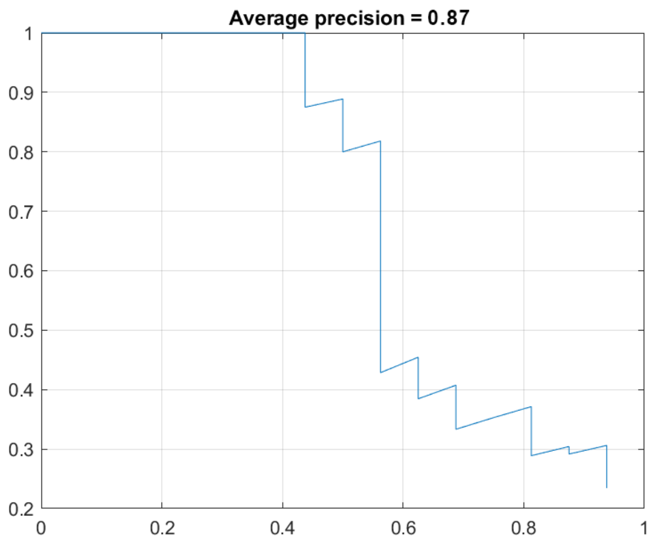


Figure IV.8:Log Average Miss Precision of the Seconde scenario/ Log Average Miss Rate of the Seconde scenario

c) Scenario 3 Results:



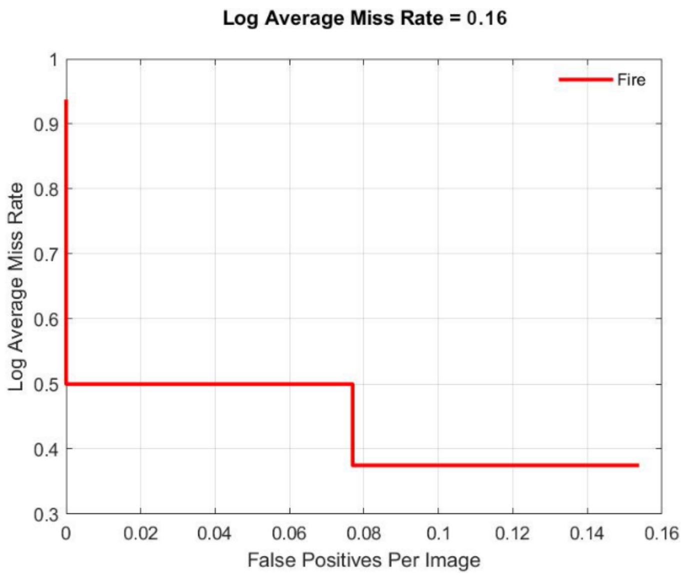


Figure IV.9:*Log Average Miss Precision of the Third scenario/ Log Average Miss Rate of the Third scenario*

3.3 Results Discussion:

the evaluation and the detection results show different values variations between the three scenarios:

a) Scenario 1 Results:

from Figure IV. 6 and Figure IV. 7 it shows that the first trained model achieved an average precision of 0.72 and an average mis rate of 0.49 ,which is considered as a high precision value that relates to a low false negative FN rate without neglecting the miss rate value which kinda not satisfying comparing to the precision ,and this is due to high false positive rate FP ,we can

false positive rates FP from the tree spots that were detected as fire (Figure IV. 2) which the train data plays an important factor in that as it is considered as a lower complex scene data.

b) .Scenario 2 Results:

from Figure IV. 8 it shows that the second trained model achieved an average precision of 0.6 and an average mis rate of 0.52,which is high miss rate that is related to a high false negative rate FN ,these values are connected directly to the difference between the test set and the train set ,the dataset that we trained our second model on is composed of multiple fire scenes but they are completely terrestrial images, while the dataset that we tested with is composed of purely aerial scenes and that's was obviously seen in Figure IV. 3 where the detector was incapable of detecting the fire spots were the aeras were less large and less clear.

c) Scenario 3 Results:

from Figure IV. 9 it shows that the third trained model achieved an average precision of 0.87 and an average miss rate of 0.16 which is considered as a high precision and a low miss rate, these results are related to the low false positive rate FP and false negative rate FN and that's what we can see clearly in Figure IV. 4 were most all of the spot fire were successfully detected. These detection results proves that the personalized dataset that we've built has overcame the challenges by gathering rich aerial fire senses and its detector satisfied the requirements of an object detection Model.

The table below summaries the Fire Detection test results of the three Detectors.

	Average Precision	**Log Average Miss Rate**
Scenario 1	0.72	0.52
Scenario 2	0.6	0.49
Scenario 3	0.87	0.16

***Table IV. 2**:summary of the three secanrios results.*

From these results we can say that we did the right thing by mixing the datasets and training the model on our own personalized dataset.

4. Real Time Detection Testing

4.1 Conducted experiment:

Real-time object detection models should be able to sense the environment, parse the scene and finally react accordingly, for that we must keep a balance between detection performance and real-time requirements.

To evaluate the detector performance in Real time it was required to do a UAV test where we deploy our Detector on a UAV equipped camera and test its detection and robust performance while the UAV is flying upon a fire region which is the real scenario that this system was built for ,unfortunately due to lack of resources and materials we couldn't achieve the real test so we tried to imitate the scenario as much as possible by replacing the UAV camera by a phone camera and try to detect it from a height of approximately 9.5 meters.

This Real time test was conducted using an environment of the following specifications:

- for shooting we used: Redmi Note 8 phone camera with 48MP.
- for processing we used Intel®Core (TM) i7-4600U CPU @ 2.10GHz ,2.70 GHz with 64 GB onboard memory.
- For the tools we used MATLAB20, and IP Webcam Android App.

We installed MATLAB Support Package for IP Cameras to bring live captured videos from camera phone into MATLAB.

4.2 Test Results:

The Results obtained from the conducted Real-Time detection test are shown in the Figure IV. 10 below.

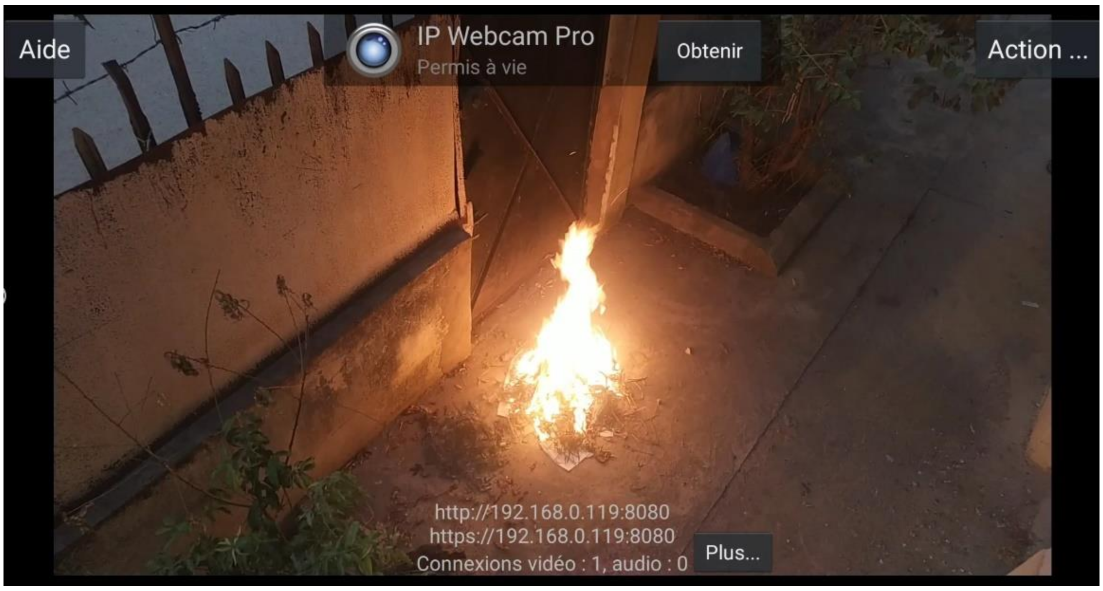

Aide
IP Webcam Pro
Permis à vie
Obtenir
Action ...
http://192.168.0.119:8080
https://192.168.0.119:8080
Connexions vidéo : 1, audio : 0
Plus...

Figure IV. 6:*Screenshot from the shooting phone*

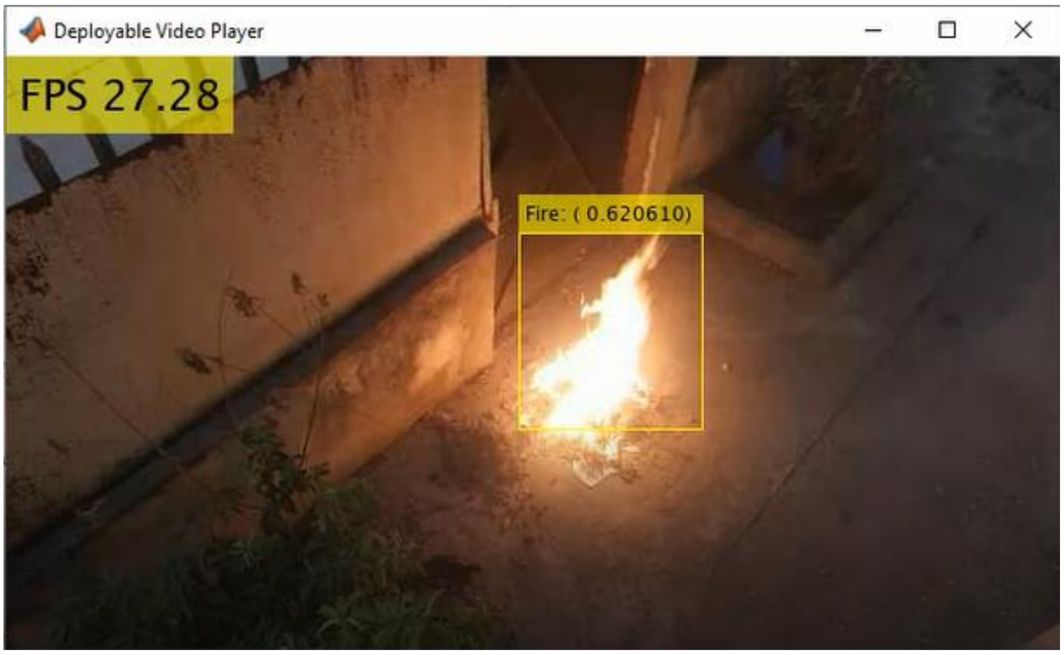


Figure IV. 7:*MATLAB live processing window for detection with 27.28 FPS.*

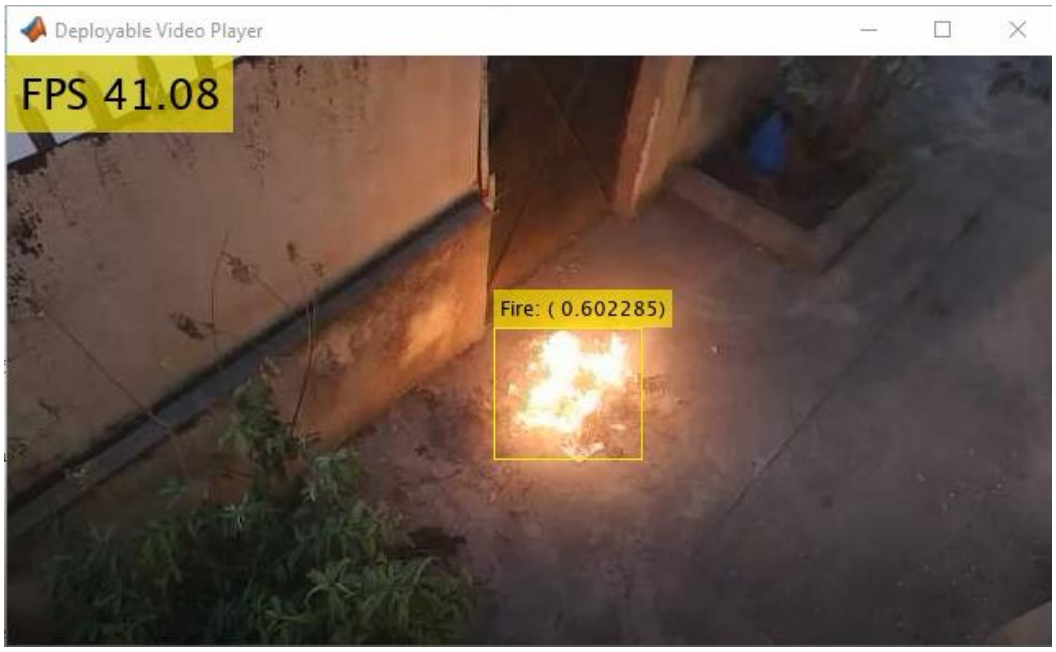


Figure IV. 8*:MATLAB live processing window for detection with 41.08 FPS.*

4.3 Results Discussion:

Figure IV.10, Figure IV.11 and Figure IV.12 above show the real-time test results ,in which we can clearly notice the detected fire spot with an average score of 0.6 ,besides the accuracy the final results has shown a rapid detection that reached out to 58 Frame per Image FPS regarding the poor calculation environment that we tested with ,these rapidity is due to the designed YOLOv2 Network model that utilize the predefined anchors that cover spatial position, scales, and aspect ratios across an image. Hence, we do not need an extra branch for extracting region proposals. Since all computations are in a single network, it likely to run faster than the two-stage detectors.

5 Conclusion

In this chapter we tested our built Model on an aerial fire images test set, we did evaluate our detector by comparing it to different trained Models using evaluation metrics in which its results has shown a mean average precision of 0.87,for further evaluation in real time environment we conducted a Real-Time detection test that end up with a very fine results, our detector has shown a rapid and robust detection that reached out to 58 FPS ,these parameters satisfy completely the requirements of a Real-Time object detector.

Conclusion and Perspectives

Forest Fires is a major problem, when a fire occurs, it seriously threatens people’s lives and causes major losses, a lot of work based on computer vision are made to detect these fires by exploiting the use of UAVs for aerial forest monitoring.

After a definition of the treated problem which concerns the Fire classification and localization, we have exposed the state of the art on the work that is already done on the same problem, we have chosen to solve this problem by using deep learning, more specifically the YOLOv2 Convolutional Neural Network Architecture.

To build this deep learning computer vision-based fire detector we had to provide two essential elements the dataset and the trained model, In deep learning dataset is so crucial ,in our way to craft this element we encountered several challenges ,the first one was the absence of benchmark dataset for forest fire detection ,so we had to surf through various resources to collect the necessary amount of data ,the second challenge is that most of the collected data were not labeled or annotated ,its labeling process in MATLAB was very time and energy consuming.

For model crafting we used a main single approach: building the model "from scratch" which requires to design the network architecture and train it. The advantage of the latter approach is the possibility of customizing the layers and the training options as we want till the validation of the model. we tried to keep a balance between detection performance and real-time requirement**s**. Generally, if the real-time requirements are met, we see a drop in performance and vice versa. So, balancing both the aspects was a big challenge

Finally, the obtained results have shown a mean average precision of 0.87 and rapid detection that reached out to 58 FPS, these parameters satisfy completely the requirements of a Real-time object detector.

Perspectives:

- Future works could consider fire spot localization strategy development which can obtain the fire position in real world coordinates.
- we will look for more effective ways and approaches based on several models to resolve existing false positives.
- we would like to increase the size of the Forest Fire benchmark and make finer grained annotations, e.g., using compute graphic engine to create synthetic data and generate pixel-wise annotations.

- we plan to mount the proposed fire detection system on UAVs for real world forest fire detection by realizing an embedded system comprising a camera and a microcontroller card such as Raspberry Pi or else base of a GPU such as the NVIDIA Jetson card.

References

Bibliography

[1] Barmpoutis, P., Papaioannou, P., Dimitropoulos, K., & Grammalidis, N. (2020). A review on early forest fire detection systems using optical remote sensing. *Sensors*, *20*(22), 6442.

[2] Alkhatib, A. A. (2014). A review on forest fire detection techniques. *International Journal of Distributed Sensor Networks*, *10*(3), 597368.

[3] Kim, B., & Lee, J. (2019). A video-based fire detection using deep learning models. *Applied Sciences*, *9*(14), 2862.

[4] Yuan, C., Liu, Z., & Zhang, Y. (2017). Aerial images-based forest fire detection for firefighting using optical remote sensing techniques and unmanned aerial vehicles. *Journal of Intelligent & Robotic Systems*, *88*(2), 635-654.

[5] Pan, H., Badawi, D., & Cetin, A. E. (2020). Computationally efficient wildfire detection method using a deep convolutional network pruned via Fourier analysis. *Sensors*, *20*(10), 2891.

[6] Center of fire statistics CTIF REPORT World Fire Statistics 2020 N °25.

[7] Celik, T, Demirel, H. Ozkaramanli Celik, T., Demirel, H., Ozkaramanli, H., & Uyguroglu, M. (2007). Fire detection using statistical color model in video sequences. *Journal of Visual Communication and Image Representation*, *18*(2), 176-185.

[8] Chen, T. H., Wu, P. H., & Chiou, Y. C. (2004, October). An early fire-detection method based on image processing. In *2004 International Conference on Image Processing, 2004. ICIP'04.* (Vol. 3, pp. 1707-1710). IEEE.

[9] Wang, T., Shi, L., Yuan, P., Bu, L., & Hou, X. (2017, October). A new fire detection method based on flame color dispersion and similarity in consecutive frames. In *2017 Chinese Automation Congress (CAC)* (pp. 151-156). IEEE.

[10] Borges, P. V. K., & Izquierdo, E. (2010). A probabilistic approach for vision-based fire detection in videos. *IEEE transactions on circuits and systems for video technology*, *20*(5), 721-731.

[11] Foggia, P., Saggese, A., & Vento, M. (2015). Real-time fire detection for video-surveillance applications using a combination of experts based on color, shape, and motion. *IEEE TRANSACTIONS on circuits and systems for video technology*, *25*(9), 1545-1556.

[12] Günay, O., & Çetin, A. E. (2015, September). Real-time dynamic texture recognition using random sampling and dimension reduction. In *2015 IEEE International Conference on Image Processing (ICIP)* (pp. 3087-3091). IEEE.

[13] Zhao, Y., Ma, J., Li, X., & Zhang, J. (2018). Saliency detection and deep learning-based wildfire identification in UAV imagery. *Sensors*, *18*(3), 712.

[14] Frizzi, S., Kaabi, R., Bouchouicha, M., Ginoux, J. M., Moreau, E., & Fnaiech, F. (2016, October). Convolutional neural network for video fire and smoke detection. In *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society* (pp. 877-882). IEEE.

[15] Zhang, Q., Xu, J., Xu, L., & Guo, H. (2016, January). Deep convolutional neural networks for forest fire detection. In *Proceedings of the 2016 international forum on management, education and information technology application. Atlantis Press*.

[16] Muhammad, K., Ahmad, J., Lv, Z., Bellavista, P., Yang, P., & Baik, S. W. (2018). Efficient deep CNN-based fire detection and localization in video surveillance applications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *49*(7), 1419-1434.

[17] Hochreiter, S., & Schmidhuber, J. (1997). LSTM can solve hard long time lag problems. *Advances in neural information processing systems*, 473-479.

[18] Hu, C., Tang, P., Jin, W., He, Z., & Li, W. (2018, July). Real-time fire detection based on deep convolutional long-recurrent networks and optical flow method. In *2018 37th Chinese Control Conference (CCC)* (pp. 9061-9066). IEEE.

[19] Jijitha, R., & Shabin, P. (2019). A Review on Forest Fire Detection. *Research and Applications: Embedded System*, *2*(3).

[20] Meddour-Sahar, O., & Bouisset, C. (2013). Les grands incendies de forêt en Algérie : problèmes humains et politiques publiques dans la gestion des risques. *Méditerranée. Revue géographique des pays méditerranéens/Journal of Mediterranean geography*, (121), 33-40.

[21] O'Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G. V., Krpalkova, L., ... & Walsh, J. (2019, April). Deep learning vs. traditional computer vision. In *Science and Information Conference* (pp. 128-144). Springer, Cham.

[22] Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7263-7271).

[23] Shamsoshoara, A., Afghah, F., Razi, A., Zheng, L., Fulé, P. Z., & Blasch, E. (2021). Aerial Imagery Pile burn detection using Deep Learning: the FLAME dataset. *Computer Networks*, *193*, 108001.

[24] Cazzolato, M. T., Avalhais, L. P., Chino, D. Y., Ramos, J. S., de Souza, J. A., Rodrigues-Jr, J. F., & Traina, A. J. (2017). Fismo: A compilation of datasets from emergency situations for fire and smoke analysis. In *Brazilian Symposium on Databases-SBBD* (pp. 213-223). SBC.

[25] Giuffrida, G., Meoni, G., & Fanucci, L. (2019). A YOLOv2 convolutional neural network-based human–machine interface for the control of assistive robotic manipulators. *Applied Sciences*, *9*(11), 2243.

[26] Wang, Z., Xu, K., Wu, S., Liu, L., Liu, L., & Wang, D. (2020). Sparse-YOLO: Hardware/Software co-design of an FPGA accelerator for YOLOv2. *IEEE Access*, *8*, 116569-116585

Webography

[27] “A Gentle Introduction to Object Recognition with Deep Learning’ at https://machinelearningmastery.com/object-recognition-with-deep-learning/

[28] “Everything You Ever Wanted to Know About Computer Vision”. at

https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer- vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e

[29] “What Is Deep Learning?3 things you need to know” at https://www.mathworks.com/discovery/deep-learning.html

[30] “Convolutional Neural Network 3 things you need to know” at

https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html

[32] “What is Deep Learning and How does it work?” at

https://towardsdatascience.com/what-is-deep-learning-and-how-does-it-work- 2ce44bb692ac

[32] “A Gentle Introduction to Computer Vision” at

https://machinelearningmastery.com/what-is-computer-vision/

[33] “Everything You Wanted to Know About Machine Learning but Were Too Afraid to Ask” at

https://medium.com/swlh/everything-you-wanted-to-know-about-machine-learning-but- were-too-afraid-to-ask-d7d92021038

[34] “Artificial Intelligence” at

https://nptel.ac.in/content/storage2/courses/109101004/downloads/Lecture-19%20&%2020.pdf

[35] The Flame Dataset, December 2020 at

https://ieee-dataport.org/open-access/flame-dataset-aerial-imagery-pile-burn-detection-using-drones-uavs

[36] FiSmo datasets at https://goo.gl/uW7LxW

[37] Gaiasd D-Fire Dataset at https://github.com/gaiasd/DFireDataset

[38] FIRE Dataset Outdoor-fire images and non-fire images for computer vision tasks.2018 at https://www.kaggle.com/phylake1337/fire-dataset

[39] Foggia’s Dataset at http://signal.ee.bilkent.edu.tr/VisiFire/Demo/FireClips/?C=D;O=A

[40] Handpicked Fire Images at https://drive.google.com/drive/?tab=ro

[41] All New Dataset at https://drive.google.com/drive/?tab=ro

[42] “Transform Images and Corresponding Bounding Boxes” at https://www.mathworks.com/help/vision/ref/bboxwarp.html?s_tid=doc_ta

[43] “Loss Fuction Explained” at https://deeplearningdemystified.com/article/fdl-3

[44] “Introduction to Object Detection Model Evaluation” at

https://towardsdatascience.com/introduction-to-object-detection-model-evaluation-3a789220a9bf

[45] “ Evaluating performance of an object detection model” at

https://towardsdatascience.com/evaluating-performance-of-an-object-detection-model-137a349c517b

[46] “Evalution metric” at

https://eurocity-dataset.tudelft.nl/eval/benchmarks/detection