Bachelor of Science in Computer Science and Engineering

Fire Detection Using Deep Learning

Jahidul Hasan
182001222E

Tithi Saha
191000822E

Shatu Moni Dey
191001622E

Fire Detection Using Deep Learning

This thesis is submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science & Engineering.

By

Jahidul Hasan
18200122E

Tithi Saha
191000822E

Shatu Moni Dey
191001622E

Supervised By
Arifa Sultana
Assistant Professor

Department of Computer Science and Engineering School of Science, Engineering and Technology East Delta University

EDU Permanent Campus, Noman Society, East Nasirabad, Chittagong-4209.

The thesis titled "**Fire Detection Using Deep Learning Algorithms**" submitted by ID No. 182001222e, 191000822e and 191001622e has been accepted as satisfactory in fulfillment of the requirement for the degree of Bachelor of Science in Computer Science & Engineering (CSE) as B.Sc. Engineering to be awarded by East Delta University (EDU).

Board of Examiners

1._____

Sohrab Hossain — Member
Assistant Professor
School of Science, Engineering & Technology
East Delta University (EDU)

2._____

Linkon Chowdhury — Member
Assistant Professor
School of Science, Engineering & Technology
East Delta University (EDU)

3._____

Mohammad Toufiq Ahmed. — Coordinator
Assistant Professor
School of Science, Engineering & Technology
East Delta University (EDU)

4. _____

Arifa Sultana
Assistant Professor — (Supervisor)
School of Science, Engineering & Technology
East Delta University (EDU)

Declaration

It is hereby declared that

The thesis submitted is our own original work while completing degree at East Delta University. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution. We have included literature work by others has been mentioned in the references section.

Students Name and Signature

_____

JahidulHasan
Date :

_____

TithiSaha
Date :

_____

ShatuMoniDey
Date :

Table of contents

Content **Page No**

Chapter 6 Result Analysis

Chapter 7 Conclusion And Future Work

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
AlexNet	AlexNet
BN	Batch Normalization
CNN	Convolutional Neural Network
DNCNN	Deep Normalization and Convolutional Neural Network
Darknet-53	Darknet-53
DenseNet	DenseNet
FD	Fire Detection
Faster RCNN	Faster Regions with CNN features
GPU	Graphical Processing Units
GoogleNet	GoogleNet
HIS	Hue Saturation Intensity
HSV	Hue Saturation Value
ILSVRC	ImageNet Large-Scale Visual Recognition Challenge
ImageNet	ImageNet
NN	Neural Network
PReLU	Parametric Rectifier Linear Units
ReLU	Rectified Linear Units
ResNet	Residual Neural Network
RGB	Red, Green and Blue
R-FCNN	Region-based Fully Convolutional Network
LeNet	LeNet
VGG	Very Deep Convolutional Networks for Large-Scale Image Recognition

Dedication

We dedicate this thesis to our parents, who have supported us and sacrificed so much to get us this far in our lives. We would also like to dedicate this thesis to our friends, who have cheered us on when things got difficult and have been an immense support throughout it all. This work also dedicated to our honorable supervisor who inspired us to think outside of the box and put in the hard work effort to produce better results. We would also like to dedicate this work to East Delta University which has given us this amazing platform to learn and grow.

Acknowledgement

First and foremost, we would want to express our appreciation to the Almighty for leading us through this research, for our good health throughout, and for making our journey to the completion of our research as smooth as possible. Then we'd like to express our gratitude to our parents for their support enormous faith in us, helping us through difficult moments, and as we finish our research. We hope that our achievement will make them proud. Next, we'd like to express our gratitude sincere to our respected supervisor Arifa Sultana, for her tolerance and dedication guidance and trust in us, as well as her vast expertise. She's been a fantastic mentor throughout everything. Without you, seeing the finish of our research would have been extremely tough.

And lastly, we want to thank our beloved institution East Delta University and express our sincere gratitude to the most respectable faculties, who have always encouraged us, as well as our seniors, juniors, friends, and family. They've all helped us stay motivated by believing in us and showing their unconditional support along the insightful journey. We don't want to forget to thank the internet for its existence and crucial role in the success of our research by assisting us in finding the most outstanding research articles and directing us to the incredible contributions of other researchers, making it much easier for us to achieve our goals.

Abstract

Fire can be considered an unfortunate phenomenon that can cause catastrophic damage to property and environment. It can also pose an immense threat to human safety and lives, especially when this hazard gets out of control. Early fire detection is very important for reducing fire-related losses. Therefore developing a robust fire detection algorithm is essential. It's can be based on algorithmic analysis of an image. So there are many algorithms to investigate fire detection. Each algorithm though poorer accuracy, slowly detection and extract every images features automatically. We known many algorithms such as Faster R-CNN, SSD and YoloV5 those entire algorithm based on convolutional neural network (CNN). In this work, we proposed a method of deep CNN to investigate fire detection in an image. We investigate the performance of actually designed, reduced complexity convolutional neural network architecture for this task as a follow-up to previous work in the field. In contrast to current contemporary trends, our research shows a maximum accuracy of 98% for whole image binary fire detection.

Keywords: Convolutional Neural Network (CNN), Fire Detection Algorithm, AlexNet, Custom CNN Model, Transfer Learning Technique, CNN Based Fire Detection.

Chapter1

Introduction

1.1 Introduction

The incidence of fires in Bangladesh has more than quadrupled in the last two decades, as the country's metropolitan areas have grown without basic infrastructure such as fire stations. Between January 1, 1999 and December 31, 2020, over 285,000 fires occurred in Bangladesh, according to data published by the Fire Service and Civil Defense [1]. Following fire service information, here 2,308 traffics are deaths in fire in this rural area between 2004 and 2020.The year 2019 had the most fire occurrences (24,074), while 2020 had the second most (21,073) [2,3].Aside from Chittagong, the number of fire events in this country grew from 200 to 675, indicating that the frequency has grown over time. Among 2,514 fire accidents, 47% occurred in residential areas and 27% occurred in commercial areas. Within the last 5 years, the Chittagong City Corporation dealt with a 179,091,200 BDT financial loss and 83 people were injured [4]. This chapter presents the hazardous effects of fire on human life and the environment which is one of the main motivational points of this research. It then discusses the aim and objectives of this research followed by setting its scope. Fire is one of the most devastating hazards that can cause not only catastrophic loss of lives but can also result in significant environmental, social and economic damages. Alarming facts motivate researchers to seek novel solutions for early fire detection and management. 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. In this research, we provide a novel dataset consisting of camera-captured fire photos. The images were taken with different points of view, different zoom, and camera types including regular and mobile cameras. Pile burns can be very helpful to study spot fires and early-stage fires. Piles must be monitored by fire managers for a few days after ignition to avoid spread outside the intended burn area. Using automated aerial monitoring systems can substantially reduce the forest management workload. To develop a sufficiently accurate model, most supervised learning approaches rely on huge training datasets. A fire dataset acquired from public sources to conduct fire detection using pre-trained ANN architectures such as MobileNet and AlexNet. However, that information was based on photographs of the fire taken from the ground. To the best of our knowledge, no aerial image dataset for fire analysis exists, which is critical for developing fire modeling and analytic

tools for aerial monitoring systems. It is important to note that aerial photography has distinct qualities, such as poor resolution and top-view viewpoint, than photographs captured by terrestrial cameras. After that here in this research we're used custom CNN (Convolutional Neural Network) model to identify fire with image classification.

1.2 Problem statement

Because of the unpredictable nature of fire, features are critical for image-based fire detection in outdoor contexts. Furthermore, in an open setting, the shape and velocity of the fire are determined by fuel and climatic circumstances, which might restrict the efficacy of present detection technologies.

For example, when fire starts at a great distance from the camera, it frequently looks as a static object, which is difficult to identify and yields a misleading result. As a result, the primary research goal is to demonstrate whether higher order descriptive characteristics can be leveraged to construct feasible systems for detecting fire and no fire in outdoor contexts using surveillance photos while keeping high detection accuracy and low result rates.

The image-based smoke and fire detection system necessitates more research into the following issues:

1. One of the main challenges of image based fire and no fire detection is to extract features as they tend to change adversely with rather chaotic variations in color, shade, motion and density. There is an identifiable deficiency in feature extraction methods which can provide detailed information about fire and no fire for image based detection in complex environments.

2. Despite their widespread popularity, deep learning algorithms for fire and no fire detection in surveillance footage have received little attention. Furthermore, no lightweight Convolutional Neural Network (CNN) model with a low false result that can detect both fire and no fire images is known to exist.

It is required to solve the stated research gaps, as well as examine the creation and assessment of image-based fire and no-fire detection systems, and then compare their performance to the common traditional approaches.

1.3 Aim of study

The general goal of this project is to design a fire and no fire detection system that uses both conventional and deep learning approaches to assure its detection capabilities in complicated outdoor situations, as well as to produce an usable model that includes prediction and detection for RUI zones. The overarching aim leads to the primary objectives listed below: Determine the following sets of characteristics for detecting fire and no fire: Identifying and using a set of high discriminative and light-insensitive local and global characteristics for an effective fire and no-fire detection approach. Utilize CNNs for fire and no-fire detection: Investigating cutting-edge CNN models for fire and no-fire detection and comparing them to traditional hand-crafted feature-based approaches. Create a model: Create a modest CNN model for integrated fire and no fire detection using modern deep learning techniques. Create an RUI fire safety model: A method of predicting fires by combining meteorological data and deep learning algorithms. This goal is to create a revolutionary fire monitoring system that will aid in the prediction and detection of fires. Validity testing: Reliability and validity testing will be carried out in a systematic manner.

1.4 Research methodology

We have created custom CNN model with AlexNet architecture. For training and testing the CNN models. A new custom CNN model inspired by AlexNet has been created by merging the concepts of Batch Normalization (BN) and Rectified Linear Units (ReLU). The combination of BN with ReLU improves the model's generalization and representation capability, resulting in improved accuracy when compared to AlexNet. Aside from greater accuracy, the suggested custom model includes fewer learnable parameters and a smaller memory space than the original AlexNet model. After model ready we use Adam optimizer with binary cross-entropy where Adam optimizer beats the previously employed optimizers by a wide margin in terms of providing an optimal gradient descent. On a variety of typical benchmark datasets, the custom model has been evaluated to identify both fire and no fire. Experiment results reveal that this lightweight model can detect fire and no fire effectively, as evidenced by the high detection accuracy (98 percent).

1.5 Contributions to knowledge

Several innovative contributions have been made and presented in this thesis, including:

1. A novel feature extraction approach, Local Binary Co-occurrence Patterns for RGB Color Plane (RGB), has been presented and is being used successfully for image-based smoke detection. The suggested technique provides an effective measure of picture texture, which aids in the detection of complex structures in fire zones. The co-occurrence of neighboring LBPs for each channel is then assessed in order to compute six co-occurrence features. Furthermore, when color information is combined with the co-occurrence of LBP features, it offers spatial details for a complex texture of fire zones.

2. The effectiveness of using CNNs with transfer learning in connection to the problem of fire and no fire detection has been examined by evaluating characteristics from different layers. The effect of training samples on fine-tuned CNNs has been investigated in the domains of fire detection and no fire detection. The performance of fine-tuned CNNs was compared to that of well-distinguished pre-trained CNNs utilizing off-the-shelf features and handcrafted feature-based techniques, as a benchmark for the fire and no-fire detection problems.

1.6 Scope of research

This thesis primarily focuses on the detection of fire and no fire using surveillance photos. One of the most difficult issues in image-based fire and no-fire detection is extracting features since they vary greatly in color, shade, motion, and density. The major emphasis is on extracting distinguishing descriptive elements from the input image frame in order to detect fire and no fire. It focuses on dynamic texture aspects in particular and presents a full experimental exploration of combining local and global picture features. The study also explores exploring the usefulness of CNNs on the problem of fire and no fire detection by applying transfer learning and off-the-shelf features. The capacity of modern state-of-the-art networks is tested in the context of image-based fire and no fire detection.

Furthermore, the goal of this research is to create a tiny lightweight model inspired by any of the prominent CNN models that may be used for combined fire and no fire detection. This

study is guided by the latest deep learning notions of performance enhancement applied to fire and no fire detection. This project intends to design a fire safety model that includes both fire prediction and detection for the total fire monitoring system.

1.7 Thesis outline

The remaining chapters of this thesis are outlined as follows.

Chapter 2 Presents the context for the thesis's planned research. The basic framework of the image-based detection system is presented first, followed by literature on the various phases of the framework. It gives an overview of the research on deep learning-based approaches for fire detection in Chapter 1: Introduction. It also explains how convolution neural networks function and how they relate to fire.

In chapter 3 Our datasets have been described. The data for the custom CNN model has been discussed.

Chapter 4 Describes the all CNN layers and function. This chapter provides an in-depth look at the model as well as a logical explanation of the notion. The hyper-parameters of the model and the training technique are highlighted as crucial aspects of CNN model design.

In chapter 5 Includes machine specifications, data enhancement, and a CNN model set for fire detection.

Chapter 6 Describes the creation, testing, assessment, analysis and limitation of a bespoke CNN model for fire detection.

Chapter 7 Describes conclusion and future work of fire detection.

Includes an overview of the thesis's major contributions, conclusions, and recommendations for further research.

Chapter 2

Related works

2.1 Convolutional Neural Network

2.1.1 Overview of Convolutional Neural Network

CNN is a profound neural network initially intended for picture investigation. CNN is built on the association and utility of the visual cortex and is meant to mimic the physically of neurons within the human cerebrum. Each group of neurons in the CNN is divided into a 3D structure, which is then allotted a tiny area of the picture. Similarly, each neuronal cluster has some competence in recognizing one aspect of the image. The multiple filters of the convolutional layers execute convolutional operations. It is consistently composed of two essential activities: convolution and pooling. Convolutional layers are created utilizing a few element maps, and each unit of highlight maps is created by convolving a small region from the information stream known as the adjacent responsive field. Pooling layers are commonly used after convolutional layers, which were established to reorder data and reduce the size of highlight maps. As a result, pooling layers in convolutional layers create a dense element map from each element map. These layers are also referred to as subsampling layers. The two most common pooling cycles are max-pooling and average-pooling. A CNN compresses a fully linked network by lowering the number of connections and sharing weights at the network's edges. Furthermore, max-pooling decreases complexity even further. CNNs, like MLP learning weights, will achieve competency with the best filters for recognizing explicit patterns and instances during training. CNN, on the other hand, learns a variety of filters throughout the process. Here the anatomy of all CNN layer.

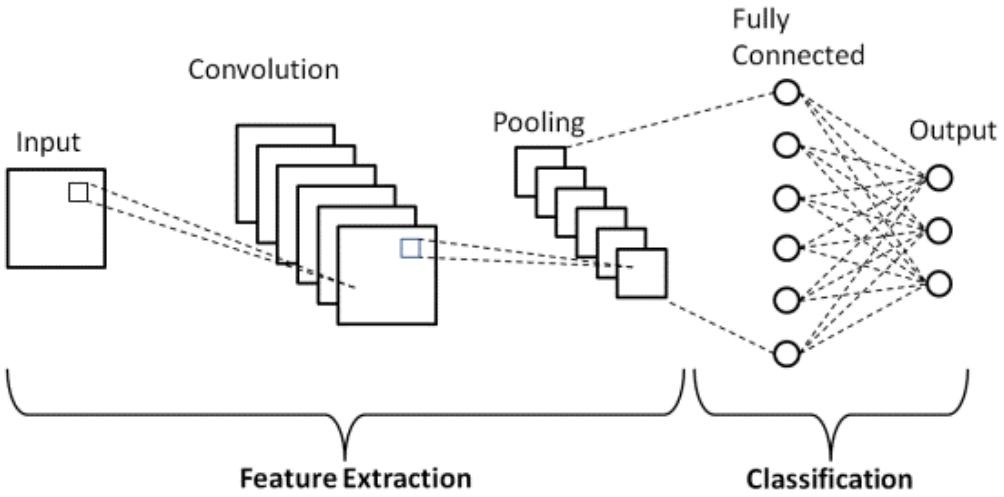


Figure 2.1.1.1: Basic structure of Convolutional neural network.

2.1.2 Related research

Punam Patel et al. set a paper for fire/flame detection using cnn system. Image-based fire detection, it has been argued, necessitates a number of consecutive frames from the original video, which includes both fire and non-fire images. It is separated into three stages: identification of fire pixels using the RGB and YCb color models, detection of moving pixels, and examination of the shape of fire colored pixels in frames. This method is used to identify fire in video sequences [1]. JareeratSeebamrungsat et al. proposed a study for fire detection utilizing a CNN image classification method. It was proposed to utilize the Color Segmentation method to separate fire from its backdrop. To distinguish the flame colors from the backdrop, the features of the HSV and YCbCr color models are applied. The HSV color model is used to identify color and brightness information. Then, for five consecutive frames, they determine the amount of white frames by dividing the difference between the preceding and actual frames [2]. Khan Muhammad et al. proposed an article titled effective deep CNN-based fire detection and localization in video surveillance systems. This method developed the intelligent feature map selection technique for selecting relevant feature maps from the learned CNN's convolutional layers that are sensitive to fire areas. When compared to previous handmade approaches, these feature maps allow for more precise fire segmentation. Using this, the model's size was decreased from 238 MB to 3 MB, lowering the cost and simplifying implementation. Another characteristic of this system is its capacity to recognize a burning object using a pre-trained model [3]. Abdulaziz et.al., study for a fire detection belongs deep learning approach. In limited dataset for prevent overfiting problem, it can be suggested to a

convolutional neural network and for fire and smoke images add new dataset to train and evaluate the model. Then it's can be improved training images as well as augmentation system. They gathered a new dataset consisting of fire and smoke photos that can be used to train and test the network, allowing the network to learn fire and smoke characteristics under diverse weather and light conditions [4]. Qingjie Zhang et.al., set a paper which is forest fire detection using deep learning technique. They employ a tiny subset to design and test the method. Due to the short size of our annotated dataset, they studied two types of classifiers in this work: linear classifiers and non-linear classifiers. For our binary classification, they reduced the number of outputs in the final fully linked layer to two. They also decreased overfitting in this network, which is trained in just a few hundred iterations and achieves a surprisingly high training and testing accuracy [5]. Sebastien Frizzi1 et al., create an article in IEEE Transactions which is video fire and smoke detection using deep learning cnn technique. Their primary goal is to determine if a picture contains fire or smoke. They divided the photos into three subsets: training (60%), validation (20%), and test (20%). To enhance fire detection and localization on video, we must update our training set. The training data was generated using a computer equipped with an Intel Xeon microprocessor (frequency CPU 3,1 GHz, RAM 16 GB) and a graphic card GTX 980 Ti (2816 cores, 6 GB memory).

Furthermore, they compare the algorithm's classification accuracy on the test set to conventional approaches across a broader range of video fire pictures such as various materials, sources, and ventilations [6]. Muhammad Khan et.al., set a paper which is fire detection using cnn techniques. A methodology is proposed that eliminates the time-consuming and difficult process of feature engineering by automatically learning rich characteristics from raw fire data. Several kernels of varying sizes are used to the input data to build feature maps in these. These feature maps are sent into the following step, known as subsampling or pooling, which selects the most activations from them within a narrow neighborhood. The purpose for such conception is to prevent unpredictable increases in computing complexity and network flexibility in dramatically increasing the number of units at each step. Furthermore, these methods enhance fire detection accuracy while minimizing false alarms, however the model size is fairly large at roughly 240 MB [7]. Sergio Saponara, AbdussalamElhanashi&AlessioGagliardi research about video fire/smoke detection using CNN techniques. It's specially using YOLOv2 algorithm for detect fire/smoke which is the one of algorithm of CNN. The author employed a vast scale of fire/smoke and negative movies in various situations, both inside and outdoor, in this case. The method takes the input picture and separates it into SXS grids for the proposed

YOLOv2 approach, which used entire images to train and test the network. It extracts the grid's characteristics and predicts the bounding boxes as well as the confidence score for the discovered items in these boxes. The confidence score will be equal to the IoU value between the ground truth and the bounding boxes if there is an item. Otherwise, if there is no object, the confidence score will be 0.

2.2 Fire detection and Neural network

2.2.1 Fire detection in Neural network

Deep learning is a type of machine learning approach that mimics the physiology and activity of neurons in the neocortex. The design was inspired by the human brain, and it is capable of classifying photos, sounds, and texts straight from raw input. Convolutional neural networks (CNNs or ConvNet), a widely used deep learning architecture, have reached cutting-edge accuracy in object identification and recognition, as well as picture segmentation and classification. In this work, a supervised machine learning algorithm is employed to identify camera collected frames. When fire and non-fire portions coexist in a mixed picture, the frame is deemed to be fire-labeled, and when there is no fire in the frame, it is termed non-fire-labeled. It's a challenging work in image analysis of CNN. Traditional image processing algorithms used RGB channel comparison to distinguish distinct items in frames or films, such as fire. These classic approaches are not error-free and are not completely trustworthy [8]. For example, RGB value comparison algorithms that typically use a threshold value to identify fire may detect sunset and dawn as a false positive result.

2.2.2 Popular CNN Architecture

Hubel and Wiesel identified the basic features of light detection of the cell receptive fields in the monkey cerebral cortex in 1968 [9]. Kunihiki Fukushima presented the NeoCognitron, a neural network model for visual pattern recognition, as a result of this study [10]. The NeoCognitron can be thought of as the forefather of current CNNs. Yan LeCun et al. developed the pioneering contemporary CNN architecture LeNet for handwritten digit recognition in 1990 [11]. LeNet was further enhanced to LeNet-5, which are multilayer convolutional networks trained using the back propagation technique. When a conventional CNN design won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) in 2012, CNNs became extremely popular among researchers. Krizhevsky et al. introduced AlexNet, a breakthrough model comparable to LeNet-5 but with a deeper structure and the ability to handle millions of

pictures [12]. Many studies have used AlexNet to effectively classify shots [13], remote sensing picture scene classification [14], computer-aided detection [15], parking lot occupancy detection [16], fingerprint detection [17], and fire detection. Many unique designs have been built as a result of the success of the AlexNet, including VGG, GoogleNet, ResNet, and Inception [18-21]. The new models are increasingly deeper and more powerful as CNNs evolve. For example, the ILSVRC winner in 2015, ResNet, has 152 layers, but AlexNet had only eight levels in 2012. ResNet research demonstrates the value of its use in visual recognition [22], as well as vehicle classification and localization [23]. Other CNN networks that are popular among academics include Densely connected network (DenseNet) [24], FractalNet [25], and Xception [26].Girshick et al. [27] created R-CNN by combining region proposals with CNNs, which was effectively employed for object detection and semantic segmentation. The sluggish training procedure of RCNN was a noteworthy constraint, which Fast R-CNN [28] addressed while enhancing detection speed and quality. Similarly, Ren et al. [29] introduced Faster R-CNN, which may create more efficient and accurate region recommendations. The investigation of these three R-CNN models is a continuous topic among computer vision experts. R-CNN, Fast R-CNN, and Faster R-CNN have been utilized effectively in pedestrian detection [30-32], fire and smoke detection [33, 34], cell identification [35], face detection [36], and vehicle detection [37-38].

These popular CNN architectures are mostly trained for probable object categories in the PASCAL VOC [39], IMAGENET [40], and MS COCO [41] benchmark datasets. However, the number of item categories in available benchmark datasets remains insufficient in contrast to the number of real-world objects recognized by people [25,32]. Among the things excluded from these popular benchmark datasets are fire and normal photos. It is obvious that CNN requires massive amounts of labeled data for training from start, which is computationally costly. These findings imply that one of the major obstacles in detecting fire and normal pictures is a lack of data sets, which might be solved through transfer learning.

2.2.3 Transfer Learning Technique

In deep learning approaches, transfer learning is a common and successful methodology. The idea behind transfer learning is to use pre-existing models as a starting point for training and transfer information from a comparable activity. In the case of transfer learning, fine-tuning a network is more easier and lot faster than starting from scratch. Early layers in the construction of CNNs contain general properties that may be re-used for a variety of applications. Otherwise

last layer is the more application-specific. The first layers are well-preserved as a result of this feature, while the final ones are altered to train with the new dataset of interest [46, 47].

2.2.4 CNN Based Fire Detection

Several ways for enhancing the accuracy of smoke and fire detection using CNNs have recently been developed. To improve the performance of smoke detection, Yin et al. suggested a unique deep normalization and convolutional neural network (DNCNN) inspired by ZFNet [44]. It was composed of fourteen layers, comprising eight normalization and convolutional layers, three pooling layers, and three fully linked layers. DNCNN used data augmentation to boost the amount of training examples from the supplied limited dataset. There were 17,315 original photos and 25,533 enhanced images in the collection. Muhammad et al. developed a GoogleNet-inspired architecture for video-based fire detection that achieved an accuracy of 94.43 percent [45]. Aerial imaging pile burn detection using deep learning with flame dataset, on the other hand, comprises video recording and thermal heatmap acquired by cameras in the dataset. In this case, two machine learning tasks are used: (1) Exist or not exist fire flames in video frames for binary classification. An Artificial Neural Network (ANN) technique is created with a classification accuracy of 76%. (2) Fire detection by the use of segmentation approaches to properly define fire boundaries. Our FLAME technique has an accuracy of 92 percent and a recall of 84%. The approach for free burning broadcast fire utilizing thermal pictures will be expanded in future study.

Chapter 3

Data collection and preprocessing

3.1 Dataset for training and testing the CNN model

We utilized many datasets from kaggle, many datasets from github, and many data personally acquired by camera for training and testing the CNN model. This collection is made up of RGB photos, with a total of 1000 images. We utilized around 500 photographs of fire, and 500 images are standard. We utilized 200 photos for testing and 800 images for training out of a total 1000 images.

Table 3.1.1: Dataset distribution.

Category	Training Data	Validation data
Fire	500 images	100 images
Normal	500 images	100 images

3.2 Dataset preparation

Datasets are important in the performance of CNN because unbalanced training data has a detrimental influence on overall performance. This section discusses the collection of picture dataset information. Deep learning has emerged as the preferred way for tackling a wide range of difficult issues. As we all know, with appropriate training, such deep networks can recognize the image's main elements. If a simple mechanism is large enough, it can have a magical effect. As a result, this well-functioning deep learning necessitates a large amount of data. The more training data there is, the greater the model's accuracy. This effort focused on creating a balanced dataset that included white, grey, black, and blue smoke, as well as yellow, orange, and reddish fire, for better training reasons. We also evaluated dense, light, sparse, and turbulent fire distributions in order to provide a balanced and demanding dataset. Furthermore, we have incorporated several sizes of fire and normal, ranging from a modest thin normal/fire to a catastrophic one. It is also worth noting that the dataset we created comprises many features such as clouds, trees, and normal/fire colored objects that might easily be mistaken with the properties of normal and fire.

Figure 3.2.1: Sample of fire images from the database.

Figure 3.2.2: Sample of normal images from the database.

Sample frames from the dataset including normal, fire and negative images are shown in Figure 3.2.1 and 3.2.2. It should be mentioned here that, this dataset is essentially different from the dataset in terms of size and complexity.

Chapter 4

Methodology

One of the most difficult problems in the image processing arena is picture categorization. Traditional image processing approaches used RGB channel comparison in the past to detect distinct items in frames, such as fire. For example, RGB value comparison algorithms that typically use a threshold value to detect fire may mistakenly identify sunset and sunrise. However, training a DNN to execute this picture classification job aids in the learning of non-fire-related components. The buried layers rely on depth-wise separable convolutions and use convolution blocks as shortcuts. The hidden layers' entrance flow is a pair of 2-Dimensional (2D) convolutional blocks with kernel size 3 and default stride of 2x2. Batch normalization and a Rectified Linear Unit (ReLU) activation function are applied to each block.

4.1 Used Tool

Jupyter Notebook:

The Jupyter Notebook is a web-based server-client program for editing and executing notebook pages. The Jupyter Notebook App can be run locally on a computer without internet access (as explained in this paper) or remotely on a server and accessible through the internet. The Jupyter Notebook App contains a "Dashboard," a "control panel" that shows local files and allows you to open notebook papers or shut down their kernels, in addition to displaying/editing/running notebook documents.

Numpy:

In Numpy library developer can conduct Mathematical and logical operations with arrays. Shape modification via Fourier transformations and algorithms. Algebraic operations are referred to as linear algebra operations. NumPy includes linear algebra and random number generating routines.

Matplotlib:

It's one of Python's most capable charting packages. It is a cross-platform package that provides a variety of tools for creating 2D charts in Python from data stored in lists or arrays.

Computer Vision (CV):

The CV is an abbreviated form of computer vision in OpenCV, which is described as a branch of research that assists computers in understanding the content of digital pictures such as photographs and videos. The goal of computer vision is to figure out what's going on in the images. It extracts the description from the images, which may be an item, a written description, a three-dimensional model, or something else entirely.

Scikit Learn (Sklearn):

It uses a Python consistency interface to give a set of fast tools for machine learning and statistical modeling, such as classification, regression, clustering, and dimensionality reduction. NumPy, SciPy, and Matplotlib are the foundations of this package, which is mostly written in Python.

TensorFlow:

Developers may use TensorFlow to design dataflow graph structures, which define how data flows across a graph, or a sequence of processing nodes. Each node in the graph symbolizes a mathematical process, and each node-to-node link or edge is a tensor, or multidimensional data array.

4.2 Fire detection algorithm

Figure is the concise version of the custom CNN model. The size of the input layer depends on the image size and the number of channels which in our case is (150 × 150). Then the value of RGBs in different channels is scaled to a float number between 0 and 1.0. The hidden layers rely on depth-wise separable convolutions and shortcut between the convolution blocks. The entry flow of the hidden layers is a pair of 2-Dimensional (2D) convolutional blocks with a kernel size of 3 and a stride of 2 × 2. Each block follows a batch normalization and a Rectified Linear Unit (ReLU) activation function. The batch normalization is used to speed up the training process and bring more randomness by decreasing the importance of initial weights and regularize the model. Next, the model follows two separable 2D convolutional blocks. The last block of the hidden layer is a separable 2D convolutional layer with a kernel size of 3 followed by another batch normalization and the ReLU function. Since the fire-detection is a binary classification task (Fire/No Fire), the activation function for the output layer is a sigmoid function. Here the algorithm of custom CNN model-

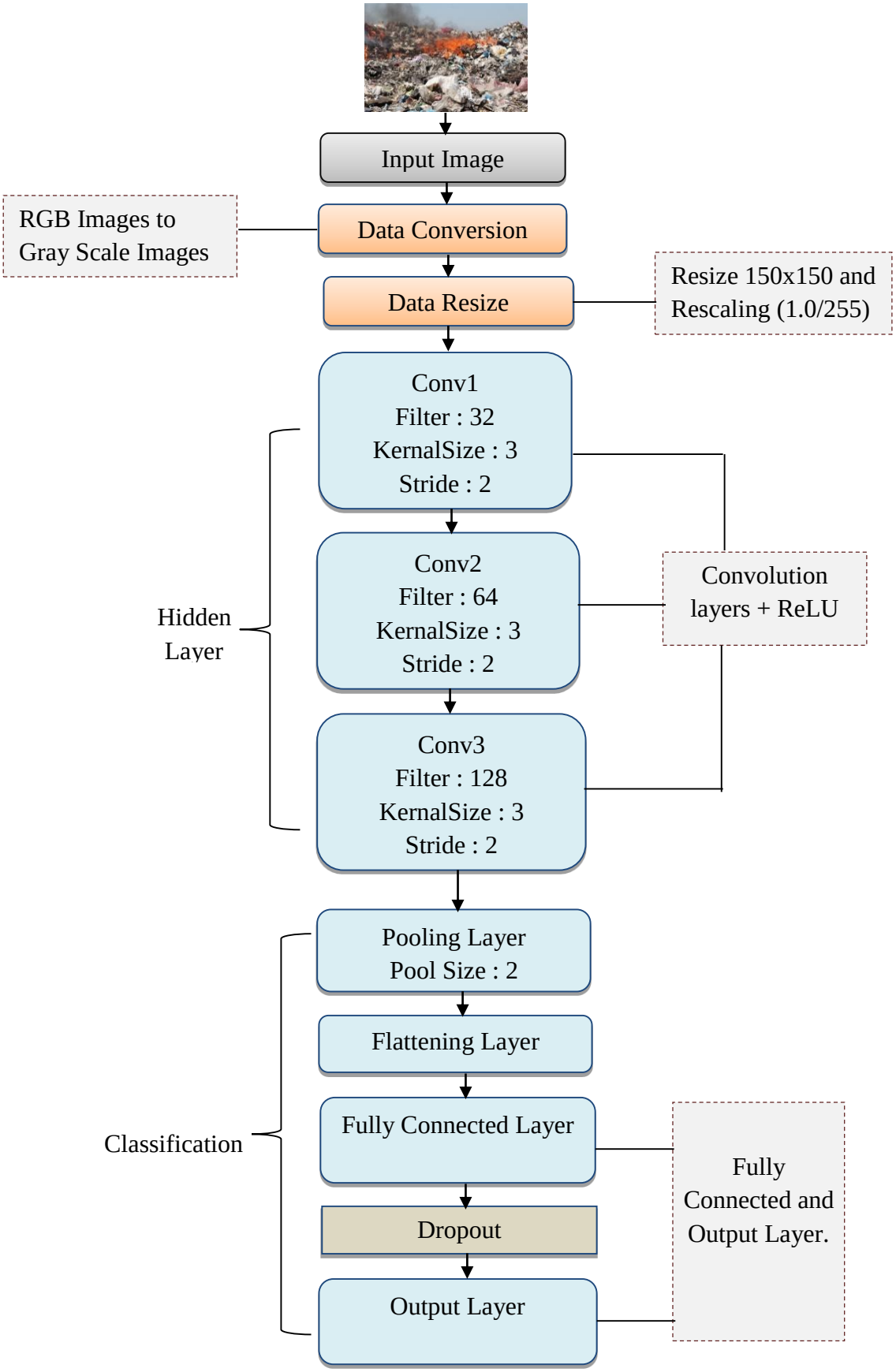
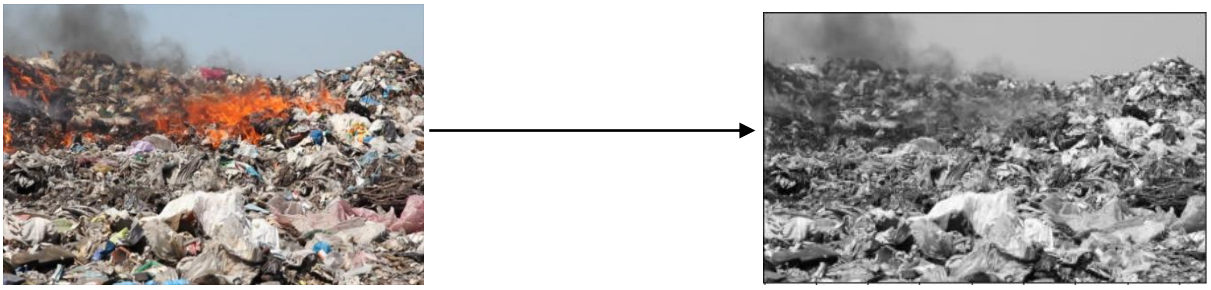


Figure4.2.1: Flowchart of proposed fire detection model.

4.3 Data conversion

Converting a color image to a grayscale image with prominent characteristics is a difficult procedure. The converted grayscale image may lose the color image's contrasts, clarity, shadows, and structure. A novel method has been presented to maintain the contrasts, sharpness, shadows, and structure of a color image. The new technique conducts RGB approximation, reduction, and addition of chrominance and brightness to transform a color image to a grayscale image. The grayscale images produced by the algorithm in the experiment demonstrate that the algorithm kept the important elements of the color image, such as contrast, sharpness, shadow, and image structure. We used 1000 photos of fire and normal in this dataset. Images with the extensions jpg/jpeg/png First, we read all of the photos from the dataset. We first read the binary file as raw binary input before converting the user data to grayscale pictures. The length of the data is then stored in a numpy array. The data is transformed to a vector, and the data length array is shaped into a square and padded with zeros. Here after conversion RGB to Gray scale.



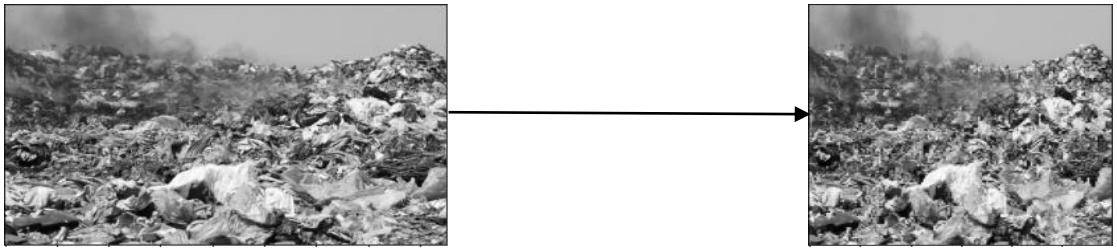
RGB **GRAY**

Figure 4.3.1: Image Conversion RGB to Gray scale.

4.4 Data resize

Image resizing is a key step in the preparation of computer vision. Our deep learning models, in general, train quicker on smaller pictures. Furthermore, many deep learning model designs need that our photos be the same size, although the size of our raw gathered images may vary. Because neural networks only accept inputs of the same size, all photos must be reduced to a set size before being fed into the CNN. The greater the fixed size, the less shrinking is needed. Less shrinking means less distortion of visual features and patterns.

After resize images with fixed 150 x 150 size.

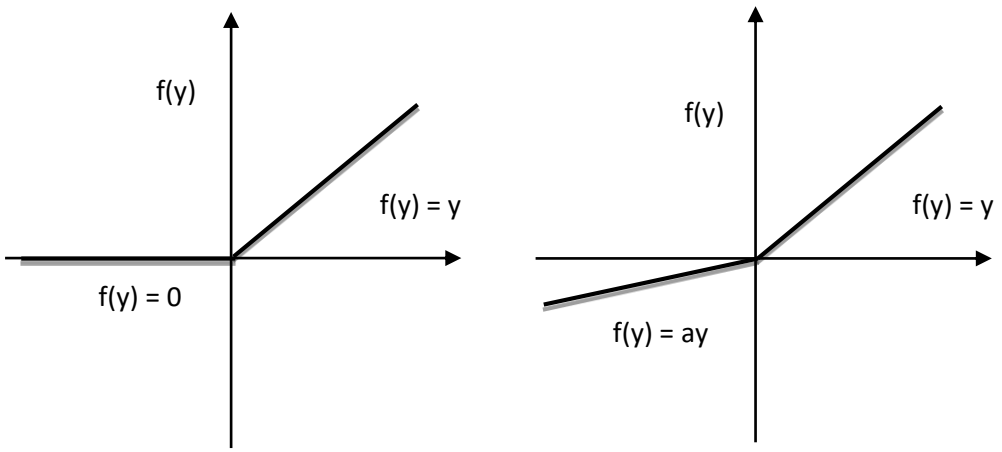


Default size **Resize with 150 x 150**

Figure 4.4.1: Output of resized image.

4.5 Convolutional layers and functions

We describe an ensemble deep learning strategy for improving deep learning prediction accuracies in the Fire and No fire classifications. We propose a network for integrated no fire and fire detection, inspired by CNN's recent success. The CNN model is created by borrowing some of the fundamental designs of the AlexNet [12] and then customizing the network with batch normalization (BN) and Rectifier linear units (ReLU). We discuss our suggested model for no fire and fire detection in the following subsections. In addition, we describe how BN and ReLU are coupled to create a model.The activation function is one of the deep learning model's innovative contributions since it endows deep networks with non-linear features. It has a substantial influence on the performance of deep neural networks. The Parametric Rectifier Linear Unit (PReLU) is a popular activation function that has reached human-level performance in picture categorization and detection. It learns the parameters of the recitifiers adaptively from the input data. PReLU is a function that regulates the activation of the ReLU. When given a negative input, the ReLU function returns to zero. Because of its zero gradient characteristic, it may slow down training. They cannot learn using gradient-based approaches since the gradients are all 0. The comparison learning graph between ReLU and PReLU is depicted in Figure 5.



(a) (b)

Figure 4.5.1: Comparison between ReLU (a) and PReLU (b).

For ReLU, the coefficient of the negative part is zero. For PReLU, the coefficient of the negative part is adaptively learned. Mathematically, PReLU is defined as follows:

$$f(y_i) = \begin{cases} y_i, & if\ y_i > 0 \\ a_i y_i, & if\ y_i \leq 0 \end{cases} \text{.....(4.5.1)}$$

Here, y_i represents the nonlinear activation's input, while f and a_i are the coefficients determining the slope of the negative half. The number of additional parameters for PReLU is extremely limited and there is no extra risk of overfitting. Convolution and normalization extract high-dimensional features, which increases the danger of overfitting and, as a result, increases computing time. As a result, a pooling layer is added after these layers to reduce the dimension of the feature maps. Convolutional layers are one of the primary components of Convolutional Neural Networks (CNNs). This layer's kernels or filters compute several feature maps. Each neuron in a feature map corresponds to a tiny area in the following layer. Convolving the input data with a kernel yields the new feature map. Mathematically, the feature map of the lth layer is given by:

$$X_j^l = \sum_{i=1}^{N^{t-1}} X_i^{l-1} * K_{ij}^l + b_j^l \text{.....(4.5.2)}$$

where X_j^l is the lth layer's output, X_i^{l-1} is the previous layer's output, K_{ij}^l is the filter's kernel for the lth layer, and b_j^l is the bias of the lth layer K_{ij}^l kernal is an abbreviation forshared by all of the input's spatial positions This sharing technique has the potential to minimize model

complexity and make training the network easier. The suggested model is made up of five convolutional layers that are then batch normalized. In our model, we used six layers: three convolution layers, two dense layers, and one flatten layer. First and foremost, the CNN model must be initialized. There are three convolution layers: filter, kernel size, padding, and activation. Here Convolutional Neural Network filters detect variations in picture intensity levels to recognize spatial patterns such as edges. And the kernel is nothing more than a filter that extracts features from pictures. The kernel is a matrix that looping the input images, computes the dot product using the provided data subregion and returns the matrix of dot products. Let's see the kernel matrix and see how it's work-

0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

Input Matrix

0	-1	0
-1	5	-1
0	-1	0

Kernel Matrix

320	206	198	188	
210	89			

Output Matrix

Figure 4.5.2 : Kernel matrix.

Padding is three types one is "Same" and other is "Valid" and the another is "Casual". We're keeping padding the same in those layers since it helps us apply padding to the input picture so that it's completely covered by the filter and given stride. It's termed SAME because the output for stride 1 will be the same as the input. Let's see in a figure of padding and see how it's works-

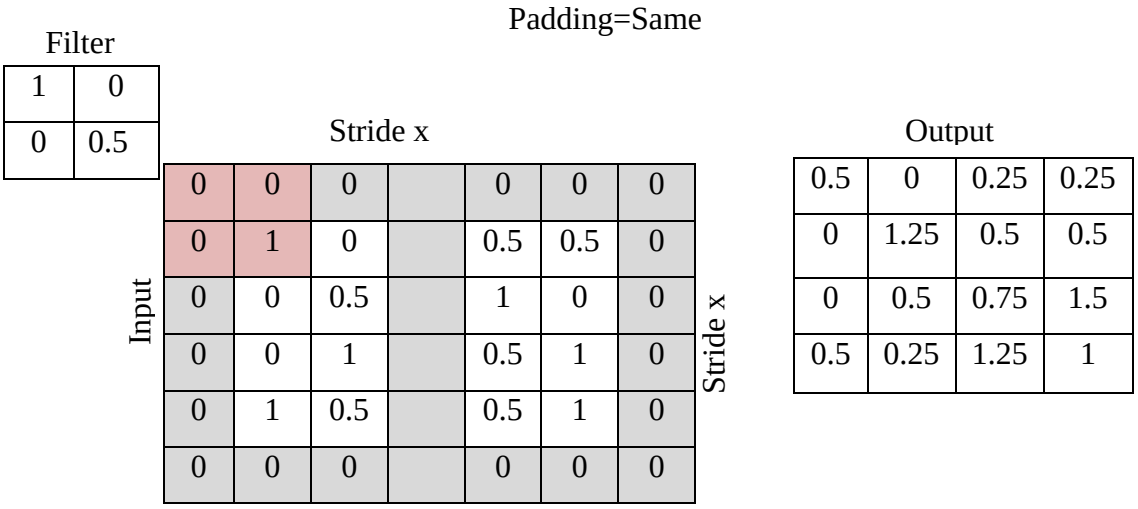


Figure 4.5.3: Added padding 0 in a number of pixel of image.

4.5.1 Pooling layers

The pooling layers' goal is to execute a down sampled operation on the input feature maps, which simply changes the dimension of the output feature maps. This layer, in particular, maintains the same number of input and output feature maps while reducing the dimension of the feature maps based on the size of the window. The pooling actions can be expressed mathematically as follows:

$$X_j^l = \text{Pool}(X_j^{l-1}) \text{ .....(4.5.1.1)}$$

where pool(.) denotes the pool function of the input feature map X_j^{l-1}. Max pooling [48] and average pooling [49] are two widely utilized pooling operations in the deep learning models. This work has employed the max pooling operation system. In the max pooling technique, the maximum value is chosen from the NxN window of the feature maps. After a stack of convolutional, batch normalization and pooling layers, one or more fully connected layers are used that has the ability to perform high level reasoning [50].Set a max pool with 2x2 size after those convolution layers. Following individual convolutional layers, max pooling is a sort of procedure that is frequently applied to CNNs. When max pooling is applied to a model, it decreases picture dimensionality by lowering the amount of pixels in the preceding convolutional layer's output. However, we are only using one maximum pool for three convolutional layers since too many lower the amount of pixels, which may result in the loss of our provided picture data and the inability to forecast object correctly. How the max pool work and see this figure-

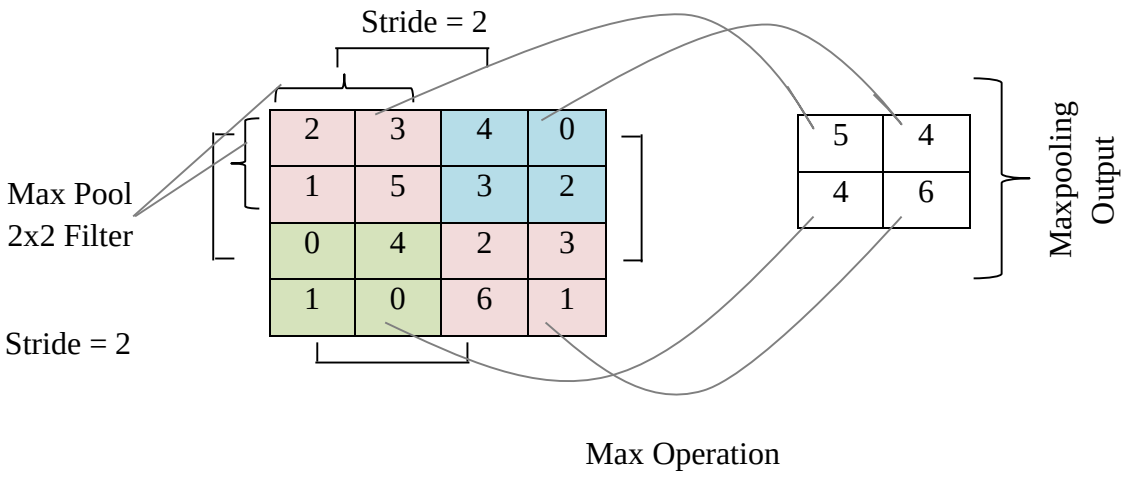


Figure 4.5.4: Max pool operation with pixels.

4.5.2 Flatten layer

Flattening a tensor involves removing all dimensions except one. In Keras, a Flatten layer reshapes the tensor to have a shape equal to the number of components in the tensor. This is equivalent to creating a 1d-array of items. The flatten layer is used to convert the data into a one-dimensional array for use as input to the fully connected layer. Flattening the output of the preceding layers yields a single lengthy feature vector. Add one flatten; rectangular or cubic forms cannot be used as direct inputs here. This is why flattening and fully-connected layers are required. Flattening is the process of turning data into a one-dimensional array for input into the following layer. When max pooling is applied to a model, it decreases picture dimensionality by lowering the amount of pixels in the preceding convolutional layer's output. After that, two thick layers were added, with this dense layer being used to categorize images based on the output of the convolutional layers.

4.5.3 Fully connected layer and output layer

The completely linked layers use paired connections to connect neurons from two neighboring layers. However, neurons in a single layer do not exchange any connections. To efficiently create global semantic knowledge, these layers accounts for the majority of learnable characteristics. We used three completely linked layers in our study, comparable to AlexNet [12]. Because completely linked layers are prone to overfitting, the dropouts approach is frequently employed to reduce the likelihood of overfitting [51]. In this study, the softmax layer is employed for classification after three completely linked layers. The softmax layer's goal is to compute the probability for each class using the retrieved high-dimensional characteristics.

The output layer has three neurons that represent the three classifications of smoke, fire, and negative. The softmax layer's probability may be calculated as follows:

$$\hat{y}_j = \frac{\exp(x_j^f)}{\sum_{i=1}^{3} \exp(x_j^f)} \text{ .....(4.5.3.1)}$$

Where $\hat{y}_j$ is the probability of the jth neuron, j=1,2,3denotes the three classes, and x_j^f denotes the fth layer's feature maps. The layers shown above are merged to form our proposed convolutional neural network for combined smoke and fire detection. Some hyper-parameters must be set and optimized for improved performance since they are crucial components for each specific application. After the fully connected layer add, we need to add a dropout. When the model overfit, this layer aids in preventing overfitting. It should be noted that the Dropout layer is only active while training is set to True, which means that no values are lost during inference. When using model.fit, training is automatically set to True, as opposed to specifying trainable=False for a Dropout layer. Trainable has no effect on the behavior of the layer because Dropout does not have any variables/weights that may be frozen during training. The dropout hyper parameter's default meaning is the likelihood of training a particular node in a layer, where 1.0 implies no dropout and 0.0 denotes no layer outputs. dropout value with a hidden layer can be between 0.5 and 0.8. Input layers employ a higher dropout rate, such as 0.8.

The output layer of an artificial neural network is the last layer of neurons that provides the program's outputs. In order to obtain the ultimate output, we must apply a completely linked layer that produces an output equal to the number of classes required. It becomes difficult to get that number using only the convolution layers. Convolution layers create 3D activation maps, but we just need to know whether or not a picture belongs to a specific class. To compute prediction error, the output layer employs a loss function similar to categorical cross-entropy.

4.5.4 Batch normalization layer (BN)

This model employs a batch normalizing layer after each convolutional layer. The batch normalization layer's goal is to speed up network training while reducing sensitivity to parameter initialization. Ioffe et al. [52] emphasized the difficulties associated with training deep neural networks owing to internal covariate shift. Batch normalization, which normalizes each training mini-batch, was introduced to solve this issue. A non-linear activation function follows each BN. We use rectifier linear units (ReLU) in this suggested model because of their human-level performance in picture categorization [53].

4.5.5 Sigmoid activation function

The sigmoid activation function, often known as the logistic function, is an S-shaped nonlinear activation function. The function's input is converted to a value between 0 and 1. This trait makes the sigmoid activation function beneficial when utilized in the network's output layer to predict probability as an output. Because of this characteristic, the sigmoid function is an excellent option for use in our CNN models.Because fire detection is a binary classification job (Fire/No Fire), the output layer's activation function is a Sigmoid function.

$$\boldsymbol{P(label = Fire) = \sigma\left(label = Fire \middle| \zeta(\theta)\right) = \frac{1}{1+e^{-\zeta(\theta)}}} \text{.....(4.5.5.1)}$$

where ζ(θ) is the value of the output layer retrieved from the input frames and the RGB values of each pixel, as well as all the weights across the hidden network. is the weight for the network's last layer. The Sigmoid function's output value is the likelihood of fire detection based on the imported frames into the network.

4.5.6 Cross entropy loss function

Loss/cost functions are used to optimize the model during training while working on a Machine Learning or Deep Learning problem. Almost generally, the goal is to minimize the loss function. The better the model, the lesser the loss. Cross-Entropy loss is a critical cost function. It is employed in the optimization of classification models. Understanding of Cross-Entropy is dependent on knowledge of the Softmax activation function. Loss functions are used to minimize errors in the network during the training process. To measure the performance of a model whose output is a probability value between 0 and 1, cross-entropy loss, or log loss is used. On the one-shot encoded output, CNNs apply a cross-entropy loss. It appears to be

$$L(\theta) = -\sum_{c=1}^{M}(y_c \cdot \log \hat{y}_c) \text{.....(4.5.6.1)}$$

where number of classes is M and for that class y_c is model's forecast. Cause all the labels is one-hot encoded and vector of ones and zeroes is y, it's can 1 else 0. As a result, only one word will be included to the total: the one with y_c=1.

4.5.7 Adam optimizer

Adam is an adaptive learning rate optimization technique built exclusively for deep neural network training. It was first released in 2014, Adam is mixture of RMSprop and Stochastic Gradient include momentum. Squared gradients used in scales of the learning rate. It's like RMSprop, and utilized of the moving average of the gradient used momentum. It's similar to SGD including momentum. In certain circumstances, Adam produces a solution that is poorer than stochastic gradient descent. Here we see how it's works with this equation.

$$m_n = E[X^n]$$

M=moment, X=random variable

However, adam is the most popular algorithm, cause it's achieved good result fast, hence we use this algorithm. Here we see compare between adam and other optimization technique with training multilayer perception. [It's a theory of Stochastic Optimization, 2015].

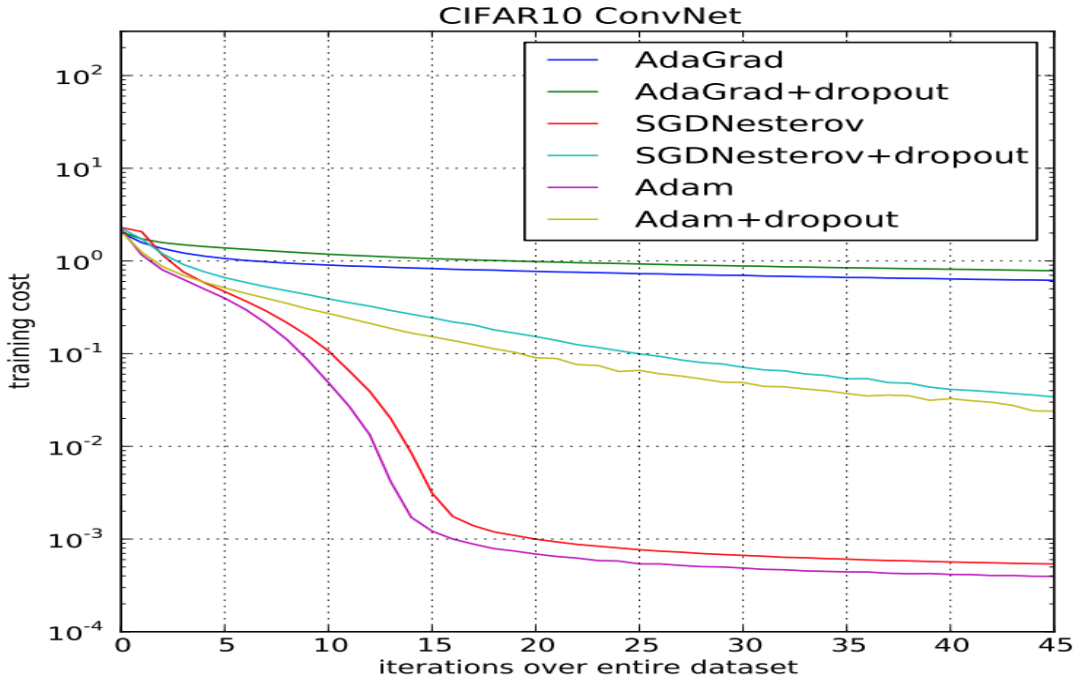


Figure 4.5.7.1: Comparison of Adam and other optimization.

4.5.8 Hyper-parameters optimization

The phrase "hyper-parameters" refers to a set of factors that influence network architecture and network training. Model optimization is the process of fine-tuning these hyper-parameters, which is one of the most difficult aspects of designing a task-specific CNN model. Kernel size, learning rate, number of hidden units, activation functions, number of epochs, batch size, percentage of chance of dropout, number of hidden layers and neurons in fully connected layers, callbacks with tensorboard, and others are among the hyper-parameters. The correct mix of hyper-parameters can make a considerable difference in state-of-the-art performance, making hyper-parameter adjustment a critical aspect of CNN training. In addition, some initialization can be detrimental for the generalization property. To avoid these difficulties and at the same time, to speed-up the optimization process, we have reused pre-trained networks after fine-tuning them based on our proposed idea. Studies of using pre-trained networks for parameter optimization suggest that an optimal combination of hyper-parameters for a given data set might be a good starting point for other similar types of data sets.

4.6 Custom CNN Model

AlexNet is one of the popular architecture of CNN. The Alexnet is made up of eight layers, each having its own set of parameters that may be learned. The model is composed of five layers, the first of which is a max pooling layer, followed by three fully connected levels, and each of these layers, with the exception of the output layer, uses ReLU activation. They discovered that employing the ReLU as an activation function nearly six times expedited the training process. They also made use of dropout layers to keep their model from overfitting. Here the architecture of AlexNet-

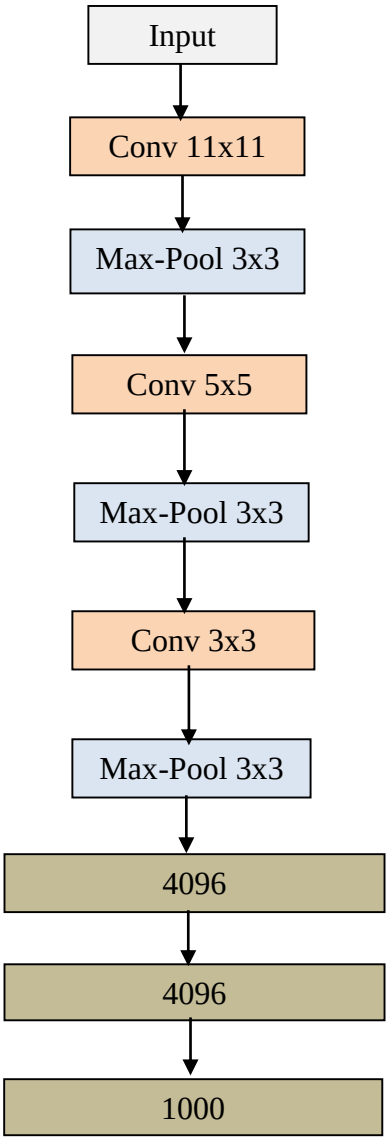


Figure 4.6.1: AlexNet architecture of CNN.

In that case follow AlexNet, In our model we have used Six layers, namely three convolution layers, two dense layers and one flatten layer . We divide the fire/nofire dataset into 80 percent train set and 20 percent test set ratio. The first convolution layer receives a grayscale picture with a fixed size of 150 × 150. In the first convolution layer we used 32 filters of 3 x 3 kernel size, in the second layer of convolution we used 64 filters of 3 x 3 kernel size and the third convolution layer we used 128 filters of 3 x 3 kernel size with keeping padding same and default stride. Next, we used a pool size of 2 x 2 in the max pooling layers. Default strides are used for both convolution and max pooling layers. After that we added flattening layer which belong fully connected layers. The dropout layer drops 50 percent neurons applied first dense

layer. The three convolution and one dense layers use the ReLU activation function. However, the last dense layer uses the sigmoid activation function to classify whether an image belongs to any of the fire classes or not. We also used loss function cross entropy and adam optimizer for optimized our algorithm while model compiling. Here our model looks like-

Table 4.6.1: Table of custom CNN Model.

Layer	Filter/ neurons	Filter Size	Padding	Activation Function	Stride	Input Shape
Input		3 x 3				
Conv1	32	3 x 3	same	ReLU	Default=1	1:
Conv2	64	3 x 3	same	ReLU	Default=1	
Conv3	128	3 x 3	same	ReLU	Default=1	
MaxPool		2 x 2				
Flatten						
Dense	Unit=128			ReLU		
Drop	Rate=0.5					
Dense	Unit=1			Sigmoid		

Chapter 5

Experimental Setup and Implementation

5.1 Machine specification for implementation

For implementing and testing all the model, we used,

- CPU: Intel core i3 1.8ghz
- GPU: AMD Radion 2 GB
- RAM: 4GB

5.2Model Summary

Each Layer of the Neural Network comprises neurons that compute the weighted average of its input and then send this weighted average via a non-linear function known as a "activation function." And the addition of one Dropout layer with a 50% success rate is critical in training CNNs since it prevents overfitting on the training data. If they are not there, the first batch of training data has a disproportionately large impact on learning. Let' see the model summary of custom CNN-

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 150, 150, 32)      320
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 150, 150, 64)      18496
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 150, 150, 128)     73856
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 75, 75, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 720000)            0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               92160128
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 129
=================================================================
Total params: 92,252,929
Trainable params: 92,252,929
Non-trainable params: 0
_________________________________________________________________
```

Figure5.2.1 : Model summary of CNN.

Following the completion of the model summary, we will compile the model with function loss binary cross entropy, Adam optimizer, and accuracy with matrices. This binary cross entropy is utilized for a loss function in binary classification jobs. These are assignments with only two alternatives for answering a question: yes or no. Several separate queries of this type can be addressed concurrently, as in multi-label classification or binary picture segmentation. This loss is the average of the categorical cross entropy losses on various two-category issues. We employed architectural hyper-parameters similar to those used by AlexNet[12]. To gain a better grasp of the impact of hyper-parameters on overall performance, we carefully tweaked only one parameter at a time and ran tests on the dataset. Here are some parameters that were utilized during the model fit-

Table 5.2.1: Table of hyper-parameters while model fit.

Batch size	**Epochs**	**Validation split**	**Callbacks**
32	10	20%	Tensorboard

Chapter 6

Result Analysis

6.1 Results

In Google CoLab, running 10 epochs for training and testing the custom CNN model takes about 43 minutes. This model has a total size of 1 GB when it was saved to be loaded from the python script. On the other hand, running 10 epochs for training and testing the Alexnet model took roughly 6 minutes, and the saved model is 62 MB. After training and testing both models, the custom CNN model had an accuracy of 98 percent while the AlexNet model had an accuracy of 89 percent, owing to the fact that the custom CNN model had substantially fewer layers and hence neurons than the AlexNet model. Here the AlexNet model fit-

```
Epoch 1/10
25/25 [==============================] - 36s 1s/step - loss: 0.8099 - accuracy: 0.5337 - val_loss: 0.6369 - val_accuracy: 0.7150
Epoch 2/10
25/25 [==============================] - 35s 1s/step - loss: 0.5637 - accuracy: 0.7525 - val_loss: 0.4590 - val_accuracy: 0.8250
Epoch 3/10
25/25 [==============================] - 35s 1s/step - loss: 0.4600 - accuracy: 0.7887 - val_loss: 0.4229 - val_accuracy: 0.8600
Epoch 4/10
25/25 [==============================] - 35s 1s/step - loss: 0.4222 - accuracy: 0.8150 - val_loss: 0.4001 - val_accuracy: 0.8600
Epoch 5/10
25/25 [==============================] - 35s 1s/step - loss: 0.3875 - accuracy: 0.8425 - val_loss: 0.4421 - val_accuracy: 0.8550
Epoch 6/10
25/25 [==============================] - 35s 1s/step - loss: 0.3891 - accuracy: 0.8425 - val_loss: 0.4366 - val_accuracy: 0.8700
Epoch 7/10
25/25 [==============================] - 35s 1s/step - loss: 0.3400 - accuracy: 0.8662 - val_loss: 0.4190 - val_accuracy: 0.8600
Epoch 8/10
25/25 [==============================] - 35s 1s/step - loss: 0.3097 - accuracy: 0.8763 - val_loss: 0.3833 - val_accuracy: 0.8750
Epoch 9/10
25/25 [==============================] - 35s 1s/step - loss: 0.2493 - accuracy: 0.8988 - val_loss: 0.4044 - val_accuracy: 0.8650
Epoch 10/10
25/25 [==============================] - 35s 1s/step - loss: 0.2044 - accuracy: 0.9187 - val_loss: 0.3706 - val_accuracy: 0.8900
```

Figure6.1.1: AlexNet model result.

```
Epoch 1/10
26/26 [==============================] - 69s 3s/step - loss: 0.2607 - accuracy: 0.9169 - val_loss: 0.0988 - val_accuracy: 0.974
7
Epoch 2/10
26/26 [==============================] - 66s 3s/step - loss: 0.0514 - accuracy: 0.9892 - val_loss: 0.1221 - val_accuracy: 0.994
4
Epoch 3/10
26/26 [==============================] - 64s 2s/step - loss: 0.0286 - accuracy: 0.9940 - val_loss: 0.0366 - val_accuracy: 0.994
4
Epoch 4/10
26/26 [==============================] - 65s 3s/step - loss: 0.0225 - accuracy: 0.9964 - val_loss: 0.0185 - val_accuracy: 0.994
4
Epoch 5/10
26/26 [==============================] - 64s 2s/step - loss: 0.0167 - accuracy: 0.9928 - val_loss: 0.0275 - val_accuracy: 0.988
8
Epoch 6/10
26/26 [==============================] - 65s 3s/step - loss: 0.0269 - accuracy: 0.9940 - val_loss: 0.0257 - val_accuracy: 0.994
4
Epoch 7/10
26/26 [==============================] - 64s 2s/step - loss: 0.0113 - accuracy: 0.9964 - val_loss: 0.0097 - val_accuracy: 0.997
2
Epoch 8/10
26/26 [==============================] - 64s 2s/step - loss: 0.0057 - accuracy: 0.9988 - val_loss: 0.0691 - val_accuracy: 0.994
4
Epoch 9/10
26/26 [==============================] - 65s 3s/step - loss: 0.0340 - accuracy: 0.9904 - val_loss: 0.0370 - val_accuracy: 0.988
8
Epoch 10/10
26/26 [==============================] - 64s 2s/step - loss: 0.0103 - accuracy: 0.9964 - val_loss: 0.0151 - val_accuracy: 0.997
2
```

Figure 6.1.2: Custom CNN model result.

The better a model is, the lesser the loss. The loss is calculated for both training and validation, and the model's interpretation is how well it performs for these two sets. Loss is not quantified in percentages, unlike precision. It's a total of all the mistakes made in training or validation sets for each example. For classification and regression, the loss in neural networks is commonly negative log-likelihood and residual sum of squares, respectively. The basic goal of a learning model, then, is to lower (minimize) the loss function's value in relation to the model's parameters by modifying the weight vector values using various optimization approaches, such as back propagation in neural networks. After each optimization iteration, the loss value reveals how well or poorly a model performs. After each, or several, iterations, one would want to see a decrease in loss (s).A model's correctness is normally measured after the model parameters have been learnt and fixed, with no learning going place. After that, the test samples are fed into the model, and the number of errors (zero-one losses) made by the model are recorded, after which the model is compared to the genuine targets. The proportion of misclassification is then determined. Here the differentiate of custom CNN model and AlexNet model loss and accuracy curves-

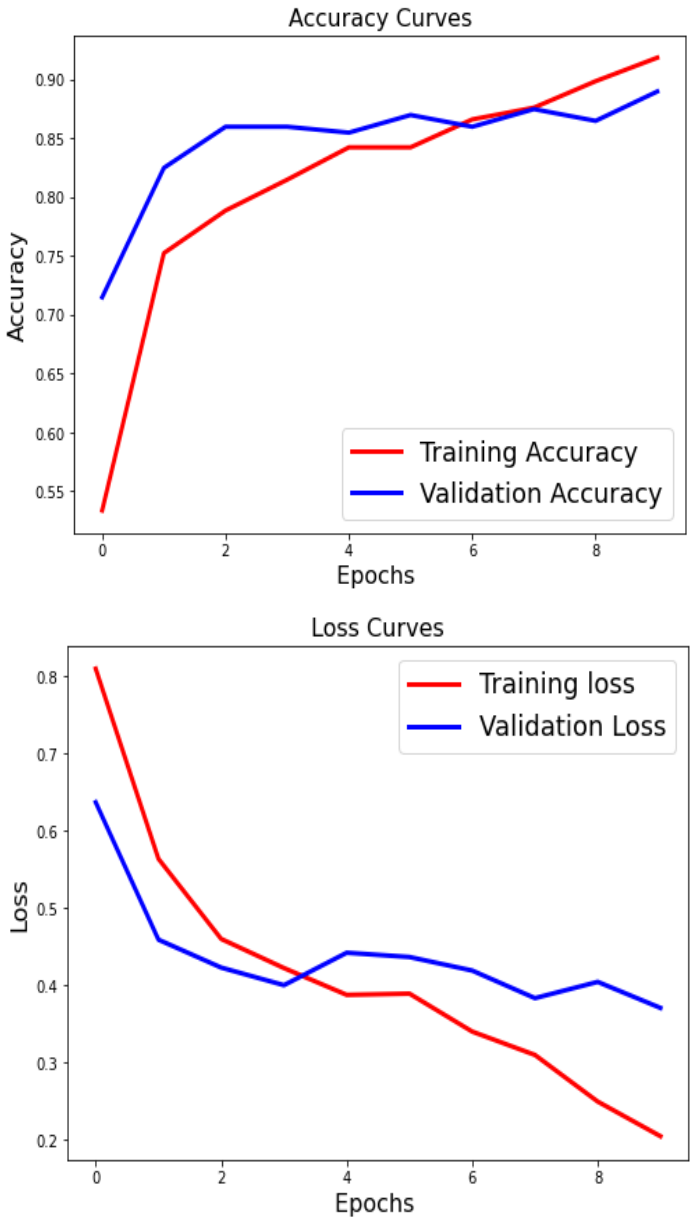


Figure6.1.3 :Alexnet model accuracy and loss curves.

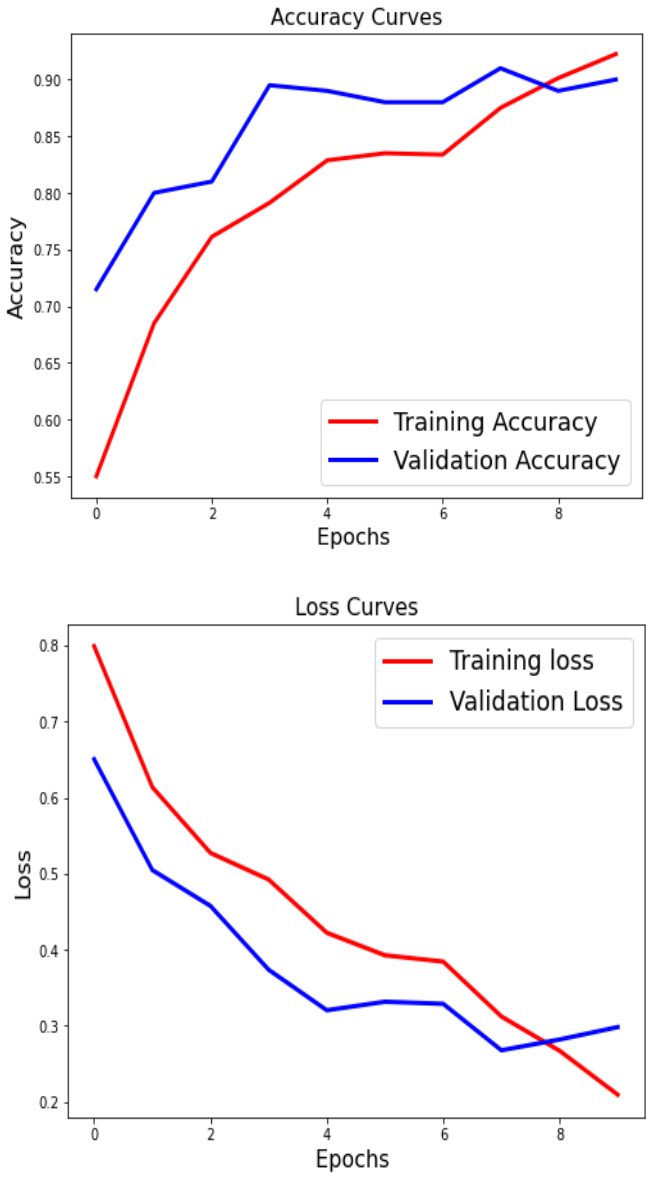


Figure 6.1.4: Custom CNN model accuracy and loss curves.

However, after compare the custom CNN model loss and accuracy curve then we're going to show confusion matrix. In One class belongs each input of binary classification of two classes. In most cases, these two groups are given names such as 1 and 0, or positive and negative. The confusion matrix allows us to see if the model is "confused" when it comes to distinguishing between the two classes. It's a 2x2 matrix, as seen in the next diagram. Positive and Negative are the labels for the two rows and columns, which correspond to the two class labels. The ground-truth labels are represented by the row labels, whereas the predicted labels are represented by the column labels in this case. This may be altered. The matrix's four elements (the red and green items) reflect the four metrics that tally the number of correct and wrong

predictions generated by the model. Each element is assigned a label consisting of two words: True or False, and Positive or Negative. It is True when the prediction is right and False when the predicted and ground-truth labels do not match. It's could be positive or it's could be negative. The structure of the confusion matrix is seen here

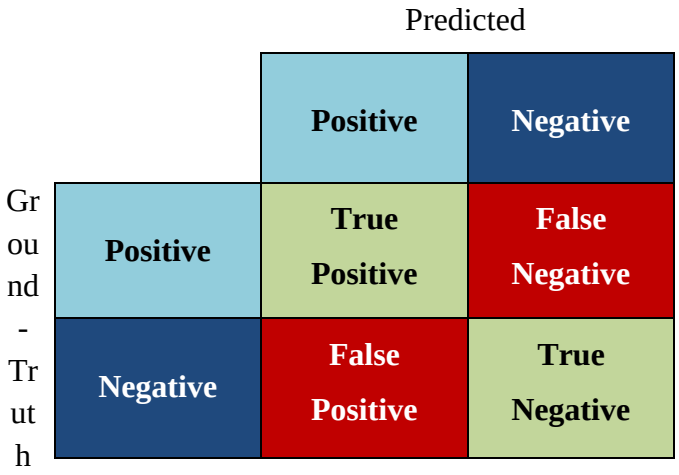


Figure6.1.5: Confusion matrix structure.

In that case we're only show the custom CNN model confusion matrix. Here the confusion matrix

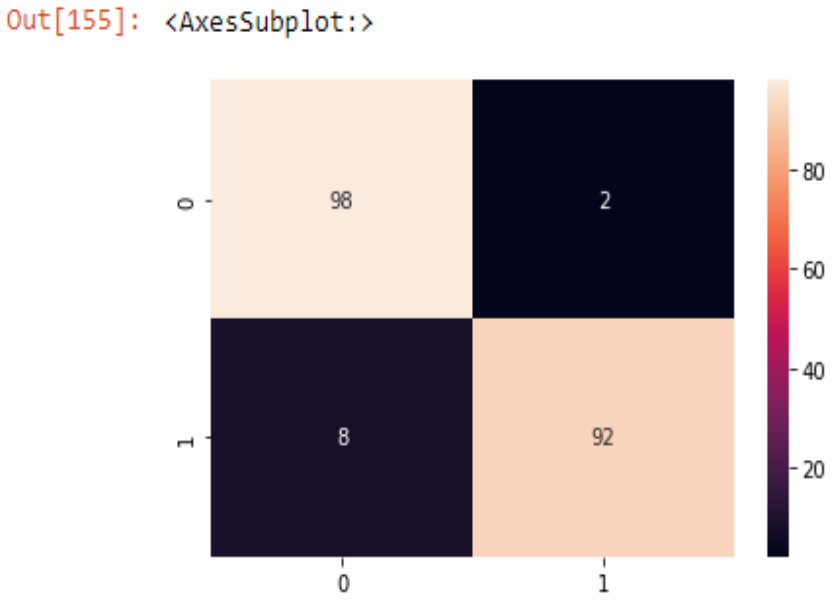


Figure6.1.6: Confusion matrix of custom cnn model.

As we've seen, the confusion matrix provides four distinct and distinct measures. Other metrics, including as accuracy, precision, and recall, may be generated based on these four metrics to provide more information about how the model acts. Accuracy is a statistic that sums up how well a model performs across all classes. It's helpful when all of the classes are equally important. The ratio between the number of right guesses and the total number of forecasts is used to compute it. The result of dividing the sum of *True Positives* and *True Negatives* over the sum of all values in the matrix. Here the true positives and true negatives are divided into all of confusion matrix sum. The result is 0.95, indicating that the model is 95% accurate in predicting the outcome.

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{True_{positive} + True_{negative}}}{\boldsymbol{True_{positive} + True_{negative} + False_{positive} + False_{negative}}}$$

The number of properly recognized Positive samples divided by the total number of Positive samples is used to calculate the accuracy (either correctly or incorrectly). The precision of the model in categorizing a sample as positive is measured.

$$\mathbf{Precision} = \frac{\mathbf{True_{positive}}}{\mathbf{True_{positive} + False_{positive}}}$$

When the model makes a large number of inaccurate Positive classifications or a small number of correct Positive classifications, the denominator rises and the accuracy falls. Two Positive samples were accurately identified, however one Negative sample was mistakenly labeled as Positive. The accuracy is 98/(98+8)=0.9245, meaning the True Positive rate is 98 and the False Positive rate is 8. In other words, the model's trustworthiness percentage when a sample is Positive is 92.45 percent. True positive is divided by using sum of two confusion matrix value one is true positive and other is false negative. The model's ability to recognize Positive samples is measured by the recall. The higher the recall, the greater the number of positive samples found.

$$\mathbf{Recall} = \frac{\mathbf{True_{positive}}}{\mathbf{True_{positive} + False_{negative}}}$$

The recall is only concerned with the classification of positive samples. This is true regardless of how negative samples are categorised, for example, for precision. Even if all the negative samples were mistakenly categorized as Positive, the recall will be 100% if the model labels all the positive samples as Positive. As a result, 98% is True positive. Because just one positive

sample is categorized as negative, the False Negative rate is two. As a consequence, the recall is 98/(98+2)=98/100=0.98, or 98 percent of the time. This assessment considers both precision and recall while calculating the score. The F1 score may be thought of as a weighted average of the accuracy and recall values, with the highest value being 1 and the poorest value being 0. It's a natural concept to use a combination of accuracy and recall. This is done by computing the harmonic mean of the F1 score. (2*precision*recall)/(precision+recall) = (2*92.45*98) /(92.45+98) = 94.91.

$$\textbf{F1 Score} = \frac{\mathbf{2} * (\textbf{Precision} * \textbf{Recall})}{\textbf{Precision} + \textbf{Recall}}$$

6.2 Output

After that saving custom cnn model in system storage. For check the result of fire and no fire we load the saved model. and then get each input of storage image let's see what's the result show-

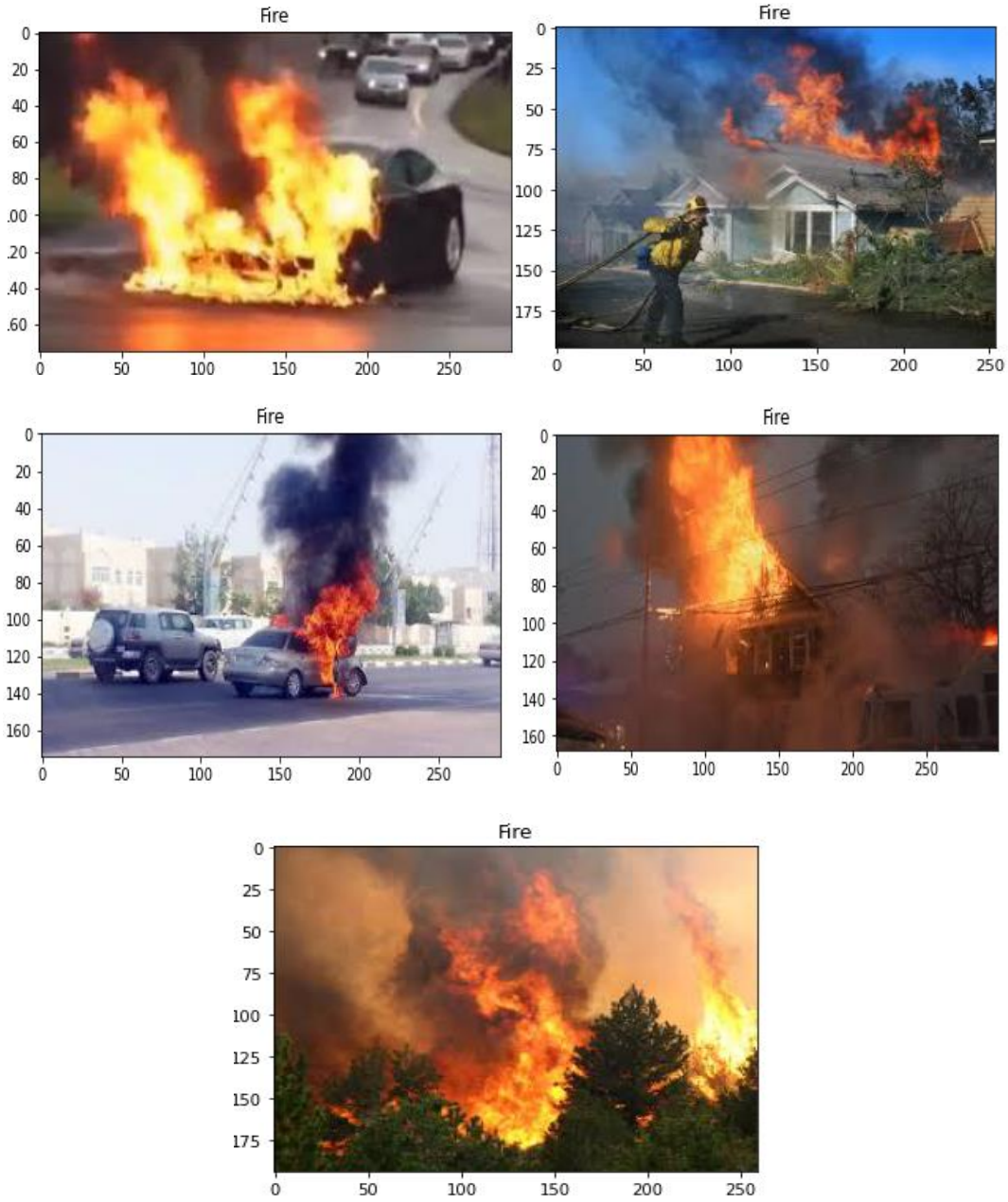


Figure 6.2.1: Classified fire images output.

Here some input images test which is successfully classified normal that means no fire detected. Let's see all the result-

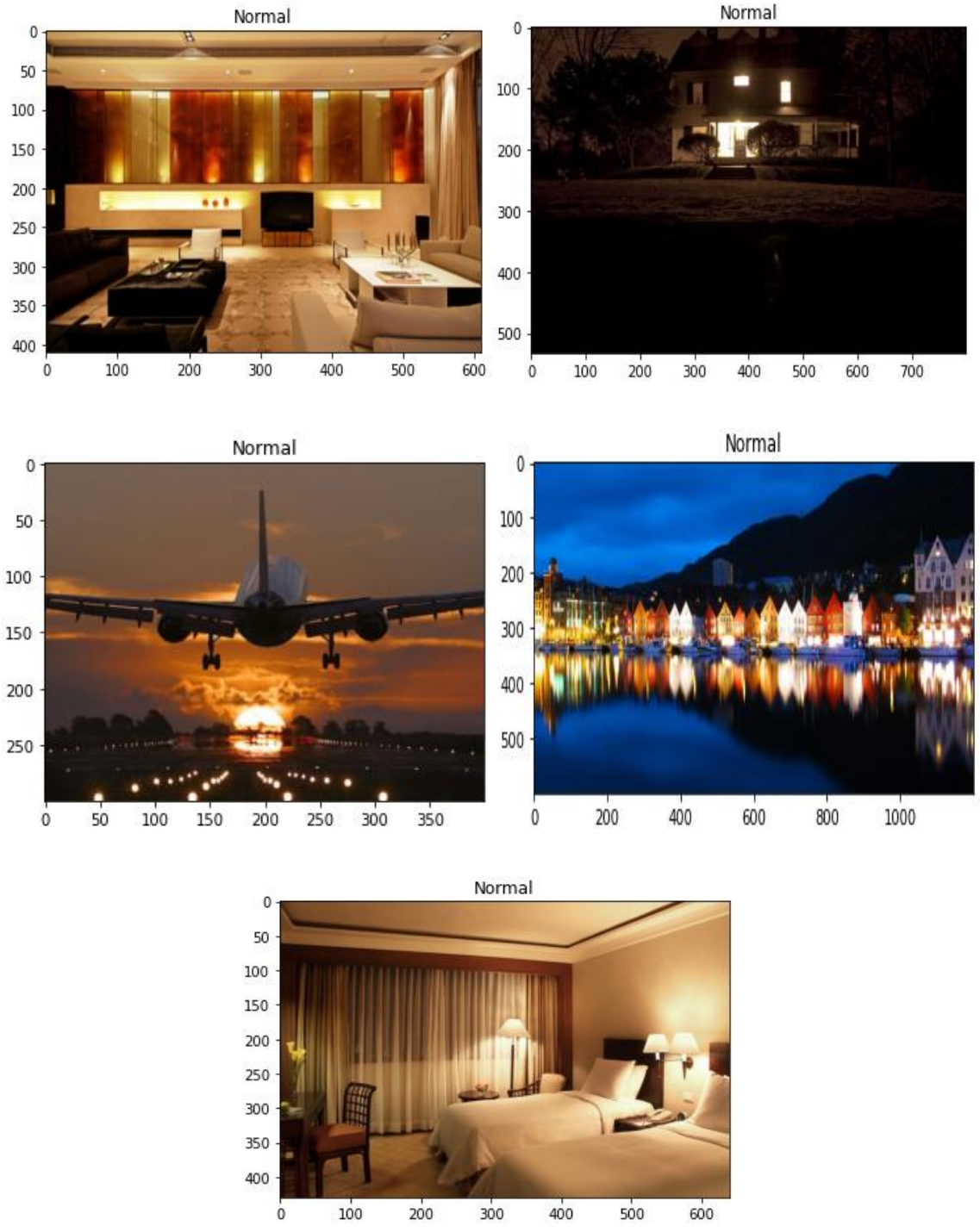


Figure 6.2.2: Classified normal images output.

Multi-class classification issues can have numerous class labels, however binary classification is the simplest and arguably most common sort of classification problem, with only two classes.

The vast majority of classification algorithms rely on the assumption that each observed class contains an equal number of instances. In truth, this isn't always the case, and imbalanced classification challenges are defined as datasets having a skewed class distribution. Most machine learning methods assume that a classifier's prediction errors, also known as miss-classifications, are the same. Here we showed the some sample of the missclassification.

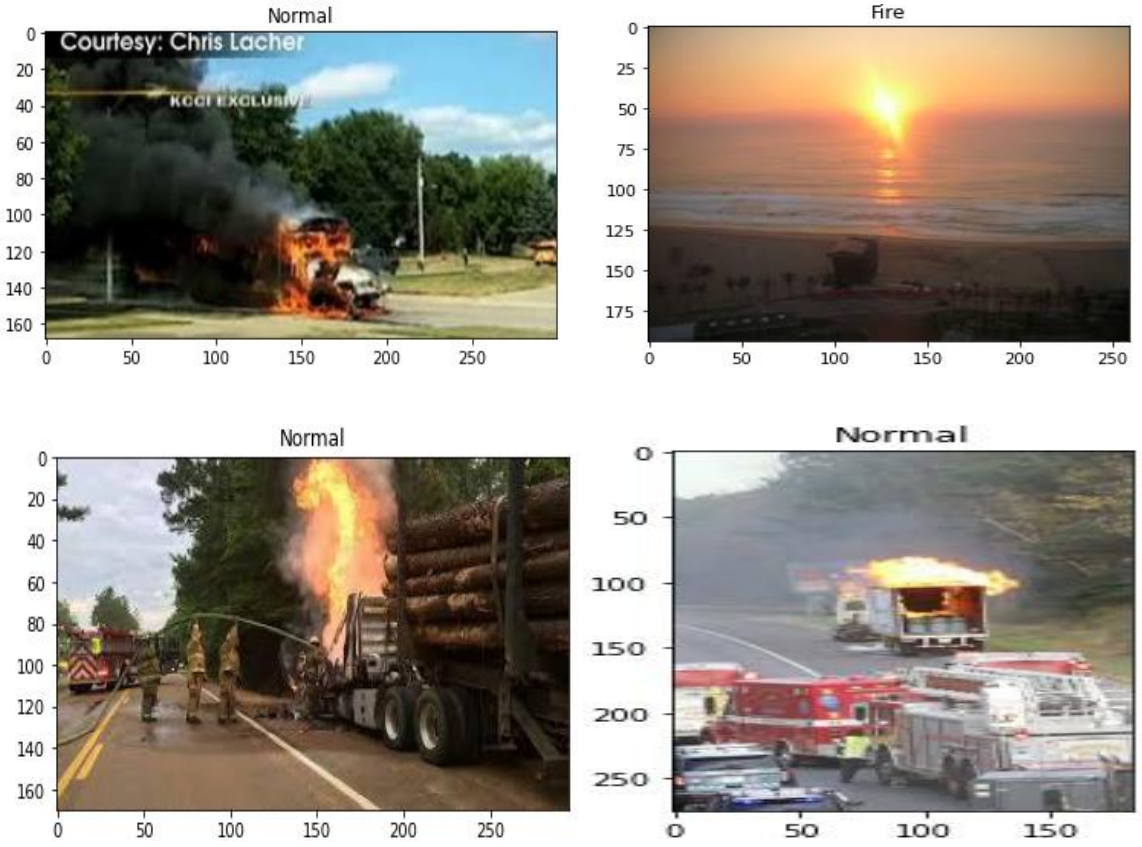


Figure 6.2.3: Miss Classified fire/normal images output.

Because instances from the majority class often represent a regular or no-event scenario, whereas examples from the minority class typically reflect an exceptional or event case, this negative vs. positive naming system was created. Each of the classification mistakes that can be produced in real-world unbalanced binary classification situations is usually interpreted differently.

6.3 Limitation

During the course of our investigation, we encountered a number of obstacles. First and foremost, we did not have access to EDU University's Thesis Lab owing to present circumstances, which would have supplied us with the higher computer capacity necessary for

the realization of our thesis. We had to design and execute advanced machine learning algorithms with extremely limited resources for the testing and training of our CNN models. We had a lot of trouble with the Alexnet model training and testing since it took a lot of CPU time, with each epoch taking a long time to complete. We also experienced kernel and system problems after a specific number of epochs. This hampered our ability to test the CNN models under various research situations, such as increasing the epoch, steps per epoch, and so on.

6.4 Comparison with other CNN based fire detection

This section compares and contrasts the proposed system's performance with three state-of-the-art models. The relevance of these models, as well as the year of release, was considered when they were chosen. The information about these approaches is summarized as follows:

1. Muhammad et al. [45] used AlexNet to identify early fires during monitoring for better disaster management. The authors employed pre-trained AlexNet (on ImageNet) that was fine-tuned by adjusting the fully connected layers and learning rate. Two datasets [54, 55] were used for the whole experimental investigation.

2. Detection of advanced objects New photo fire detection approaches are created using CNN models such as Faster-RCNN, R–FCN, SSD, and YOLO v3. According to a comparison of proposed and present methodology, object detection CNN-based fire detection algorithms outperform other methods. When employing YOLO v3, the algorithm's average accuracy is 83.7 percent [56].

Table 6.4.1 :Comparison results with other CNN based models

Model	Accuracy
AlexNet	91.87%
Yolo v3	83.7%
Proposed	98%

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis has made a lot of advances to the field of fire/no fire detection in outdoor contexts utilizing photographs. This study contributes to the field of image-based fire detection by detecting fire with transfer learning and learned features, creating a bespoke CNN model for fire detection, and presenting a model that combines fire prediction and detection. To begin, the current research on image-based fire detection and fire prediction systems was reported on and critically examined in this work. Due to the unpredictable fluctuations in color, shade, and density, feature extraction is one of the most difficult aspects of image-based fire detection. This research focused on the association between dynamic texture characteristics and color to increase the discriminative and representational capacity of the features. The suggested method's capabilities and performance were compared to other state-of-the-art approaches in this study. The suggested technique is shown to be successful in reliably detecting fire while displaying a well-balanced trade-off between accuracy and computing complexity, according to the comparative findings. Despite their high accuracy, handmade features have several restrictions, are computationally costly, and have the potential to overspecialize. This study was motivated by the outstanding performance of fine-tuned networks and was inspired by AlexNet to create a lightweight model. In comparison to current approaches, several studies shown that the suggested lightweight model is capable of detecting fire with high detection accuracy (98%) compared to the existing methods. In addition, results show that the proposed model yields higher accuracy than the pretrained networks, with smaller architecture that are memory efficient to be used in intelligent surveillance system. Finally, given the contributions listed above, this thesis contributes knowledge in the field of surveillance video-based smoke and fire detection. The findings, methodological progress, and experimental analyses of this thesis provide a comprehensive source of information which can be assimilated towards future research to design even more advanced smart and intelligent systems for fire detection.

7.2 Future work

Each of the investigations given in this thesis points to potential future directions. A number of possible paths are listed below:

1. We'll attempt to build a multiclass CNN architecture. Because there are several categories but only one is allocated to each instance, such difficulties are referred to as multiclass categorization. Such as rate of fire and smoke.

Figure 7.2.1: Fire detected. But it’s not harmful.

Figure 7.2.2: Fire detected. But it’s harmful.

2. Due to a shortage of validation datasets, this study suggested a proposal for an integrated model that has not been fully addressed or verified before. Further research might be focused on continuing to build the fire safety model that combines prediction and detection, as well as researching more effective ways of merging image and video with meteorological data and using deep learning algorithms.
3. Given the scarcity of smoke and fire picture datasets for training, Generative Adversarial Networks (GANs) might be used to supplement training data. Furthermore, the few-shot learning approach may be used to train CNNs from the ground up, allowing the network to perform effectively even after only viewing a few examples from each class.

References

[1] UNB., “Fire incidents four-fold to 285,000 in two decades”,2021.

[2] “Youth-burnt-to-death-in-Dhaka”, 2022.

[3] UNB., “Fire incidents in Bangladesh triple in 22 years”, 2018.

[4] F . Sustain., “ Fire Hazard in Chattogram City Corporation Area: A Critical Analysis of Its Causes and Mitigation Measures”, September 2021.

[5] P . Patel, S . Tiwari., “Flame Detection Using Image Processing Techniques. International Journal of computer application”(2012).

[6] J . Seebamrungsat, S. Praising, P . Riyamongkol., “Fire Detection in Buildings Using Image Processing”, (2014).

[7] K . Muhammad, J . Ahmad., “CNN based fire detection system and localization in video surveillance system”.

[8] Abdulazi, Y . I . CHO., “Deep Learning Algorithm for Fire and Smoke Detection with Limited Data”, IEE. (2018).

[9] Q . Zhang, J . Xu, L . Xu and H . Guo., “Deep Convolutional Neural Networks for Forest Fire Detection”, (2016).

[10] S . Frizzi1, R . Kaabi, M . Bouchouicha, J.M. Ginoux., “CNN for Video Fire and Smoke Detection”, IEEE (2017).

[11] M . Khan, J . Ahmad., “fire detection using convolutional neural networks during surveillance”, IEEE. (2017).

[12] D. H. Hubel, T. N. Wiesel., “Receptive fields and functional architecture of monkey”.

[13] K. Fukushima., “A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”, (1980).

[14] Y. LeCun et al., “Handwritten digit recognition with a back-propagation”, pp. 396-404. (1990).

[15] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner., “Gradient-based learning applied to document recognition”, (1998).

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton., “Imagenet classification with deep convolutional neural networks”, pp. 1097-1105, (2012).

[17] R. A. Minhas, A. Javed, A. Irtaza, M. T. Mahmood, and Y. B. Joo, AlexNet., “CNN Shot Classification of Field Sports Videos, Applied Sciences”,(2019).

[18] X. Han, Y. Zhong, L. Cao, and L. Zhang., “Pre-Trained AlexNet Architecture with For High Spatial Resolution Remote Sensing Image Scene Classification”, (2017).

[19] H. C. Shin et al., “Transfer Learning in Deep Convolutional Neural Networks for Computer-Aided Detection," IEEE,(2016).

[20] G. Amato, F. Carrara, F. Falchi, C. Gennaro, C. Meghini, and C. Vairo.., “Deep Convolutional Neural Networks for Computer-Aided Detection” ,(2017).

[21] R. F. Nogueira, R. d. A. Lotufo, and R. C. Machado., “Convolutional Neural Networks for Fingerprint Liveness Detection”, IEEE,2016.

[22] K. Simonyan and A. Zisserman., “very deep convolutional networks for large-scale image recognition”, (2015).

[23] C. Szegedy et al., “Going deeper with convolutions”, Cvpr. (2015).

[24] K. He, X. Zhang, S. Ren, and J. Sun., “Deep residual learning for image recognition “, IEEE. (2016).

[25] J. Gu et al., “Recent advances in convolutional neural networks”, Pattern Recognition. (2017).

[26] Z. Wu, C. Shen, and A. van den Hengel., “Wider or Deeper: Revisiting the ResNet Model for Visual Recognition”, Pattern Recognition. (2019).

[27] H. Jung, M.-K. Choi, J. Jung, J.-H. Lee, S. Kwon, and W. Young Jung., “Vehicle categorization and localisation in traffic surveillance systems using ResNet, IEEE. (2017).

[28] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger., “Densely connected convolutional networks”, IEEE. (2017).

[29] G. Larsson, M. Maire, and G. Shakhnarovich., ”Ultra-deep neural networks without residuals”, (2016).

[30] F. Chollet., “Deep learning with depthwise separable convolutions”. IEEE Conference.

[31] R. Girshick, J. Donahue, T. Darrell, and J. Malik., “For accurate object recognition and semantic segmentation”,IEEE conference. (2014).

[32] R. Girshick., “Fast r-cnn”, IEEE international conference. (2014).

[33] S. Ren, K. He, R. Girshick, and J. Sun., “Faster r-cnn: Using area proposal networks for real-time object detection" , (2015).

[34] L. Zhang, L. Lin, X. Liang, and K. He., “Is the faster R-CNN effective in detecting pedestrians?”, European Conference on Computer Vision. (2016).

[35] Z.-Q. Zhao, H. Bian, D. Hu, W. Cheng, and H. Glotin., “Fast R-CNN and Batch Normalization for Pedestrian Detection”, International Conference on Intelligent Computing, (2017).

[36] J. H. Kim, G. Batchuluun, and K. R. Park., “Pedestrian identification in the dark using a faster R-CNN and combining deep neural features from many pictures”, (2018).

[37] Q.-x. Zhang, G.-h. Lin, Y.-m. Zhang, G. Xu, and J.-j. Wang., “Wildland Forest Fire Smoke Detection Using Synthetic Smoke Images and a Faster R-CNN”, (2018).

[38] Y.-J. Kim and E.-G. Kim., “A Fire Detection Study with Faster R-CNN and ResNet," International Information Institute (Tokyo),(2018).

[39] J. Hung and A. Carpenter., “Using a Faster R-CNN to Detect Objects in Malaria Images”, IEEE Conference,(2017).

[40] X. Sun, P. Wu, and S. C. Hoi., “Deep learning for face recognition: A quicker RCNN technique, Neurocomputing”,(2018).

[41] A. A. Yilmaz, M. S. Guzel, I. Askerbeyli, and E. Bostanci., “Using Deep Learning Methodologies for Vehicle Detection”, (2018).

[42] L. Suhao, L. Jinzhao, L. Guoquan, B. Tong, W. Huiqian, and P. Yu., (2018)," Deep learning-based vehicle type recognition in a traffic situation", (2018).

[43] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman., "The visual object classes (voc) of Pascal provide a hurdle", International journal of computer vision. (2010).

[44] O. Russakovsky et al., "The Imagenet Visual Recognition Task is a large-scale visual recognition challenge". International Journal of Computer Vision,(2015).

[45] T.-Y. Lin et al., "Microsoft coco: Common objects in context", European conference, (2014).

[46] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson., "How transferrable are deep neural network features?", (2014).

[47] K. Nogueira, O. A. Penatti, and J. A. dos Santos., "Towards a deeper understanding of convolutional neural networks for scene categorization in remote sensing",(2017).

[48] Z. Yin, B. Wan, F. Yuan, X. Xia, and J. Shi., "Image Smoke Detection Using Deep Normalization and a Convolutional Neural Network", IEEE, (2014).

[49] K. Muhammad, J. Ahmad, I. Mehmood, S. Rho, and S. W. Baik., "Fire Detection in Surveillance Videos Using Convolutional Neural Networks", IEEE,(2018).

[50] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson., "How transferrable are deep neural network features?",(2014).

[51] K. Nogueira, O. A. Penatti, and J. A. dos Santos., "Towards a deeper understanding of convolutional neural networks for scene categorization in remote sensing",(2017).

[52] Y.-L. Boureau, J. Ponce, and Y. LeCun., "Feature pooling in visual recognition: a theoretical study, International conference (ICML)", (2010).

[53] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng., "Convolutional neural networks for end-to-end text recognition", International Conference IEEE, (2012).

[54] J. Gu et al., "Convolutional neural networks have made significant progress recently", (2018).

[55] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov., "Dropout is a simple method for preventing neural networks from being overfit", Journal of ML.(2014).

[56] S. Ioffe and C. Szegedy., “Batch normalization: Reduces internal covariate shift to speed up deep network training”, (2015).

[57] K. He, X. Zhang, S. Ren, and J. Sun., “Exploring rectifiers in depth: Exceeding human performance on imagenet classification”, IEEE international conference,(2015).

[58] P. Foggia, A. Saggese, and M. Vento., “Fire Detection in Real Time for Video Surveillance Applications Using a Combination of Experts Based on Color, Shape, and Motion”, IEEE, (2015).

[59] D. Y. T. Chino, L. P. S. Avalhais, J. F. Rodrigues, and A. J. M. Traina., “Integrating Pixel Color and Texture Analysis to Detect Fire in Still Images”, Conference on Graphics.(2015).

[60] Pu Li, Wangda Zhao., “Convolutional neural networks-based image fire detection algorithms Civil Engineering School”.