**Fire Detection Using R-CNN**

A project Report submitted

in partial fulfilment of the degree

**BACHELOR OF TECHNOLOGY**

In

**COMPUTER SCIENCE AND ENGINEERING**

**By**

CH.AKHILA 17568T0909

A.LAHARI 17568T0926

G.YAMUNA 17568T0914

E.NIREEKSHANA 185680962L

P.SAI SARADHVI 17568T0936

**Under the Guidance of**

**Mrs.K.Vanisree**

Department of CSE



Submitted to

**Department of Computer Science And Engineering**

**UNIVERSITY COLLEGE OF ENGINEERING & TECHNOLOGY**

**FOR WOMEN**

**KAKATIYA UNIVERSITY CAMPUS,VIDYARANYAPURI**

**WARANGAL-506009,INDIA**

**JULY-2021**

**UNIVERSITY COLLEGE OF ENGINEERING & TECHNOLOGY**

**FOR WOMEN**

**KAKATIYA UNIVERSITY CAMPUS,VIDYARANYAPURI**

**WARANGAL-506009**



**CERTIFICATE**

This is to certify that the Project Report entitled **“FIRE DETECTION USING R-CNN”** is a bonafide work of the students CH.AKHILA (178568T0909), A.LAHARI (17568T0926), G.YAMUNA (17568T0914), E.NIREEKSHANA(18568T0962L), P. SAI SARADHVI (17568T0936) bearing Roll No’s submitted in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** to the University College of Engineering and Technology for Women, Kakatiya University Campus Warangal-506009.It is a record of work carried out during the academic year **2020-2021** under the guidance and supervision.

**Internal Guide/H.O.D External guide**

(Mrs. K.Vanisree)

**Principal**

(Prof.T.Srinivasulu**)**

**DECLARATION**

We declare that this written submission represents our ideas in our own words and where other’s ideas or words have been included, we have adequately cited and referenced the original sources. We declare that the work presented in this project report is original and carried out in the development of computer science and engineering, University College of Engineering and Technology for Women, Kakatiya University Campus. we have not been submitted elsewhere for any graduate in part or in full.

Ch.Akhila 17568T0909

A.Lahari 17568T0926

G.Yamuna 17568T0914

E. Nireekshana 18568T0962L

P. Sai Sharadhvi 17568T0936

**ACKNOWLEDGEMENT**

It plunges us in talking privilege in expressing our heartfelt gratitude to all who helped encourage and foresee successful completion of our project. Ecstasy to work under gregarious guidance of **Mrs. K. Vanisree** to whom we are extremely in debt we are for her valuable and timely suggestions. It was a pleasure for us to receive valuable suggestions and motivating guidance during our project work. We owe our sincere thanks to principal **Prof. T. Srinivasulu** for his consistent cooperation and encouragement. Finally, we would like to thank everybody from the bottom of our hearts for their kind cooperation in helping us to complete our project work in time.

Ch.Akhila 17568T0909

A.Lahari 17568T0926

G.Yamuna 17568T0914

E. Nireekshana 18568T0962L

P. Sai Sharadhvi 17568T0936

# **ABSTRACT**

Fires are one of the main causes of death in the world. Although there are various fire detection systems, most of them did not prove their effectiveness in detecting fires. Due to the diversity of shape and texture of flame, and interference objects that are similar to flame images is a difficult task. This project detects fire flames by studying the fire properties that are color and characteristics of fire by using Computer vision and Image processing. The required environments and tools are python 3.5,tensorflow 1.31.1,OpenCV. For the machine learning process of the system,1000 images that include fire and non-fire scenes are used and 80%of them are used for training and the other 20% for validation. The developed real time fire detector model was tested in real time using lighter as fire source. By using a faster R-CNN(Regions with Convolutional Neural Network features) fire detector model we can get 99% accuracy.

# **CONTENTS**

1. INTRODUCTION 7

2. LITERATURE SURVEY 8

3. EXISTING SYSTEM 9-10

4. PROPOSED SYSTEM 11-12

5. SYSTEM SPECIFICATIONS 13-14

6. SOFTWARE ENVIRONMENT 15-22

7. MODULES 23-28

8. IMPLEMENTATION 29-32

9. SYSTEM DESIGN 33-34

10. DATA FLOW DIAGRAMS 35-36

11. SYSTEM TESTING 37-40

12. SCREEN SHOTS 41-43

13. CONCLUSION 44

14. REFERENCES 45-47

# **INTRODUCTION**

Every year, fire causes enormous damage to human society. Forest fires take the lives of many firefighters, causing serious impacts on the local environment and ecology. In addition, in densely populated areas such as factories and residential areas, fire disasters are more frequent due to the large amount of combustibles, and the widely distributed kitchen areas, directly causing considerable damage to properties and casualties[1]. With rapid spread of urbanization in the world, both the number of permanent residents in cities and the population density are increasing.In march and April of 2019, there were many large scale fire accidents around the world, such as forest fires in Liangshan (china), The Notre Dame fire in France , forest fires in italy, and the grassland fire in Russia , which caused great damage to people’s lives and property . In 2020, fire at sanitizer manufacturing company , Ahmedabad . In 2021,fire at Indonesian oil refinery(march 29);fire at shopping complex(march 23) are some instances. In recent years, generic multi-class object detection methods have been used for flame detection [4], [5]. Each year, an area of vegetation of 10,000 km2 is affected by fire disasters in Europe. The statistic for fire damage is about 100,000 km2 in Russia and North America.Other examples of fire disasters include (1) the disaster of Arizona (USA, June 2013) which ruined 100 houses and killed 19 firefighters, and (2) the forest fire of California (August 2013) which burned an area of 1042 km2 and damaged around 111 structures, incurring a firefighting cost of $127.35 million [6]. Considering these examples of damage, early detection of fire is of paramount interest to disaster management systems, so as to avoid such disasters. In this context, researchers have explored different approaches to fire detection including conventional fire alerting systems and visual sensors based systems. The systems belonging to the first category are based on ion or optical sensors, needing close proximity to the fire,and thus failing to provide additional information such as the fire size, location, and degree of burning. With the development of computer vision, computer vision-based flame detection technology has been extensively studied. Detecting the flame using vision sensors is faster than the traditional fixed smoke sensor, and the flame can be detected at an early stage, providing precious time for putting the fire out. At present, computer vision-based flame detection methods (hereinafter referred to as flame detection) are mainly divided into two categories: features by artificial design and features by convolutional neural network (CNN) extraction.

# **LITERATURE SURVEY**

Flame detection is a special single-class object detection problem, so these methods have room for improvement in flame detection. To this end, it is necessary to improve the generic object detection method to make it more suitable for flame detection, thereby obtaining better performance. Fire disasters mainly occur due to human error or the failure of a system, causing economic as well as ecological damage along with endangering human lives [7]. According to [8], wildfire disasters alone in the year 2015 resulted in 494,000 victims and caused damage worth US$ 3.1 billion. Firefighting is an extremely hard and dangerous task. As we know, in case of any fire accidents there will be much damage in an area i.e., in case of property , or large area of trees(forests), and loss of human lives too. And it’s not so easy to fight against fire, after it got spread over a large area. To overcome this , if we could detect the fire before it spreads ,we could control it easier .This system helps in detecting fire at the initial point of attack , so we can save the environment(nature , lives, property) to a great extent. The main objective of this system is to stop causing damage to nature and lives(i.e., environment).

# **EXISTING SYSTEM**

Töreyin et al. [9] trained a Gaussian mixture model as the color model of the flame, which is combined with motion detection algorithms to extract regions, and then applied temporal and spatial wavelet analysis on these regions to determine the appearance of flame in each region. Foggia et al. [10] adopted three evaluation modules of color, morphological change and motion to discriminate each candidate region and designed a voting strategy with weight to make a comprehensive decision. These two methods can only be used on fixed cameras and cannot be applied to self moving visual systems. Chino et al. [11] constructed the BoWFire dataset, and proposed a flame detection algorithm framework by combining color features with superpixel texture discrimination. Barmpoutis et al. [4] trained a Faster R-CNN model to obtain the candidate flame regions in the image and then used the vector of indigenous aggregated descriptors (VLAD) to determine whether each candidate region is a flame region. However, the training and test data sets used are small in scale. Zhang et al. [5] presented two methods to add simulated smoke to images without smoke, thereby enhancing the original dataset and solving the problem of lack of training data.

**DISADVANTAGES**

* There are systems based on smoke sensors in detecting fires , detection systems based upon temperature , based on color models.
* Smoke detecting systems are restricted to the existence of a ceiling or a wall. In open large areas, it may also detect smoke from vehicles that leads to false detection.
* In the case of systems based on color models, it may also detect moving objects of similar color as of fire and result in fake detection (alarm) .
* When an alarm is triggered, a fire may already be too strong to control, defeating the purpose of early warning.
* There may be false alarms i.e., when non-fire particle concentration reaches the alarm concentration, it will automatically sound the alarm.
* All these systems are limited to indoor usages.
* Most of the alarms can only be functional in a closed environment , which is ineffective for a wide space, such as outdoors or public places.
* To prevent fires and control their rapid growth , it is necessary to establish a monitoring system that can detect early fires.

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# **PROPOSED SYSTEM**

In the proposed system, in order to overcome the drawbacks in existing systems, sensor and color based systems are replaced with computer vision based . This project uses computer vision and image processing techniques. Development in digital camera , image and video processing also helps in detecting fire that is far from surveillance cameras. It detects fire flames based on studying the fire properties (color and characteristics of fire ).This developed real time fire detector model gives accuracy of 99% .

The required environments and tools are python 3.5,tensorflow 1.31.1,OpenCV. For machine learning

process of the system,1000 images that include fire and non-fire scenes are used and 80% of them are

used for training and the other 20% for validation. The developed real time fire detector model was tested in real time using a lighter as a fire source. By using a faster R-CNN(Regions with Convolutional Neural Network features) fire detector model we can get 99% accuracy.

**ADVANTAGES**

Visual sensors based fire detection systems are motivated by several encouraging advantages

including:

* Low cost due to the existing setup of installed cameras for surveillance.
* Monitoring of larger regions.
* Comparatively fast response time due to the elimination of waiting time for heat diffusion.
* Fire confirmation without visiting the fire location.
* Flexibility for the detection of smoke and flames through adjustment of certain parameters.
* The availability of fire details such as size, location, and degree of burning.
* Due to these characteristics, they have attracted the attention of many researchers and as a result, many fire detection methods [12–17] have been investigated based on numerous visual features, achieving good performance.

# **SYSTEM** **REQUIREMENTS SPECIFICATION**

The aim of this study is to develop a real time fire detector using Faster R-CNN (Faster region based convolutional neural network). For the machine learning process of the system; 1,000 images (including fire and non-fire scenes) 80 and 20% for training and validation, respectively were used.The machine learning process was conducted using a system with the specifications of NVidia GeForce GTX 1070 Ti with 17 GB onboard memory. The required environments and tools (Python 3.5, Tensorflow 1.13.1, OpenCV, CUDA-cuDNN toolkits) . The fire scenes on the images were labeled as fire and non-fire using LabelIng software. The metrics of the training process were obtained from the Tensorboard. The total loss value decreased from 2 to 0.02 with the steps of 40,000 at training. As the loss function was lower than the level of 0.05, the inference graph was frozen and exported to detect the fire source. The total loss value decreased from 2 to 0.02 with the steps of 40,000 at training. As the loss function was lower than the level of 0.05, the inference graph was frozen and exported to detect the fire source. The developed real time fire detector model was tested in real time using lighter as fire source. In the test results; the 99% of accuracy was obtained using the developed FasterR-CNN fire detector model. Firefighting is an extremely hard and dangerous task. After starting, it is almost impossible to control and it is extremely hard to recover the damaged area and lost lives. Therefore, the most efficient way for firefighting is to detect the fire source before it spreads and reaches the point of no return. For this reason,some early fire detection systems have been developed . Luo & Su developed an intelligent security system for buildings . The color based systems generally use the models such as RGB (Red-Green and Blue) and HSI (Hue, Saturation and Intensity). The detection accuracy of these systems are too low since the color of the flames can change depending upon environmental conditions and burning material . Machine learning based systems detect the fires using some developed algorithms such as Bayesian network, SVM (Support vector machine) and CNN (Convolutional neural networks). While former systems have high errors, the latter systems can significantly reduce the false detection rate and provide the best results. The multiple sensors and fused sensory data were used in the system in order to both detect the fire and generate a reliable fire detection signal. Khoon et al. developed an Autonomous firefighting mobile platform. It had capability to patrol and monitor the prescribed area and to search for the fire occurrence with flame sensors. Chang et al. designed and manufactured a FSR (Fire searching robot) using task oriented design (TOD) methodology. Kim et al. composed a portable fire evacuation guide system that can monitor indoor fires. Roberto et al. projected a multi sensor data fusion technique for fire detection. The detection system was based upon temperature, luminosity and flame measurements.

Although the proposed sensor based fire detection systems can detect the fire sources, they are generally limited to indoor usages. They also have limitations according to location and type of sensors used in the system . Currently; the fire detection methods have been replaced from sensor based to computer vision based due to research and developments in digital camera, image and video processing techniques. Image based fire detection systems can be categorized as computer vision, color model and machine learning. In the computer vision based systems; the fire source is detected using both color and characteristics of the fire extracted from the motion of flames

Software Requirement Specifications:

• Nvidia GeForce GTX 1070 Ti with 17 Gb onboard memory. GeForce GTX 1070 is a powerful graphics card that gives you the fast,smooth,quiet gaming you are looking for in all your favourite titles. Advantage of Nvidia’s tens of thousands of hours of testing to optimize your result in one click.

• Required environment and tools: - Python3.5 - tensorflow 1.13.1 - OpenCV - CUDA -cuDNN toolkit

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# **SOFTWARE ENVIRONMENT**

Python :

Python consistently ranks as one of the most popular programming languages.

Python programming language is a high-level language that can be characterized by the following buzzwords:

⦁ Simple

⦁ Architecture neutral

⦁ Object-oriented

⦁ Structured

⦁ Multi-paradigm programming

⦁ Interpreted

⦁ Dynamic

⦁ High performance

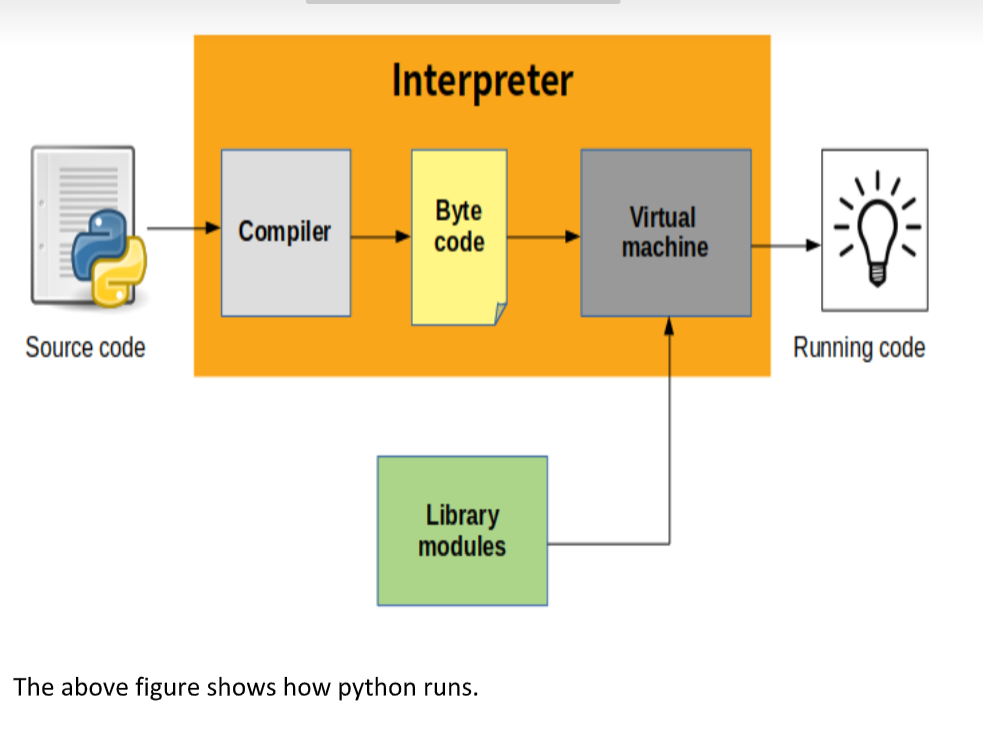
Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

⦁ Python is dynamically-typed and garbage-collected.

⦁ It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming.

⦁ Python is often described as a "batteries included" language due to its comprehensive standard library

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0.[32] Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible and much Python 2 code does not run unmodified on Python 3. Python 2 was discontinued with version 2.7.18 in 2020.



Python Features:

⦁ functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)).

⦁ Many other paradigms are supported via extensions, including design by contract and logic programming.

⦁ Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management.

⦁ It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

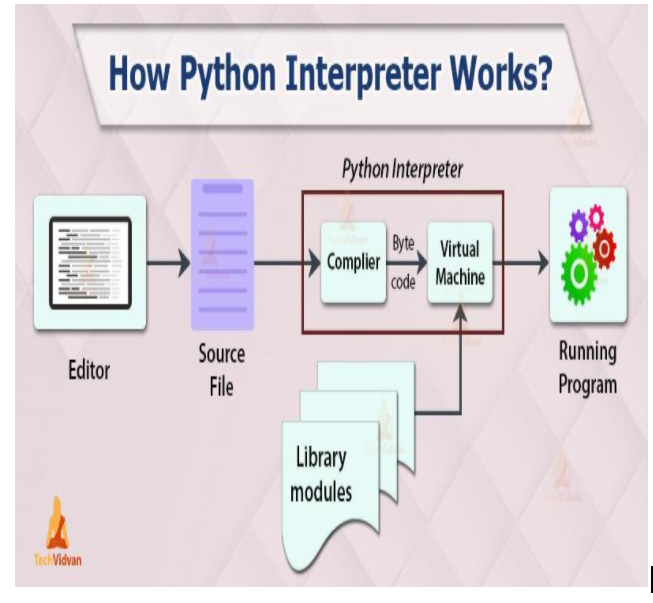
Python's design offers some support for functional programming in the Lisp tradition.

It has

⦁ filter,map and reduce functions;

⦁ list comprehensions, dictionaries, sets, and generator expressions.

The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.



Syntax:

The syntax of the Python programming language is the set of rules that defines how a Python program will be written and interpreted (by both the runtime system and by human readers). The Python language has many similarities to Perl, C, and Java. However, there are some definite differences between the languages.

Python is meant to be an easily readable language.

Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are allowed but are rarely, if ever, used. It has fewer syntactic exceptions and special cases than C or Pascal.

Indentation:

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block.

Thus, the program's visual structure accurately represents the program's semantic structure.This feature is sometimes termed the off-side rule, which some other languages share, but in most languages indentation doesn't have any semantic meaning. The recommended indent size is four spaces.

In so-called "free-format" languages that use the block structure derived from ALGOL-blocks of code are set off with braces ({ }) or keywords. In most coding conventions for these languages, programmers conventionally indent the code within a block, to visually set it apart from the surrounding code.

A recursive function named "foo",which is passed a single parameter, 'x', and if the parameter is '0' will call a different function named "bar" and otherwise will call "baz", passing 'x', and also call itself recursively, passing "x-1"as the parameter could be implemented like this in Python:

def foo(x):

if x == 0:

bar()

else:

baz(x)

foo(x - 1)

and could be written like this in C with K&R indent style:

void foo(int x)

{

if (x == 0) {

bar();

} else {

baz(x);

foo(x - 1);

}

}

Statements and control flow :

Python's statements include (among others):

⦁ The assignment statement, using a single equals sign '='.

⦁ The "if"statement, which conditionally executes a block of code, along with 'else' and 'elif' (a contraction of else-if).

⦁ The "for" statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.

⦁ The "while" statement, which executes a block of code as long as its condition is true.

⦁ The "try" statement, which allows exceptions raised in its attached code block to be caught and handled by 'except' clauses; it also ensures that clean-up code in a 'finally' block will always be run regardless of how the block exits.

⦁ The "raise" statement, used to raise a specified exception or re-raise a caught exception.

⦁ The "class" statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.

⦁ The "def" statement, which defines a function or method.

⦁ The "with" statement, which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing resource-acquisition-is-initialization (RAII)-like behavior and replaces a common try/finally idiom.[79]

⦁ The "break" statement exits from a loop.

⦁ The "continue" statement, skips this iteration and continues with the next item.

⦁ The "del" statement removes a variable, which means the reference from the name to the value is deleted and trying to use that variable will cause an error. A deleted variable can be reassigned.

⦁ The "pass" statement, which serves as a NOP. It is syntactically needed to create an empty code block.

⦁ The "assert" statement, used during debugging to check for conditions that should apply.

⦁ The "yield" statement, which returns a value from a generator function and 'yield' is also an operator. This form is used to implement coroutines.

⦁ The "return" statement, used to return a value from a function.

⦁ The "import" statement, which is used to import modules whose functions or variables can be used in the current program.

Expressions:

Some Python expressions are similar to those found in languages such as C and Java, while some are not:

⦁ Addition, subtraction, and multiplication are the same, but the behavior of division differs. There are two types of divisions in Python. They are floor division (or integer division) // and floating-point/division.Python also uses the \*\* operator for exponentiation.

⦁ From Python 3.5, the new '@'infix operator was introduced. It is intended to be used by libraries such as NumPy for matrix multiplication.

⦁ From Python 3.8, the syntax ':=', called the 'walrus operator' was introduced. It assigns values to variables as part of a larger expression.

⦁ In Python, '=='compares by value, versus Java, which compares numerics by value and objects by reference.(Value comparisons in Java on objects can be performed with the equals() method.) Python's operator may be used to compare object identities (comparison by reference). In Python, comparisons may be chained, for example a <= b <= c.

⦁ Python uses the words and, or, not for its boolean operators rather than the symbolic && ||, ! used in Java and C.

**Methods:**

Methods on objects are functions attached to the object's class; the syntax 'instance.method(argument)' is, for normal methods and functions, 'syntactic sugar for Class.method(instance, argument)'.

Python methods have an explicit 'self' parameter to access instance data, in contrast to the implicit 'self' (or 'this') in some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby).Apart from this Python also provides methods, sometimes called dunder methods due to their names beginning and ending with double-underscores, to extend the functionality of custom class to support native functions such as print, length, comparison, support for arithmetic operations, type conversion, and many more.

**Libraries:**

Python's large standard library, commonly cited as one of its greatest strengths,provides tools suited to many tasks. For Internet-facing applications, many standard formats and protocols such as MIME and HTTP are supported. It includes modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, arithmetic with arbitrary-precision decimals, manipulating regular expressions, and unit testing.

Some parts of the standard library are covered by specifications (for example, the Web Server Gateway Interface (WSGI) implementation wsgiref follows PEP 333), but most modules are not. They are specified by their code, internal documentation, and test suites. However, because most of the standard library is cross-platform Python code, only a few modules need altering or rewriting for variant implementations.

As of March 2021, the Python Package Index (PyPI), the official repository for third-party Python software, contains over 290,000 packages with a wide range of functionality, including:

⦁ Automation

⦁ Data analytics

⦁ Databases

⦁ Documentation

⦁ Graphical user interfaces

⦁ Image processing

⦁ Machine learning

⦁ Mobile App

⦁ Multimedia

⦁ Computer Networking

⦁ Scientific computing

⦁ System administration

⦁ Test frameworks

⦁ Text processing

⦁ Web frameworks

⦁ Web scraping



**Uses:**

Python can serve as a scripting language for web applications, e.g., via mod wsgi for the Apache web server.[182]

With Web Server Gateway Interface, a standard API has evolved to facilitate these applications. Web frameworks like Django, Pylons, Pyramid, TurboGears, web2py, Tornado, Flask, Bottle and Zope support developers in the design and maintenance of complex applications. Pyjs and IronPython can be used to develop the client-side of Ajax-based applications.

SQLAlchemy can be used as a data mapper to a relational database. Twisted is a framework to program communications between computers, and is used (for example) by Dropbox.

**Languages influenced by python:**

Python's design and philosophy have influenced many other programming languages:

⦁ 'Boo' uses indentation, a similar syntax, and a similar object model.

⦁ 'Cobra' uses indentation and a similar syntax, and its Acknowledgements document lists Python first among languages that influenced it.

⦁ 'CoffeeScript', a programming language that cross-compiles to JavaScript, has Python-inspired syntax.

⦁ 'ECMAScript/JavaScript' borrowed iterators and generators from Python.

⦁ 'GDScript', a scripting language very similar to Python, built-in to the Godot game engine.

⦁ 'Go',is designed for the "speed of working in a dynamic language like Python"and shares the same syntax for slicing arrays.

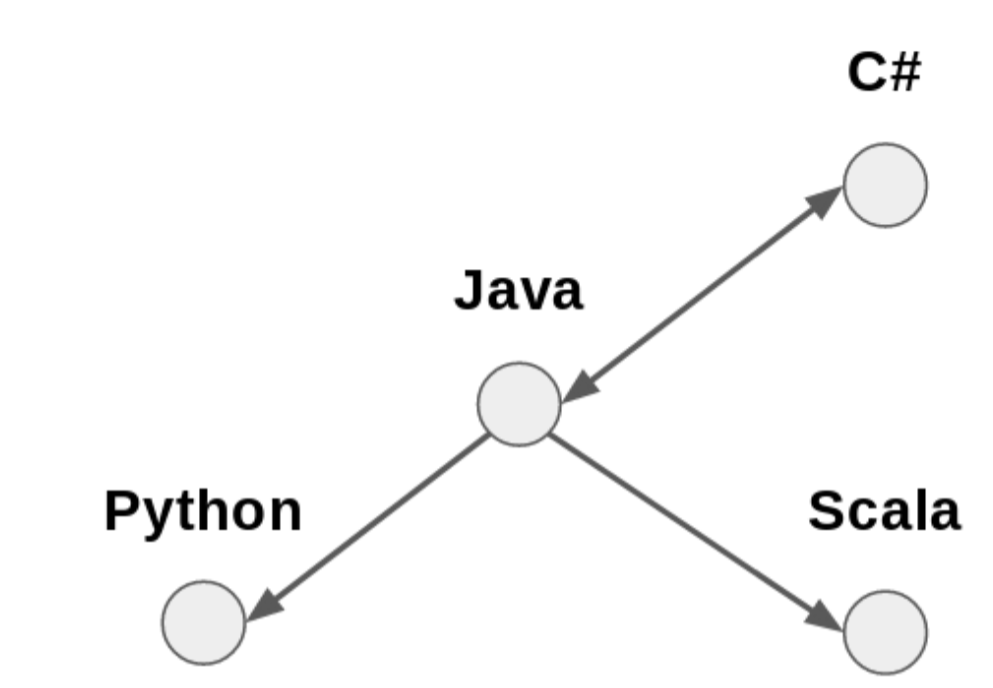
⦁ 'Groovy' was motivated by the desire to bring the Python design philosophy to Java.

⦁ 'Julia' was designed to be "as usable for general programming as Python".

⦁ 'Nim' uses indentation and similar syntax.

⦁ Ruby's creator, Yukihiro Matsumoto, has said: "I wanted a scripting language that was more powerful than Perl, and more object-oriented than Python. That's why I decided to design my own language."

⦁ Swift, a programming language developed by Apple, has some Python-inspired syntax.



The above figure visualises the programming language influence graph.

# **MODULES**

This section mainly introduces the proposed flame detection method. First, the Faster R-CNN model for object detection is introduced in part A. Then, the color-guided anchoring strategy and the global information-guided flame detection method are reported in part B and part C, respectively.

**A.FASTER R-CNN :**

Faster R-CNN is a general-purpose object detection method based on R-CNN and Fast R-CNN. The main difference between Faster R-CNN and the two methods is the method of candidate region generation. Faster R-CNN obtained the initial candidate box using RPN(Region Proposal Network) instead of the time-consuming Selective Search algorithm. Faster R-CNN has achieved good results in object detection tasks on Microsoft coco and Pascal VOC .Faster R-CNN generates detection boxes based on anchors, which are the initial set of candidate boxes generated during the object detection process. Combining the m scales with n aspect ratios, which are determined artificially, m × n anchors with fixed shapes can be obtained at each anchor point. An anchor point and all anchor candidate boxes generated at that point. Here, the red point is an anchor point, and the green boxes are the anchors. The anchor points are distributed over the entire image at regular intervals, so anchors can cover the entire image. Anchors generated in each anchor point have the same shape. The structure of the Faster R-CNN model is shown in Fig. 2. The model can be divided into three parts, namely, feature extractor, RPN and RCNN head. The feature extractor is mainly composed of a series of convolutional layers. The input image passes through the feature extractor to obtain a series of feature maps of the image. These feature maps are then sent to the RPN, which is an object detector in the form of a sliding window. The main structure of RPN consists of three convolutional layers with the input of the feature map. The feature map is mapped to a lower space through a 3 × 3 convolutional layer and then fed into two parallel convolution layers with 1 × 1 kernels. At each pixel of the feature map, the RPN generates 9 anchors. Each anchor has an objectness score and four offsets generated from the two parallel convolution layers. The four offsets are t r x , t r y , t r w , and t r h , where t r x and t r y represent the offset of the anchor center position in the X and Y directions of the image, respectively, and t r w and t r h represent the change in the anchor width and length, respectively. After applying the offsets to their corresponding anchors, we obtain a series of candidate regions, which are the output of the RPN. The candidate regions are then sent to non-maximum suppression, where the region that overlaps highly with the region with the highest objectness score is filtered out. After the non- maximum suppression, each region is fed to the ROI pooling layer to obtain a fixed shape feature map corresponding to each region. Finally, the linear layer of the R-CNN Head outputs the classification score of each region and the further offsets t h x , t h y , t h w , and t h h . The physical meaning of the offset from R-CNN Head is exactly the same as the offset from RPN.

**B. COLOR-GUIDED ANCHORING STRATEGY :**

In the Faster R-CNN, the anchor points are distributed densely, and anchors are generated at each pixel of the feature map. This design can make the distribution of the anchor points as wide as possible in an image and further guarantee that each area is covered by an appropriately sized anchor. In other words, the dense anchoring strategy can be used to ensure the universality of the method. However, in the flame detection task, there are only two types of image regions: fire or background. The dense anchoring strategy causes the anchors to be generated in many image areas without flames. Considering the distinct characters of flames in image, this allows the detection of Faster R-CNN to be guided. Inspired by the guided anchoring in [27], we propose a sparse anchoring strategy. The image characteristics of flames are used to guide the anchor generation, taking the place of the original dense anchoring strategy in original Faster R-CNN. Using this sparse anchoring strategy, the efficiency and accuracy of flame detection can be improved. Among the flame image features, the color features are easier to obtain and are also widely used in flame detection. To enable the generation of the anchor at the regions that are similar in color to flames and ensure the anchor boxes will not miss the flame regions as much as possible, we adopt a relatively loose color model as follows:

M (x, y) = 1, fR(x, y) > fG(x, y) > fB(x, y)&& fR(x, y) > TR 0, otherwise (1)

where M (x, y) represents the generated flame color mask; fR, fG, and fB represent the image values of the R, G, and B channels of the image, respectively; and TR represents the threshold set on the R channel. This color model can cover the color of the flames in general situations, regardless of the color (blue, etc.) produced by the burning of special materials. With this color model, we can obtain a color mask that contains flames and all areas of the image that are similar in color to flames. Thus, the original dense anchoring strategy

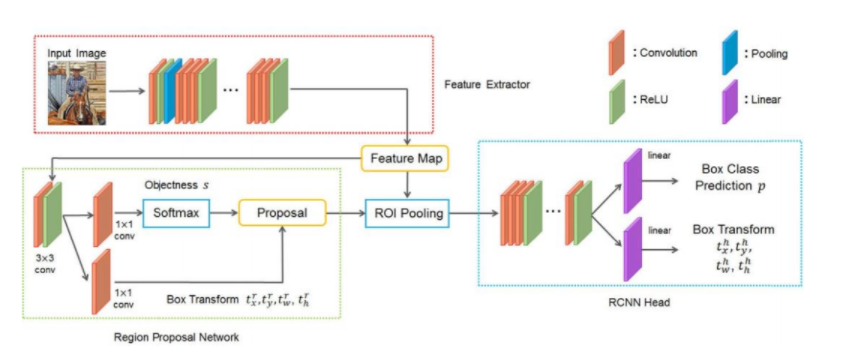


FIGURE 2. The model structure of the Faster R-CNN.

in Faster R-CNN is changed to a color-guided anchoring strategy. Specifically, the anchor point is only set in the region of interest in the mask, so the other regions of the mask do not generate any anchors. Thus, the number of anchors initially generated can be greatly reduced, and the anchors are distributed around objects that are similar in color to flames. This feature also increases the average quality of the anchors. Fig. 3 shows the anchor positions of the color guided anchoring strategy, where the red points are anchor points. To achieve the anchoring strategy described above, we modify the RPN, which is the core of Faster R-CNN. We call it Masked RPN, which accepts an additional mask consistent with the shape of the input feature map. The information of the mask can be combined with the feature map such that the anchor is only generated in the region of interest. The overall structure of the model is shown in Fig. 4. The ResNet101 is selected as the backbone network. The image is input into the feature extractor to obtain the feature map, and it is passed through the color model to obtain the flame color mask. The color mask is scaled to the same shape of the feature map, and then the feature map and the color mask are input to the Masked RPN to obtain the filtered and shifted candidate boxes. Finally, each candidate box is sent into the RCNN Head to obtain the final detection results. Fig. 4 shows the specific structure of the Masked RPN. On the basis of the original RPN, each convolution layer is replaced by the masked convolution layer. In masked convolution, only the region of interest of the image is convoluted, and other regions are simply set to zero, thereby reducing part of the matrix point multiplication. Using masked convolution to implement the color-guided anchoring strategy, the efficiency of model calculations can be improved.

**C. GLOBAL INFORMATION GUIDED FLAME DETECTION**

The color-guided anchoring strategy can increase efficiency and recall of the Faster R-CNN for flame detection, but the strategy cannot solve the problem of high false alarm rate of Faster R-CNN. The Faster R-CNN can only detect the



FIGURE 3. Anchor generated by color-guided anchoring strategy: (a) original image (left); (b) flame color mask (middle); (c) anchor generation locations, marked by red dots (right).

flame information in the candidate box. Thus, the global information of the image cannot be obtained, and the false alarm rate will be relatively high for flame detection for some challenging images (e.g., images such as sunsets). As shown in Fig. 5, there are regions in these images that are very similar to flames, so that the image features within these regions will also be highly similar to flames. When such a region appears in image, the Faster R-CNN is very likely to generate a false alarm. Moreover, it is mentioned that the image block classifier is trained only by local image patterns, so the global information cannot be understood by this classifier. Therefore, we introduce global information of the image into the flame detection method. The features of the entire image can be used for general CNNs that perform classification tasks, and a variety of flame images and non-flame images can be introduced for network training. Thus, the false alarm rate (also called false positives) of a general CNN can reach a considerably low value compared to the Faster R-CNN, and the overall accuracy of such CNN can be trained to a fairly high value. Therefore, we design a general CNN to generate global information of the image and use this information to guide the flame detection progress. Based on the color-guided anchoring strategy, a global information network (GIN) is connected in parallel with the Faster R-CNN model to guide the flame detection process

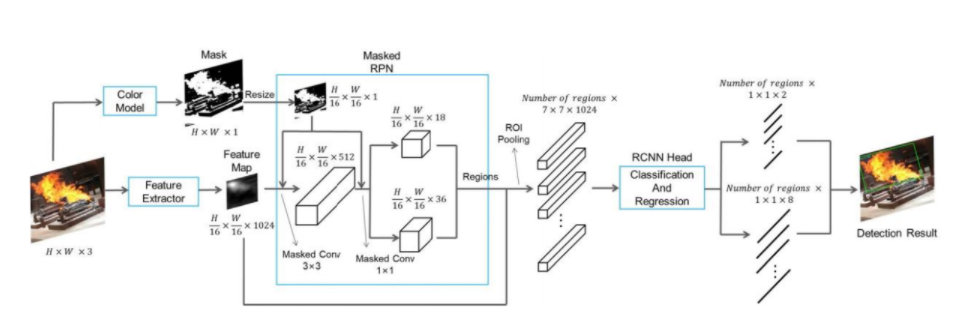


FIGURE 4. Model structure of the color-guided faster R-CNN and the masked RPN.

Considering that Faster R-CNN is a relatively large network, the GIN as a subnet does not have to be very fast, so the ResNet101 is chosen as the GIN. The trained GIN is connected outside the Faster R-CNN in a parallel form. The global information obtained from the GIN and the detection results generated by the Faster R-CNN are comprehensively analyzed to obtain the final detection result. The structural diagram of the global information-guided model is given in Fig. 6. The input image is classified by the parallel GIN to generate a classification result. This classification result together with the detection result generated by the model introduced in section III part B are input into a decision strategy. The strategy is fairly simple. Since we aim at reducing false positives, we can absolutely perform flame detection guided by GIN. When the GIN identifies a flame in the image, the detection result of the Faster R-CNN is retained; otherwise, simply discard all the Faster R-CNN detection results. In fact, the introduction of the GIN will not significantly impact the detection speed of the model because the GIN and the Faster R-CNN are structurally independent, the GIN can be placed in another thread juxtaposed with the main thread, thereby reducing the impact of the additional parallel network on the flame detection process.

**D. TRAINING**

According to the training idea, we design a training strategy for the Faster R-CNN. In each training image, we select 128 sample regions, including 64 positive sample regions and 64 negative sample regions. If the number of positive sample regions is less than 64, the number of the negative sample regions will be increased to maintain a total number of 128. In addition, instead of using the four- step alternating training method in, we adopt approximate joint training such that the training is performed in an end-to-end manner, which is faster and more convenient. We train our Faster R-CNN for 15 epochs. The learning rate is set to 0.001 initially, and it is multiplied by 0.1 every 5 epochs. The batch size for training the Faster R-CNN is 1. The main difficulty training our model comes from the masked RPN



FIGURE 5. Image containing region highly similar to flames.

more specifically, the error back propagation problem of the masked convolution layer. To solve this problem, we design the following scheme. While training, the masked RPN layers are replaced with the normal convolutional layer. For the region output by masked RPN, all the regions outside the mask’s region of interest are filtered out.For the loss calculation of the Faster R-CNN, there are four sources of errors, namely, the classification error and positioning error of the RPN and the R-CNN Head. Cross entropy loss is used both for the calculation of the classification error of the region output by the RPN, and the classification error of the region output by the RCNN Head. The smooth L1 loss is used for calculating the regression error of each region. The loss function of the RPN can be defined as follows: Lr = l r cls + l r reg = X i f r ce(si,s ∗ i ) + X i f r sL1 (t r i , t r∗ i ), (2) where l r cls is the classification loss defined by the cross entropy loss function f r ce, and l r reg is the regression loss defined by the smooth L1 loss function f r sL1 . Here, si is the predicted probability score of the ith region of the RPN output, indicating whether the candidate box contains object; s ∗ i is the ground-truth label of the labeled detection box corresponding to the region; and t r i and t r∗ i are the offset of the ith candidate box of the RPN output and the ground-truth offset, respectively.



TABLE 1. Representative examples from different datasets

The loss function of the R-CNN Head can be defined as follows:

Lh = l h cls + l h reg = X i f h ce(pi, p ∗ i ) + X i f h sL1 (t h i , t h∗ i ) (3) where pi is the predicted class of the ith candidate box of the R-CNN Head output, p ∗ i is the ground truth classification label, t h i is the offset predicted by the R-CNN Head, and t h∗ i is the offsets of ground-truth box corresponding to the candidate. Finally, the loss of the Faster R-CNN is the sum of the loss of RPN and the loss of R-CNN Head. In addition, the training of the GIN can be performed separately. We refer to the general training method of CNNs using binary cross entropy loss for the network training. The GIN is first trained with a learning rate of 0.001 for 10 epochs, then the learning rate is set to 0.0001 for another 10 epochs. The batch size during training the GIN is 4.

# **IMPLEMENTATION**

# 

There is a full-frame binary detection (FireNet, InceptionV1-OnFire, InceptionV3-OnFire, InceptionV4-OnFire) architectures determine whether an image frame contains fire globally, whereas the superpixel based approaches (SP-InceptionV1-OnFire,SP-InceptionV3-OnFire, SP-InceptionV4-OnFire) breaks down the frame into segments and performs classification on each superpixel segment to provide in-frame localization.

To test the supplied code and pre-trained models do:

$ python firenet.py models/test.mp4

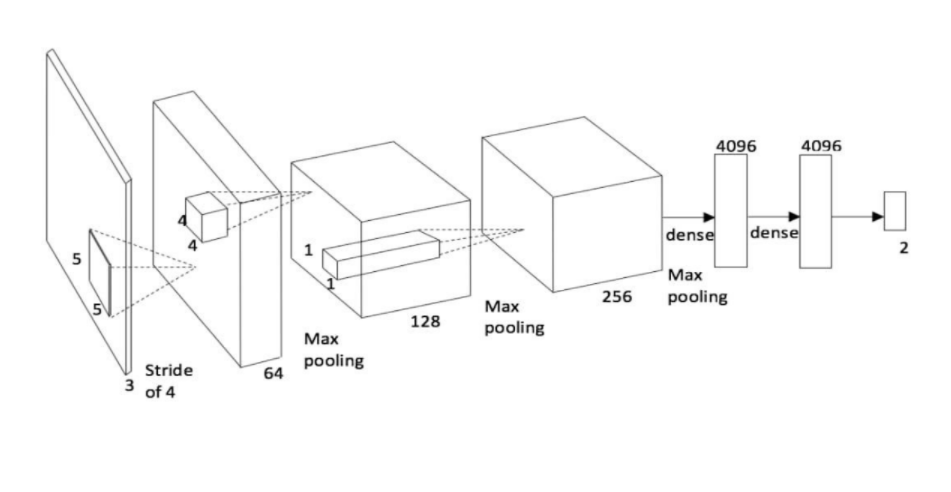
$ python inceptionVxOnFire.py -m 1 models/test.mp4

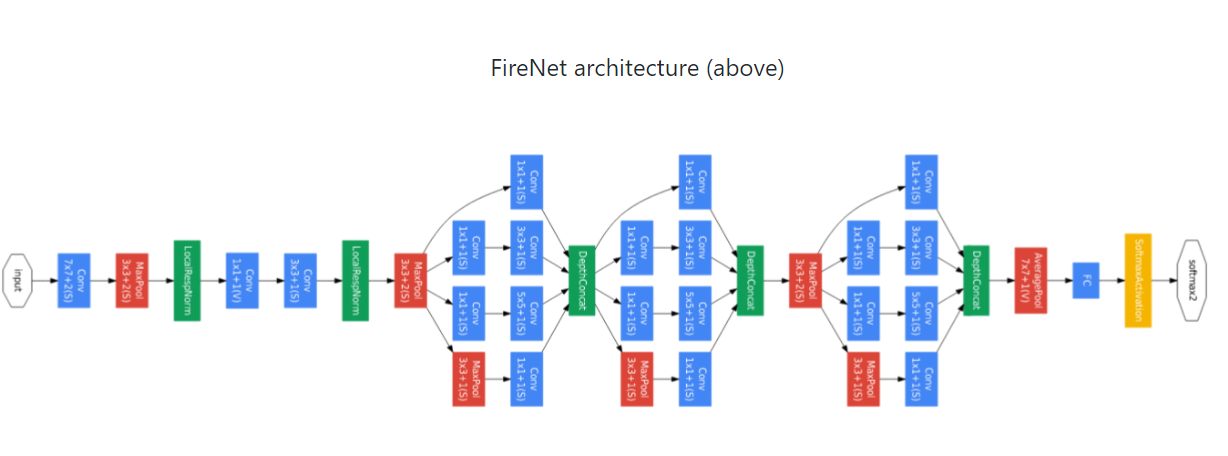
$ python superpixel-inceptionVxOnFire.py -m 1 models/test.mp4

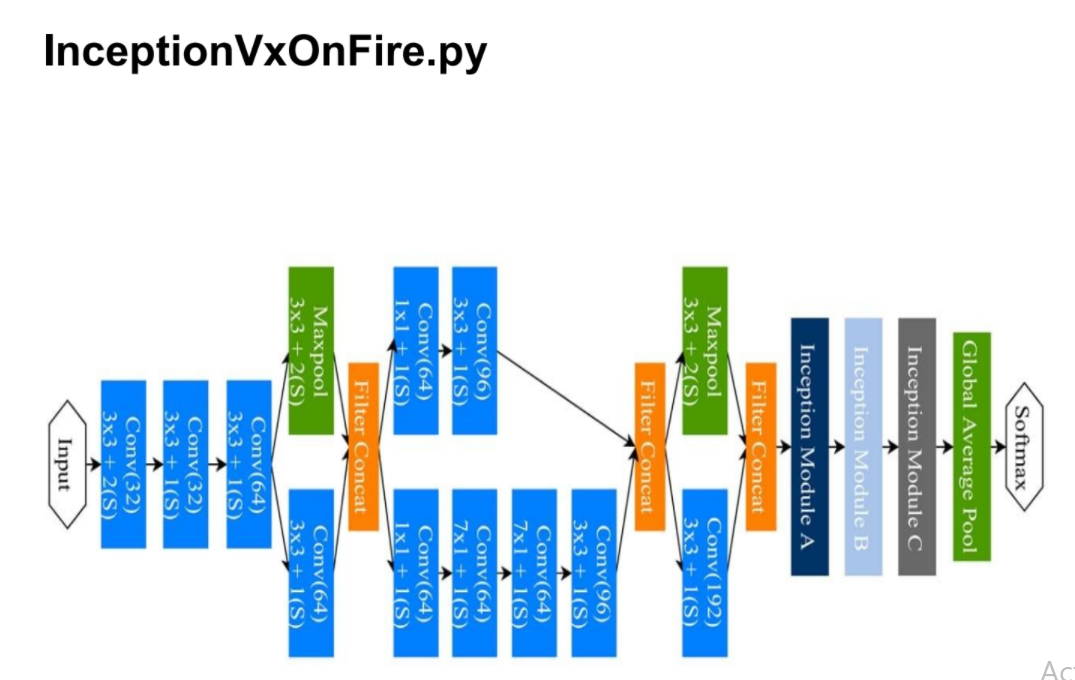
Code:

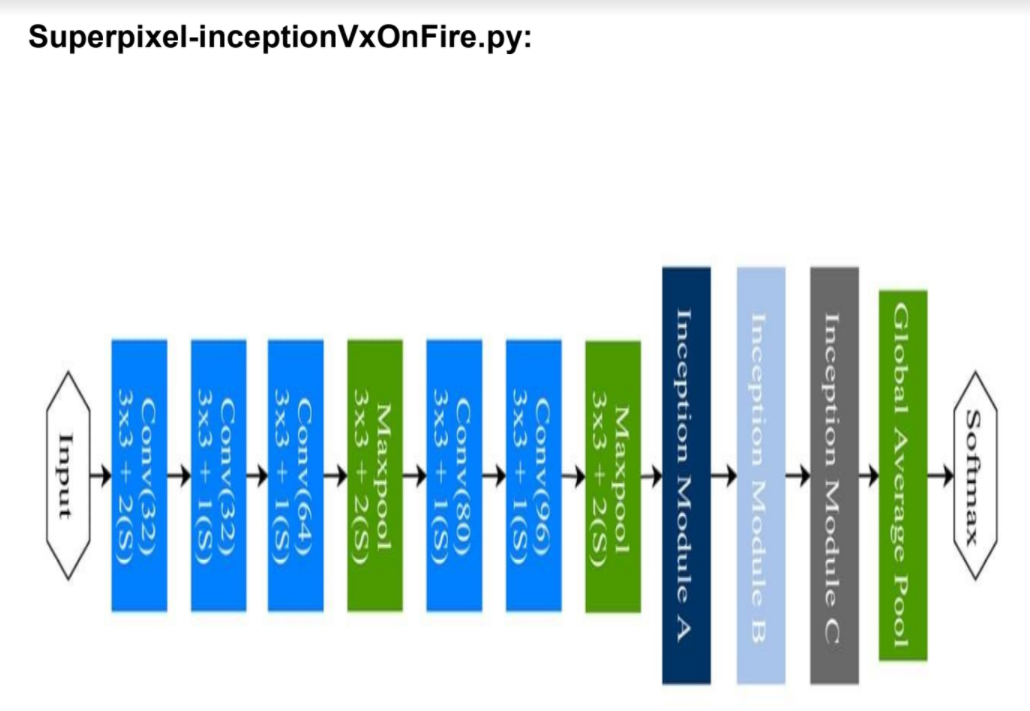
|  | import cv2 |
| --- | --- |
|  | import os |
|  | import sys |
|  | import math |
|  | ################################################################################ |
|  | import tflearn |
|  | from tflearn.layers.core import \* |
|  | from tflearn.layers.conv import \* |
|  | from tflearn.layers.normalization import \* |
|  | from tflearn.layers.estimator import regression |
|  | ################################################################################ |
|  | def construct\_firenet (x,y, training=False): |
|  | network = tflearn.input\_data(shape=[None, y, x, 3], dtype=tf.float32) |
|  | network = conv\_2d(network, 64, 5, strides=4, activation='relu') |
|  | network = max\_pool\_2d(network, 3, strides=2) |
|  | network = local\_response\_normalization(network) |
|  | network = conv\_2d(network, 128, 4, activation='relu') |
|  | network = max\_pool\_2d(network, 3, strides=2) |
|  | network = local\_response\_normalization(network) |
|  | network = conv\_2d(network, 256, 1, activation='relu') |
|  | network = max\_pool\_2d(network, 3, strides=2) |
|  | network = local\_response\_normalization(network) |
|  | network = fully\_connected(network, 4096, activation='tanh') |
|  | if(training): |
|  | network = dropout(network, 0.5) |
|  | network = fully\_connected(network, 4096, activation='tanh') |
|  | if(training): |
|  | network = dropout(network, 0.5) |
|  | network = fully\_connected(network, 2, activation='softmax') |
|  | # if training then add training hyperparameters |
|  | if(training): |
|  | loss='categorical\_crossentropy', |
|  | learning\_rate=0.001) |
|  | # construct final model |
|  | model = tflearn.DNN(network, checkpoint\_path='firenet', |
|  | return model |
|  | ################################################################################ |
|  | if \_\_name\_\_ == '\_\_main\_\_': |
|  | ################################################################################ |
|  | # construct and display model |
|  | model = construct\_firenet (224, 224, training=False) |
|  | print("Constructed FireNet ...") |
|  | model.load(os.path.join("models/FireNet", "firenet"),weights\_only=True) |
|  | print("Loaded CNN network weights ...") |
|  | ################################################################################ |
|  | # network input sizes |
|  | rows = 224 |
|  | cols = 224 |
|  | # display and loop settings |
|  | windowName = "Live Fire Detection - FireNet CNN"; |
|  | keepProcessing = True; |
|  | ######################################################### |
|  | if len(sys.argv) == 2: |
|  | # load video file from first command line argument |
|  | video = cv2.VideoCapture(sys.argv[1]) |
|  | print("Loaded video ...") |
|  | # create window |
|  | cv2.namedWindow(windowName, cv2.WINDOW\_NORMAL); |
|  | # get video properties |
|  | width = int(video.get(cv2.CAP\_PROP\_FRAME\_WIDTH)); |
|  | height = int(video.get(cv2.CAP\_PROP\_FRAME\_HEIGHT)) |
|  | fps = video.get(cv2.CAP\_PROP\_FPS) |
|  | frame\_time = round(1000/fps); |
|  | while (keepProcessing): |
|  | # start a timer (to see how long processing and display takes) |
|  | start\_t = cv2.getTickCount(); |
|  | # get video frame from file, handle end of file |
|  | ret, frame = video.read() |
|  | if not ret: |
|  | print("... end of video file reached"); |
|  | break; |
|  | # re-size image to network input size and perform prediction |
|  | small\_frame = cv2.resize(frame, (rows, cols), cv2.INTER\_AREA) |
|  | # perform prediction on the image frame which is: |
|  | # - an image (tensor) of dimension 224 x 224 x 3 |
|  | # - a 3 channel colour image with channel ordering BGR (not RGB) |
|  | # - un-normalised (i.e. pixel range going into network is 0->255) |
|  | output = model.predict([small\_frame]) |
|  | # label image based on prediction |
|  | if round(output[0][0]) == 1: |
|  | cv2.rectangle(frame, (0,0), (width,height), (0,0,255), 50) |
|  | cv2.putText(frame,'FIRE',(int(width/16),int(height/4)), |
|  | cv2.FONT\_HERSHEY\_SIMPLEX, 4,(255,255,255),10,cv2.LINE\_AA); |
|  | else: |
|  | cv2.rectangle(frame, (0,0), (width,height), (0,255,0), 50) |
|  | cv2.putText(frame,'CLEAR',(int(width/16),int(height/4)), |
|  | cv2.FONT\_HERSHEY\_SIMPLEX, 4,(255,255,255),10,cv2.LINE\_AA); |
|  | # stop the timer and convert to ms. (to see how long processing and display takes) |
|  | stop\_t = ((cv2.getTickCount() - start\_t)/cv2.getTickFrequency()) \* 1000; |
|  | # image display and key handling |
|  | cv2.imshow(windowName, frame); |
|  | # wait fps time or less depending on processing time taken (e.g. 1000ms / 25 fps = 40 ms) |
|  | key = cv2.waitKey(max(2, frame\_time - int(math.ceil(stop\_t)))) & 0xFF; |
|  | if (key == ord('x')): |
|  | keepProcessing = False; |
|  | elif (key == ord('f')): |
|  | cv2.setWindowProperty(windowName, cv2.WND\_PROP\_FULLSCREEN, cv2.WINDOW\_FULLSCREEN); |
|  | else: |
|  | print("usage: python firenet.py videofile.ext"); |

# **SYSTEM DESIGN**

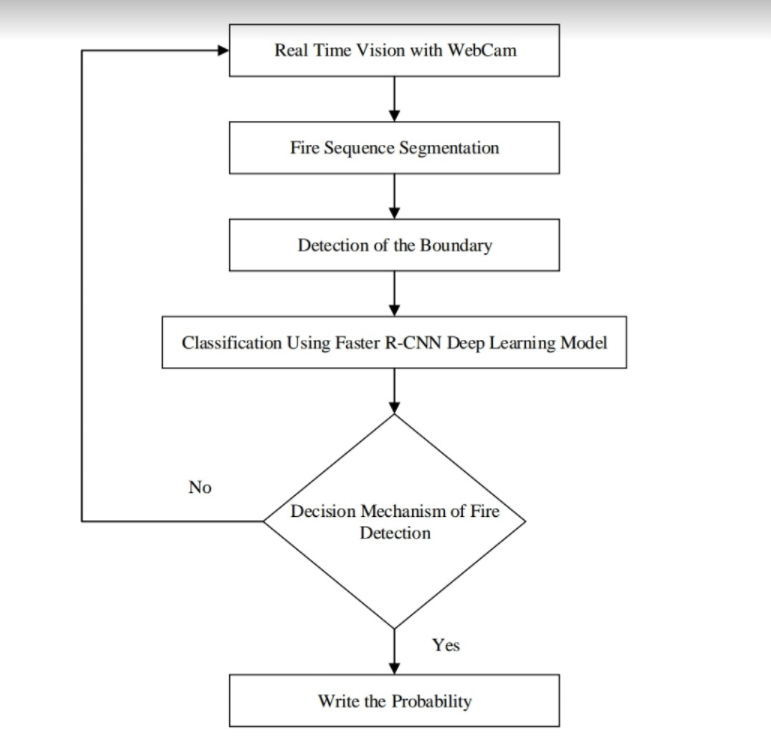
****

****

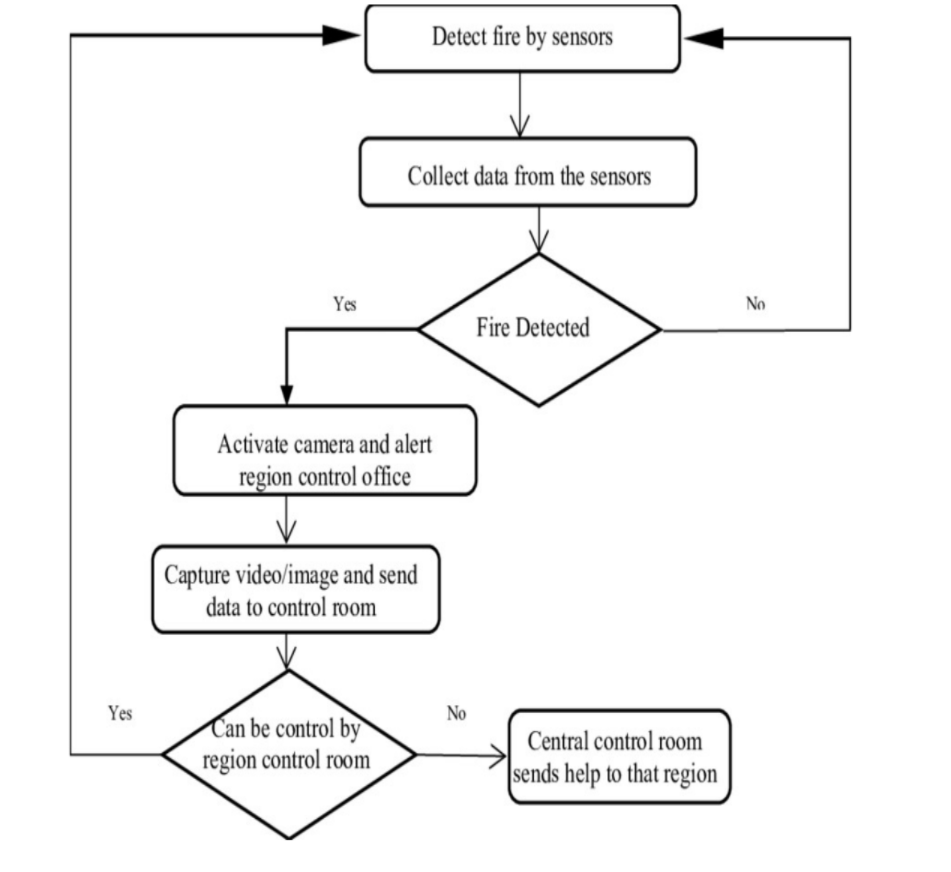
****

****

# **DATA FLOW DIAGRAMS**

****

**Figure. The flow chart of the fire detection process with a fire detector.**

****

**Figure. Data flow diagram of fire detection system.**

# **SYSTEM TESTING**

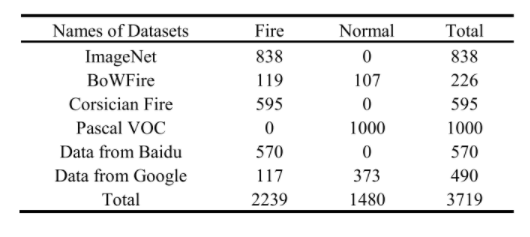
This section describes the datasets used in the experiments and the experimental results of the methods presented above. The hardware platforms used in the experiments are the Intel I7-9700K and the NVIDIA GeForce GTX TITAN X with 12 GB onboard memory. The experimental software development environment is Python, and the flame detection model is built by the PyTorch framework. The ResNet101 is used as the backbone network of the Faster R-CNN.

**A.DATASET DETAILS**

To make the sample of the training dataset as rich as possible, we used multiple datasets from

different sources

TABLE 2. Different datasets and their numbers of images



Representative example images from each dataset are shown in Table 1. The first is the ImageNet dataset, which contains 838 flame images (after manual filtration). The second is the BoWFire dataset, which is the most difficult among the current fire datasets. Although there are only 226 images in total, the non-flame images are very challenging. Many non flame images are similar to flame images in feature. The third is the Corsician Fire, including 595 flame images in which some images are dark in color; thus, the detection is more difficult. The fourth contains 1000 images randomly selected from the PascalVOC 2012 dataset as non-flame image data. Finally, some flame images and non-flame images were downloaded through Google and Baidu to improve the diversity of our dataset. The detailed statistics of each dataset is given in Table 2. The dataset for training the Faster R-CNN was obtained from Baidu, Google and ImageNet, with a total of 1525 images. To train the GIN, 1480 non-flame images are added in the training dataset. Tests were performed on the BoWFire dataset and the Corsician Fire dataset. The images in test datasets are not in the training datasets of the model. In a generic object detection task, the mean average precision (MAP) is often adopted as the metric for measuring the model performance. Since flame detection is single object detection, this metric is equivalent to the average precision (AP). However, in flame detection, there is no certain standard for the labeling of the flames. Therefore, the existing performance criteria for the flame detection are calculated in frames (accuracy, false positives, etc.) instead of using AP. However, in the training of the Faster R-CNN, it is still necessary to determine whether the model converges the correct direction by observing AP. To obtain the AP on the BoWFire dataset and the Corsician Fire dataset, we labeled the two datasets ∗.

For comparison with other methods, the metrics for measuring the model performance are also calculated in frames. For the image with flames, when all the flames are covered by detection boxes, then the detection is considered a True Positive. For non-flame images, if no detection box is generated, the detection result is considered a True Negative.

**B. COMPARISON WITH OTHER METHODS**

First, the effect of the color-guided anchoring strategy on the Faster R-CNN (Color-Guided in the following tables)

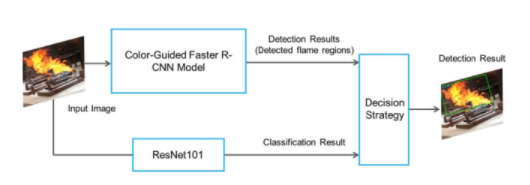


FIGURE 6. Model structure of global information-guided flame detection

method is studied on the BoWFire dataset. As shown in Table 3, the results are compared with the method of Chino et al. [10], which used handcrafted features for flame detection, and the methods of Muhammad et al. [16], [17], which used SqueezeNet and MobileNet. The comparison results of the accuracy, recall rate and F-measure metrics, and the comparison results of the false negatives, false positives and accuracy metrics are given in Table 3. This table demonstrates that the color-guided anchoring strategy can improve the recall of the flame detection. The flames in 97.48% of flame images are correctly detected, but this will also increase false positives. The overall accuracy of the dataset is slightly increased.

In addition, as noted in Table 3, the advantage of the Faster R-CNN is the recall of the flame target, but the drawbacks are the absence of the global information and the limitations of the training data.The number of false positives using only Faster R-CNN is greater than 25%, which indicates that this method alone is still insufficient in the face of challenging images. In addition, we found that most of the objects that are mistakenly identified as flames are warm color objects, such as sunsets and lanterns. In daily life, warm color objects appear fairly frequent, so it is important to use global information to guide flame detection.After using the color-guided anchoring strategy and the GIN (Color + Global Information Guided in the following tables), the false positives of the model are significantly reduced, and the accuracy also increases to 93.36%, reaching an advanced level. However, the recall is reduced because the training data are not rich enough. Since all the flame images of the test dataset are not trained in GIN, some flame features are unknown, and the GIN tends to mistakenly suppress the detection result of the Faster R-CNN. Compared with other methods based on CNN, our method is superior regarding precision, F-value, false positives and accuracy. Sample images of the detection results guided by GIN are given in Fig. 7. The artificial light source in Fig. 7(b) and the sun in Fig. 7(d) are not detected as flames, because the two images are noted as non-flame by GIN.

**C. TEST EXPERIMENTS OF PROPOSED METHODS**

We performed a series of additional tests for three methods, including the two proposed methods and the original Faster R-CNN. First, the efficiency test results for Color-Guided (the sparse anchoring strategy) and the original Faster R-CNN are shown in Table 4. Since the areas of interest in the mask TABLE 3. Comparison of different methods.

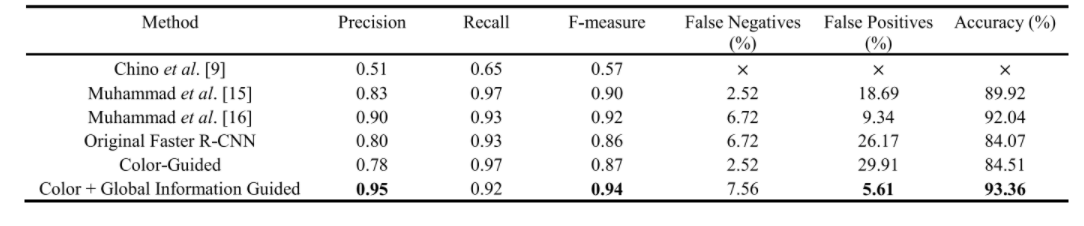




FIGURE 7. Detection results guided by GIN: (a) flame in a factory (top left); (b) street with artificial light sources (top right); (c) burning house (down left); (d) river in sunset (down right).

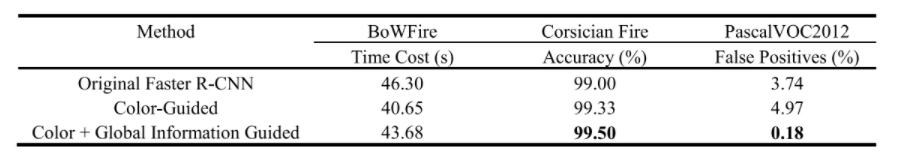


TABLE 4. Comparisons of the original Faster R-CNN and the two improved method on BoWFire,

Corsician Fire and PascalVOC2012 datasets.

are ignored in the masked convolution, the calculation speed is improved. From the comparison results, one can see that the improvement is not large because only the convolutional layers of the RPN are replaced. If the convolutional layers of the feature extractor are also replaced by the masked convolution layer, the detection speed can be further boosted. In addition, mask calculations take some time (approximately 3 s for the BoWFire dataset). In addition, the influence on the detection speed is not large after joining the GIN, and the influence is further reduced by running in two threads, thus suggesting the possibility of using a larger, more accurate CNN model as a GIN. Second, we tested the three methods on the Corsician Fire dataset to further examine their performance. Table 4 shows the experimental results on the Corsician Fire dataset. These results demonstrate that the Faster R-CNN using the color-guided anchoring strategy achieves the best accuracy. Finally, we performed experiments on the complete VOC2012 dataset (a total of 17,125 non-flame images) to examine the reliability of the two proposed methods in normal indoor and outdoor environments. The VOC2012 dataset contains 20 types of targets, which are divided into four categories: humans, birds, vehicles and indoor objects. The images of the dataset are not too challenging for flame detection compared to the non-flame images of the BoWFire dataset. The comparison results on the VOC2012 dataset are shown in Table 4. The results demonstrate that the overall false positives of the network can still reach 4% when there is no guidance of global information, and the false positives rate is only 0.27% when using global information. In the absence of global information, it is difficult for Faster R-CNN to meet the actual application requirements in the real-world flame detection tasks. Therefore, it is necessary to combine the global information to reduce false positives.

# 

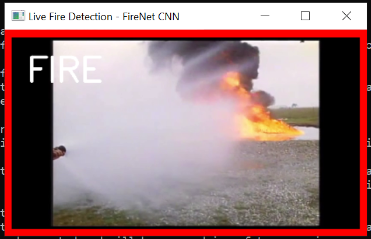
# 

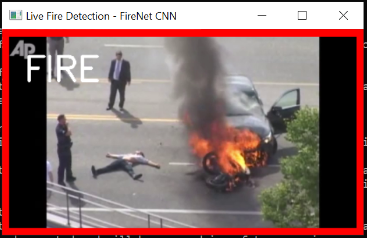
# 

# 

# 

# **SCREENSHOTS**



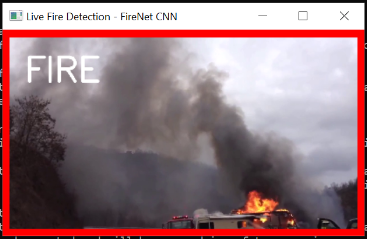
















# **CONCLUSION**

Fire is the most dangerous abnormal event, as failing to control it at an early stage can result in huge disasters leading to human, ecological and economic losses. Fire accidents can be detected using the cameras. So, here we proposed a CNN approach for fire detection using cameras. Our approach can identify the fire under the camera surveillance. This system is developed using the Faster R-CNN object detection model.We give images that include fire scenes and non-fire scenes to the system to get trained.Few images can be used in validation.This system can be tested in real-time using lighter as fire source.Furthermore, our proposed system balances the accuracy of fire detection and the size of the model using fine-tuning of datasets. We have obtained an accuracy of 99%. Also the F-measure value is 0.95. These values show that the model gives a better prediction. We conduct experiments using datasets collected from recording of fire and verified it to our proposed system. In view of the CNN model’s reasonable accuracy for fire detection, its size, and the rate of false alarms, the system can be helpful to disaster management teams in controlling fire disasters in a short time. Thus, avoiding huge losses. This work mainly focuses on the detection of fire scenes under observation. Future studies may focus on deploying the model into raspberry pi and using necessary support packages to detect the real time fire by making challenging and specific scene understanding datasets for fire detection methods and detailed experiments.

# **REFERENCES**

[1] A. M. Hasofer and I. Thomas, ‘‘Analysis of fatalities and injuries in building fire statistics,’’ Fire Saf. J., vol. 41, no. 1, pp. 2–14, Feb. 2006.

[2] J.-H. Kim and B. Y. Lattimer, ‘‘Real-time probabilistic classification of fire and smoke using thermal imagery for intelligent fire fighting robots,’’ Fire Saf. J., vol. 72, pp. 40–49, Feb. 2015.

[3] Y. Xin, S. Thumuluru, F. Jiang, R. Yin, B. Yao, K. Zhang, and B. Liu, ‘‘An experimental study of automatic water cannon systems for fire protection of large open spaces,’’ Fire Technol., vol. 50, no. 2, pp. 233–248, Mar. 2014.

[4] P. Barmpoutis, K. Dimitropoulos, K. Kaza, and N. Grammalidis, ‘‘Fire detection from images using faster R-CNN and multidimensional texture analysis,’’ in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Brighton, U.K., May 2019, pp. 8301–8305

[5] Q.-X. Zhang, G.-H. Lin, Y.-M. Zhang, G. Xu, and J.-J. Wang, ‘‘Wildland forest fire smoke detection based on faster R-CNN using synthetic smoke images,’’ Procedia Eng., vol. 211, pp. 441–446, 2018.

[6] T. Toulouse, L. Rossi, M. Akhloufi, T. Celik, X. Maldague, Benchmarking of wildland fire colour segmentation algorithms, IET Image Process. 9 (2015)1064–1072.

[7] T.-H. Chen, P.-H. Wu, Y.-C. Chiou, An early fire-detection method based on image processing, in: Proceedings of International Conference on Image Processing, ICIP’04, 2004, 2004, pp. 1707–1710.

[8] D. Guha-Sapir, F. Vos, R. Below, S. Penserre, Annual disaster statistical review 2015: the numbers and trends, 2015. http://www.cred.be/sites/default/files/ADSR\_2015.pdf.

[9] B. U. Töreyin, Y. Dedeoğlu, U. Güdükbay, and A. E. Çetin, ‘‘Computer vision based method for real-time fire and flame detection,’’ Pattern Recognit. Lett., vol. 27, no. 1, pp. 49–58, Jan. 2006.

[10] P. Foggia, A. Saggese, and M. Vento, ‘‘Real-time fire detection for video surveillance applications using a combination of experts based on color,shape, and motion,’’ IEEE Trans. Circuits Syst. Video Technol., vol. 25,no. 9, pp. 1545–1556, Sep. 2015.

[11]D. Y. T. Chino, L. P. S. Avalhais, J. F. Rodrigues, and A. J. M. Traina, ‘‘BoWFire: Detection of fire in still images by integrating pixel color and texture analysis,’’ in Proc. 28th SIBGRAPI Conf. Graph., Patterns Images,Salvador, Brazil, Aug. 2015, pp. 95–102.

[12] B.C. Ko, S.J. Ham, J.Y. Nam, Modeling and formalization of fuzzy finite automata for detection of irregular fire flames, IEEE Trans. Circuits Syst. Video Technol. 21 (2011) 1903–1912.

[13] P. Foggia, A. Saggese, M. Vento, Real-time fire detection for video-surveillance applications using a combination of experts based on color, shape, and motion,IEEE Trans. Circuits Syst. Video Technol. 25 (2015) 1545–1556.

[14] M. Mueller, P. Karasev, I. Kolesov, A. Tannenbaum, Optical flow estimation for flame detection in videos, IEEE Trans. Image Process. 22 (2013) 2786–2797.

[15] B.U. Töreyin, Y. Dedeoglu ̆ , U. Güdükbay, A.E. Cetin, Computer vision based method for real-time fire and flame detection, Pattern Recogn. Lett. 27 (2006) 49–58.

[16] R.C. Luo, K.L. Su, Autonomous fire-detection system using adaptive sensory fusion for intelligent security robot, IEEE/ASME Trans. Mechatron. 12 (2007)274–281.

[17] P.V.K. Borges, E. Izquierdo, A probabilistic approach for vision-based fire detection in videos, IEEE Trans. Circuits Syst. Video Technol. 20 (2010) 721–731.

[18] S. Frizzi, R. Kaabi, M. Bouchouicha, J.-M. Ginoux, E. Moreau, and F. Fnaiech, ‘‘Convolutional neural network for video fire and smoke detection,’’ in Proc. IECON - 42nd Annu. Conf. IEEE Ind. Electron. Soc.,Firenze, Italy, Oct. 2016, pp. 877–882.

[19] Z. Zhong, M. Wang, Y. Shi, and W. Gao, ‘‘A convolutional neural network based flame detection

method in video sequence,’’ Signal, Image Video Process., vol. 12, no. 8, pp. 1619–1627, Nov. 2018.

[20] Z. Yin, B. Wan, F. Yuan, X. Xia, and J. Shi, ‘‘A deep normalization and convolutional neural network for image smoke detection,’’ IEEE Access, vol. 5, pp. 18429–18438, 2017.

[21] O. Maksymiv, T. Rak, and D. Pileshko, ‘‘Real-time fire detection method combining AdaBoost, LBP and convolutional neural network in video sequence,’’ in Proc. 14th Int. Conf. Exper. Designing Appl. CAD Syst. Microelectron. (CADSM), Lviv, Ukraine, 2017, pp. 351–353.

[22] S. Wu and L. Zhang, ‘‘Using popular object detection methods for real time forest fire detection,’’ in Proc. 11th Int. Symp. Comput. Intell. Design (ISCID), Hangzhou, China, Dec. 2018, pp. 280–284.

[23] R. Girshick, J. Donahue, T. Darrell, and J. Malik, ‘‘Rich feature hierarchies for accurate object

detection and semantic segmentation,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Columbus, OH, USA, Jun. 2014, pp. 580–587.

[24] R. Girshick, ‘‘Fast R-CNN,’’ in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Santiago, Chile, Dec. 2015, pp. 1440–1448.

[25] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, ‘‘Microsoft COCO: Common objects in context,’’ in Proc. Europ. Conf. Comp. Vis. (ECCV), Zürich, Switzerland, 2014, pp. 740–755.

[26].Qingjie Zhang, Jiaolong Xu, Liang Xu and Haifeng Guo in ([2016) proposed a paper entitled Deep Convolutional Neural Networks for Forest Fire Detection in International Forum on Management, Education and Information Technology Application byAviation University ofAir Force, Changchun.

[27].Sebastien Frizzi1, Rabeb Kaabi, Moez Bouchouicha, Jean-Marc Ginoux in (2017) proposed a paper entitled Convolutional Neural Network for Video Fire and Smoke Detection in IEEE Transactions.

[28].Muhammad Khan and Jamil Ahmad in (2017) proposed a paper entitledEarly fire detection using

convolutional neural networks during surveillance for effective disaster management.

[29] Tan, C.F., Liew, S.M., Alkahari, M.R., Ranjit, S.S.S., Said, M.R., Chen W., Rauterberg, G.W.M., Sivarao D.S. “Fire Fighting Mobile Robot: State of the Art and Recent Development”, Australian Journal of Basic and Applied Sciences, Vol.10, Pages 220-230, 2013.

[30] Luo, R.C., Su, K.L. “Autonomous Fire-Detection System Using Adaptive Sensory Fusion for Intelligent Security Robot”, IEEE/ASME Transactions on Mechatronics, Vol. 12, Pages 274-281, 2007.

[31] B. U. Töreyin, Y. Dedeoğlu, U. Güdükbay, and A. E. Çetin, ‘‘Computer vision based method for real- time fire and flame detection,’’ Pattern Recognit. Lett., vol. 27, no. 1, pp. 49–58, Jan. 2006.

[32]P. Foggia, A. Saggese, and M. Vento, ‘‘Real-time fire detection for video surveillance applications using a combination of experts based on color, shape, and motion,’’ IEEE Trans. Circuits Syst. Video

Technol., vol. 25, no. 9, pp. 1545–1556, Sep. 2015.

[33] K. Muhammad, J. Ahmad, and S. W. Baik, ‘‘Early fire detection using convolutional neural networks during surveillance for effective disaster management,’’ Neurocomputing, vol. 288, pp. 30–42, May 2018.

[34] K. Muhammad, J. Ahmad, I. Mehmood, S. Rho, and S. W. Baik, ‘‘Convolutional neural networks

based fire detection in surveillance videos,’’ IEEE Access, vol. 6, pp. 18174–18183, 2018.

[35] K. Muhammad, J. Ahmad, Z. Lv, P. Bellavista, P. Yang, and S. W. Baik, ‘‘Efficient deep CNN-based fire detection and localization in video surveillance applications,’’ IEEE Trans. Syst., Man, Cybern. Syst., vol. 49, no. 7, pp. 1419–1434, Jul. 2019.

[36] K. Muhammad, S. Khan, M. Elhoseny, S. H. Ahmed, and S. W. Baik, ‘‘Efficient fire detection for

uncertain surveillance environment,’’ IEEE Trans. Ind. Informat., vol. 15, no. 5, pp. 3113–3122, May 2019.