B.ÖZEL

ATILIM UNIVERSITY 2024

FOREST FIRE DETECTION USING DEEP LEARNING

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

BERK ÖZEL

A MASTER OF SCIENCE THESIS
IN
THE DEPARTMENT OF MECHATRONICS ENGINEERING

APRIL 2024

FOREST FIRE DETECTION USING DEEP LEARNING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

BY

BERK ÖZEL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF MECHATRONICS ENGINEERING

APRIL 2024

Approval of the Graduate School of Natural and Applied Sciences, Atılım University.

_____

Prof. Dr. Ender KESKİNKILIÇ
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Master of Science Engineering in Mechatronics Atılım University.**

_____

Prof. Dr. Hulusi Bülent ERTAN
Head of Department

This is to certify that we have read the thesis Forest Fire Detection Using Deep Learning submitted by Berk Özel and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Assoc. Prof. Dr. Muhammad Umer KHAN
Supervisor

Examining Committee Members:

Asst. Prof. Dr. Babek NASERİ
Mechatronics Eng. Department,
Atılım University _____

Assoc. Prof. Dr. Muhammad Umer KHAN
Mechatronics Eng. Department,
Atılım University _____

Asst. Prof. Dr. Bülent İRFANOĞLU
Electrical and Electronics Eng. Department,
Başkent University _____

Date: *22/04/2024*

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

Name, Last Name : Berk ÖZEL

Signature :

ABSTRACT

FOREST FIRE DETECTION USING DEEP LEARNING

ÖZEL, Berk

M.S., Department of Mechatronics Engineering

Supervisor: Assoc. Prof. Dr. M. Umer Khan

April 2024, 45 pages

Fire detection systems are critical for safeguarding lives and minimizing property damage. One of the key areas where such systems are vital are forest fires. In recent years, there have been several record-breaking forest fires in terms of size, duration, and destruction. Traditional methods of fire detection, such as smoke or heat sensors, have their limitations, leading to the emergence of innovative approaches based on advanced technologies. This thesis examines the application of Batch-Instance Normalization combined with ResNet, a deep learning model for wildfire detection. The study compares the performance of Batch-Instance Normalization with other normalization approaches. In this study, a forest fire dataset is used which is taken from the Kaggle for training the model. The Dataset includes 4609 images, 2120 Fire and 2499 Non-Fire images. The ResNet model is tested with eight different optimizers and trained with the one that gives the best results. The experiments evaluate the impact of normalization techniques and optimizers on the accuracy of wildfire detection. The results show that Batch-Instance Normalization with single exponential smoothing significantly improves the accuracy of the model. It attains F1-score of 96.14%, accuracy of 96.56%, and precision of 99.49%. A minimum 1%, accuracy difference, %0.6 F1 score difference, %1.05 precision difference were obtained from other normalization methods. Combining the capabilities of deep learning with the innovative Batch-Instance Normalization has demonstrated a promising and effective solution for wildfire detection.

Keywords: fire; forest fire; deep learning; artificial intelligence; imaging system; wildfire; normalization; batch; instance; batch-instance

ÖZ

DERIN ÖĞRENME KULLANARAK ORMAN YANGINI TESPITI

ÖZEL, Berk

Department of Mechatronics Engineering

Danışman: Doç. Dr. M. Umer Khan

Nisan 2024, 45 Sayfa

Yangın algılama sistemleri can güvenliği ve maddi hasarın en aza indirilmesi açısından kritik öneme sahiptir. Bu tür sistemlerin hayati önem taşıdığı alanlardan biri de orman yangınlarıdır. Son yıllarda büyüklük, süre ve tahribat açısından rekor sayıda orman yangını yaşandı. Duman veya ısı sensörleri gibi geleneksel yangın algılama yöntemlerinin sınırlamaları vardır ve bu da ileri teknolojilere dayalı yenilikçi yaklaşımların ortaya çıkmasına neden olur. Bu tez, orman yangını tespiti için bir derin öğrenme modeli olan ResNet ile birlikte Batch-Instance Normalizasyonunun uygulanmasını incelemektedir. Çalışma, Batch-Instance Normalizasyonunun performansını diğer normalleştirme yaklaşımlarıyla karşılaştırmaktadır. Bu çalışmada modelin eğitimi için orman yangını veri seti kullanılmıştır. Bu veri seti 4609 görsel içermektedir. Bu görseller 2120 Yangın, 2499 yangın içermeyen görselden oluşmaktadır. ResNet modeli sekiz farklı optimize edici ile test edilmiş ve en iyi sonuçları veren ile eğitilmiştir. Deneyler, normalizasyon tekniklerinin ve optimize edicilerin yangın tespitinin doğruluğu üzerindeki etkisini değerlendirmektedir. Sonuçlar, tek üstel düzeltmeyle Batch-Instance Normalizasyonunun modelin doğruluğunu önemli ölçüde artırdığını göstermektedir. Deneyde model, 96.14% F1 skoruna, 96.56% doğruluğa ve 99.49% kesinlik değerlerine ulaşmıştır. Diğer yaklaşımlardan minimum %1 doğruluk farkı, %0,6 F1 skor farkı, %1,05 kesinlik farkı elde edilmiştir. Derin öğrenmenin yeteneklerini Batch-Instance Normalizasyonunuyla birleştirmek, orman yangını tespiti için umut verici ve etkili bir çözüm ortaya koydu.

Anahtar Kelimeler: Yangın; Orman Yangını; Derin Öğrenme; Yapay Zeka; Görüntüleme sistemi; Normalizasyon; Batch; Instance; Batch-Instance

To my wife and my parents

ACKNOWLEDGEMENTS

I would like to thank my mentor and supervisor Associate Professor Dr. Muhammad Umer Khan for his advice, constant support and motivating me every time I felt unmotivated.

I would also like to thank my wife and my parents who supported me under all circumstances.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF ABBREVIATIONS

CV	Computer Vision
NN	Neural Network
DL	Deep Learning
AI	Artificial Intelligence
ML	Machine Learning
BN	Batch Normalization
IN	Instance Normalization
LN	Layer Normalization
GN	Group Normalization
IP	Image Processing
BIN	Batch-Instance Normalization

CHAPTER 1

INTRODUCTION

Forests cover approximately 4 billion hectares of the world's landmass, roughly equivalent to 30% of the total land [1]. The preservation of forests is essential for maintaining biodiversity on a global scale. Wildfires are destructive events that could adversely change the balance of our planet and threaten our future [2]. Wildfires have long-term devastating effects on ecosystems, such as destroying vegetation dynamics, greenhouse gas emissions, loss of wildlife habitat, and destruction of land covers. In 2021, 9 million hectares of land, or the area of Portugal in Europe were lost. 7.8 million hectares were in the United States, Canada, and Russia. In Italy, 723,924 hectares were destroyed in the preceding 14 years, and 159,437 hectares of forest were destroyed by fire in 2021 alone [3]. In 2022, 4,392 hectares of forest were lost in Turkey [4]. These numbers demonstrate a worrying pattern of increasing annual exploitation of the natural resources of our planet. Today, many traditional fire detection methods are used to prevent wildfires, but these methods are insufficient to convey timely and accurate information. In this regard, the latest technology needs to be investigated for more effective possible solutions. This research focuses on combining the capabilities of deep learning with the innovative Batch-Instance Normalization technique to increase the accuracy and speed of forest fire detection systems. The emergence of deep learning has revolutionized the field of computer vision. The focus of this revolution is that it offers unique capabilities in image and pattern recognition [5]. Models inspired by neural networks in the human brain demonstrate a significant ability to distinguish complex structures and differences. Using these capabilities in forest fire detection promises significant success. Combining the advantages of batch and instance norms, it is used to speed up the training phases of deep neural networks and increase the accuracy rate by improving the model's ability to handle

changes in input distributions and improving generalization and robustness, which is one of the key points of this research [6]. The integration of the Batch-Instance Normalization approach into the Deep learning framework for forest fire detection aims to improve the detection capabilities of the model. This thesis discusses the integration of the Batch-Instance Normalization method for forest fire detection into one of the deep networks in the literature, its application and comparison with other normalization approaches. The research aims to contribute to the development processes of smart fire detection systems.

1.1 Motivation

The motivation behind this article is that the high-accuracy classification ability offered by deep learning and the features of improving the generalization and robustness of the model promised by Batch-Instance Normalization offer significant potential.

1.2 Problem Statement

The threat of forest fires in the world is increasing day by day and cannot be prevented. While many solutions to this problem are promising, they struggle with accuracy and adaptability to dynamic fire datasets. The problem is developing accurate deep learning models for wildfire detection by addressing challenges related to feature scaling and optimization.

1.3 Contributions

In order to determine the forest fire under real-life conditions, an uncommon Batch-Instance Normalization method is employed with ResNet. Other normalization methods were applied and compared with the Batch-Instance Normalization. To achieve the best accuracy, eight different optimizers were employed and compared and different versions of ResNet were implemented and compared. In addition to the Batch-Instance Normalization method, double exponential smoothing approach is implemented and compared. A systematic review is performed to provide insights into the potential and future directions of fire detection and extinguishing using image processing, computer vision, and deep learning.

1.4 Organization of the Thesis

The proposed study consists of five chapters: Introduction, Literature Review, Methodology, Experimental Results, and Conclusion.

Chapter 1 (Introduction): In the Introduction, it is mentioned that forest fires are a worldwide problem. The consequences of this problem and the extent of the damage caused are highlighted. Instead of traditional methods used to solve this problem, more innovative and applicable solutions are offered. In continuation, the motivation of this thesis, the problem statement, and the contributions are mentioned.

Chapter 2 (Literature Review): While conducting the literature review, the research approach and the keywords used were mentioned. The Literature review covers the approaches used to date for fire detection. The details of these approaches are mentioned. Normalization approaches commonly used today are explained with their equations.

Chapter 3 (Methodology): In Methodology, details of the deep learning model (ResNet) , details of the Batch-Instance Normalization method and relevant formulas are discussed. Single exponential smoothing, which is a statistical method and a part of this approach, is mentioned. In addition, the double exponential smoothing method that has been tried with this normalization method is presented.

Chapter 4 (Experimental Results): In Experimental Results, the forest fire dataset, data augmentation, optimizers used, comparison criteria, experimental conditions are mentioned. Experiment are shown with different graphs. At the end of the chapter, the results were evaluated in the discussion.

Chapter 5 (Conclusion): In this conclusion section, the description of the findings of the study, the results and analysis of the experiments are presented. It also highlighted the important contributions and potential of the study conducted to the field. The future work of this study is mentioned.

CHAPTER 2

LITERATURE REVIEW

2.1 Fire Detection

Early detection and rapid extinguishing of fires are crucial in minimizing the loss of life and property [7]. Traditional fire detection systems that rely on smoke or heat detectors suffer from low accuracy and long response time. However, advancements in image processing (IP), computer vision (CV), and deep learning (DL) have opened up new possibilities for more effective and efficient fire detection and extinguishing systems [8]. These systems utilize cameras and sophisticated algorithms to analyze visual data in real time, enabling early fire detection and efficient fire suppression strategies.

In most of the literature, researchers have mainly posed their problem under the paradigm of fire detection. But, some researchers have also explored different aspects of the phenomenon of combustion i.e., smoke, flame, and fire with the intent to effectively determine the threats due to fire. In summary, fire is the overall phenomenon of combustion involving the rapid oxidation of a fuel source, while flame represents the visible, gaseous part of a fire that emits light and heat. Smoke, on the other hand, is the collection of particles and gases released during a fire, which can be toxic and pose health hazards.

With the developments in computer technology, object recognition technology in the natural environment has become widespread in daily life. Many successful applications are used in many fields, such as handwritten character recognition, optical character recognition, fingerprint, voice, face and emotion recognition. Concrete objects can be distinguished and classified from an environment with a complex background. Animals, flowers, and natural events such as fire can be given as examples of these objects [9].

Image processing techniques enable the extraction of relevant features from images or

video streams captured by cameras [10]. This includes analyzing color, texture, and spatial information to identify potentially fire-related patterns [11]. By applying algorithms such as edge detection, segmentation, and object recognition, fire can be detected and differentiated from non-fire elements with a high degree of accuracy [12] [13].

Computer vision encompasses extracting image patterns, recording raw data, and interpreting information [14]. These fields include many concepts such as digital image processing, artificial intelligence and pattern recognition [15]. An important part of the concepts in computer vision is related to gaining knowledge by extracting features and information from input images. Computer vision can play a crucial role in early fire detection by leveraging image and video processing techniques to analyze visual data and identify signs of fire [16]. Computer Vision algorithms can identify patterns based on features such as color, shape, and motion [17] [18]. CV with thermal imaging technology can detect fires based on temperature variations [19] [20]. It is important to note that CV conjugated with other fire safety measures, such as smoke detectors, heat sensors, and human intervention, enhances early fire detection. DL combined with CV can also effectively recognize various fire characteristics, including flames, smoke patterns, and heat signatures [19] [21]. It enables more precise and reliable fire detection, even in challenging environments with variable lighting conditions or occlusions.

Deep learning, a branch of machine learning, is derived from artificial neural networks based on the communication and processing model between neurons in the human brain. It consists of two main categories: supervised learning and unsupervised learning. Data markers is an indispensable aspect for supervised learning. The relationship between markers and features of the training data is an important part of the training process, continuously correcting the bias to improve the predictive rate of learning [22] . Convolutional neural networks, deep stacked networks, recurrent neural networks are in the supervised learning category. There is no marker in unsupervised learning. Boltzmann machine and deep confidence networks are in the unsupervised learning category [23].

Deep learning has revolutionized the field of CV by enabling the training of highly complex and accurate models [24]. Deep learning models can be trained on vast amounts

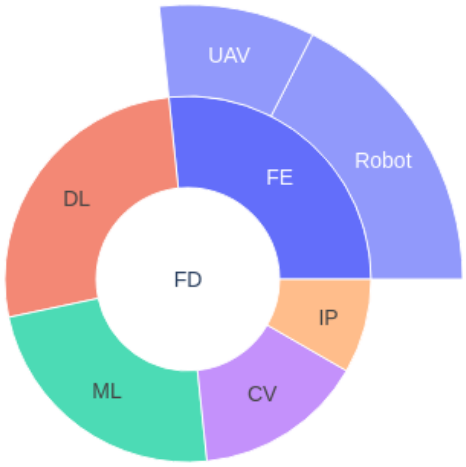
of labeled fire-related images and videos, learning to automatically extract relevant features and classify fire instances with remarkable precision [25] [26] [27]. These models can continuously improve their performance through iterative training, enhancing their ability to detect fires and reducing false alarms [28] [29].

While conducting the literature search, all aspects are tried to cover by contributing to the overall topic. Though these can be considered distinct research topics, from the perspective of deep learning, they play their part mutually.

Image Processing: Research that focuses on fire detection based on the features extracted after processing the image.

Computer Vision: Research focusing on the algorithms to understand and interpret the visual data to identify fire.

Deep Learning: Research associated with models that can continuously enhance their ability to detect fires.



IP: Image processing
CV: Computer vision
ML: Machine learning
DL: Deep learning
FD: Fire detection
FE: Fire extinguishing
UAV:Unmanned aerial vehicle

Figure 2.1: Search Strings

In literature search, different search strings are used for gathering the sources in this context. Search strings are shown in Figure 2.1. Based on the literature search, four main groups are formulated to classify the publication results. This classification is mainly based on the research topic, theme, title, and keywords. Each publication in our search falls broadly into one of these categories:

1. Fire: Research that addresses the methods capable of identifying forest fires in real-time or based on datasets.
2. Smoke: Research focusing on the methods to identify smoke with its different color variations.
3. Fire and Flame: Research associated with the methods that have the capability to identify fire as well as flame.
4. Fire and Smoke: Research that explores methods focusing on the accurate determination of fire and smoke.

Another category is defined which is application, it is a part of the other categories.

5. Applications: Research that addresses a robot's ability not only to detect fire but also extinguish it.

From the defined categories, Fire detection is the most dominant class containing almost 48 (49%) of the 97 total publications, followed by Smoke with 19 (20%), Fire and Smoke with 17 (18%), Applications with 9 (9%), and Fire and Flame with 5 (5%). The figures support the fact that image processing is considered the root field of other subsequent domains, whereas Applications with the least dominance indicate that on field utilization for the fire extinguishing is still very afar.

Deep learning has been successfully applied to fire, flame, and smoke detection tasks, leveraging its ability to learn complex patterns and features from large amounts of data. The primary task in fire detection is dataset collection which consists of a large dataset of images or videos containing both fire and non-fire scenes. These datasets can be obtained from various sources, including public databases, surveillance cameras, or generated synthetically. Pre-processing of collected data may be used to ensure consistency and quality. This preprocessing may include resizing images, normalizing pixel values, removing noise. Additionally, rotation, scaling, translation can be used for expanding the dataset. Afterward, a deep learning model needs to be designed and trained to perform fire, smoke, or flame detection. CNNs are commonly used for this purpose due to their effectiveness in image-processing tasks. The architecture can be customized based on the specific requirements and complexity of the detection task.

According to the literature survey about fire, flame, smoke detection, we decided to work on forest fire detection with deep learning. About training a deep learning model, we are interested in the effects of normalization techniques.

2.2 Normalization

Normalization is a preprocessing technique used in machine learning that scales and transforms a dataset's features to a similar range. There are various reasons why this process is crucial, which are outlined below.

Equal Treatment:

Normalization ensures that every feature is trained with the same consideration throughout. Larger sizes or variability in features could be dominant in the learning process. It can cause biased models. By normalizing characteristics, one may make sure that every feature makes a proportionate contribution to the learning process [30].

Improving Convergence:

Several gradient descent-based machine learning methods converge more quickly when features are normalized. The ability of normalization to resize the data to a smaller range is the cause of this behavior, since it helps to prevent oscillations or divergences in updates to the model parameters [31].

Enhancing Interpretability:

The process of normalization simplifies the interpretation of model parameters. Interpreting the relative importance of individual elements in the model gets difficult when features are on different scales. By bringing features to a same scale, making it easier to understand their relative contributions [32].

Handling Numerical Stability:

Large input values may cause numerical instability, particularly in algorithms involving computations like division or exponentiation. By normalizing the input features, these problems can be reduced and the algorithm's numerical stability can be increased [33].

Avoiding Model Overfitting:

In machine learning models, normalization can help to prevent overfitting. If features are not normalized, the model may become overly dependent on the training set and fail to generalize well to unseen data. Normalization encourages the model to focus on the

underlying patterns in the data rather than on the specific scales or ranges of the features [34].

Facilitating Distance-Based Algorithms:

The scale of features can affect algorithms that rely on distance measurements, like support vector machines (SVM) and k-nearest neighbors (KNN). By normalizing features, one can make sure that larger-scale features don't dominate distance estimates [35].

Outliers:

In certain cases, normalization might be useful in handling data outliers. By bringing all features to the same scale, outliers can lessen their impact on the model's overall behavior [36].

In summary, explaining the need for the normalization process in machine learning involves highlighting its role in ensuring fair treatment of features, improving convergence, enhancing interpretability, maintaining numerical stability, preventing overfitting, facilitating distance-based algorithms, and dealing with outliers [37]. In machine learning, there are several different normalization methods. These normalization methods are Batch, Instance, Group, Layer, Batch-Instance Normalizations.

2.2.1 Batch Normalization:

In deep learning, batch normalization is a method used to normalize each layer's activations across the batch dimension. Neural network training is stabilized and accelerated with the help of this normalization [38].

Normalization Operation:

The batch normalization operation is defined as follows x is an input tensor of shape (N,C,H,W), where N is the batch size, C is the number of channels, H is the height, and W is the width:

Compute the mean μ and variance σ^2 of the input tensor across the batch dimension:

$$\mu_B = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad (1)$$

$$\sigma_B^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_B)^2$$

(2)

Normalize the input tensor using the mean and variance:

$$\bar{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

(3)

Scale and shift the normalized tensor using learnable parameters γ (scale) and β (shift):

$$y_i = \gamma \bar{x}_i + \beta$$

(4)

where ϵ is a small constant (usually 10^{-5}) added for numerical stability, and yi represents the output of the batch normalization operation [39].

Training and Inference:

The variance and mean are calculated for the current mini-batch during training. On the other hand, normalization during inference is achieved by using a running average of the mean and variance over several mini-batches or the full training dataset.

Batch normalization has become a widely adopted technique in deep learning due to its effectiveness in stabilizing training, accelerating convergence, and improving the generalization ability of neural networks.

2.2.2 Layer Normalization:

Another method in deep learning for normalizing each layer's activations across the feature dimension is called layer normalization [40].

Normalization Operation:

The layer normalization operation can be defined as follows x is a input tensor of shape (N,D), where N is the batch size and D is the number of features:

Compute the mean μ and standard deviation σ of the input tensor across the feature dimension:

$$\mu = \frac{1}{D}\sum_{i=1}^{D} x_i \quad (5)$$

$$\sigma = \sqrt{\frac{1}{D}\sum_{i=1}^{D}(x_i - \mu)^2 + \epsilon} \quad (6)$$

Normalize the input tensor using the mean and standard deviation:

$$\bar{x}_i = \frac{x_i - \mu}{\sigma} \quad (7)$$

Scale and shift the normalized tensor using learnable parameters γ (scale) and β (shift):

$$y_i = \gamma \bar{x}_i + \beta \quad (8)$$

where ϵ is a small constant (usually 10^{-5}) added for numerical stability, and yi represents the output of the batch normalization operation.

Training and Inference:

For every sample in the batch, the mean and standard deviation are calculated across the feature dimension during training, just like in batch normalization. On the other hand, normalization during inference is done using a running average of the mean and standard deviation over several samples or the complete training dataset.

An efficient method for leveling activations inside neural network layers, layer normalization has grown in favor, particularly in recurrent neural networks (RNNs) and transformers. It enhances the capacity of deep learning models to generalize, speeds up convergence, and stabilizes training [40].

2.2.3 Instance Normalization:

A deep learning approach called instance normalization is used to standardize each instance's (sample's) activations across the feature dimension [41].

Normalization Operation:

The instance normalization operation is described as follows x is an input tensor of shape (N,C,H,W), where N is the batch size, C is the number of channels, H is the height, and W is the width:

Compute the mean μ_i and standard deviation σ_i across the spatial dimensions (height and width) for channel:

$$\mu_i = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{i,c,h,w} \quad (9)$$

$$\sigma_i = \sqrt{\frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{i,c,h,w} - \mu_i)^2 + \epsilon} \quad (10)$$

Normalize the activations of each channel for the sample x_i using its own mean and standard deviation:

$$\bar{x}_{i,c,h,w} = \frac{x_{i,c,h,w} - \mu_i}{\sigma_i} \quad (11)$$

where ϵ is a small constant (usually 10^{-5}) added for numerical stability.

Training and Inference:

The mean and standard deviation are calculated individually for every instance across the feature dimension during training. Normalization is done during inference using a running

average of the mean and standard deviation across instances or the full training dataset [42].

As a popular approach for normalizing activations within neural network layers, instance normalization has gained appeal, especially for tasks like image processing and style transfer. It enhances the capacity of deep learning models to generalize, speeds up convergence, and stabilizes training.

2.2.4 Group Normalization:

In group normalization, channels are divided into groups. Mean and variance operations are performed for each group [43]. In deep learning, group normalization is a technique that is used to equalize each layer's activations across channel groups.

Normalization Operation:

The group normalization is described as follows x is an input tensor of shape (N,C,H,W), where N is the batch size, C is the number of channels, H is the height, and W is the width: Divide the channels into G groups. Let C = G x group size.

Compute the mean μ_g and standard deviation σ_g across the spatial for each group g and each sample:

$$\mu_g = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{n,g,h,w} \quad (12)$$

$$\sigma_g = \sqrt{\frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{n,g,h,w} - \mu_g)^2 + \epsilon} \quad (13)$$

Normalize the activations of each channel for the sample n using its own mean and standard deviation:

$$\bar{x}_{n,g,h,w} = \frac{x_{n,g,h,w} - \mu_g}{\sigma_g} \quad (14)$$

where ϵ is a small constant (usually 10^{-5}) added for numerical stability.

Training and Inference:

The mean and standard deviation are calculated for every sample in the batch for every group during training. The running statistics are utilized to normalize data during inference [44].

It has been demonstrated that group normalization is a flexible method that works well for a variety of deep learning applications, particularly when batch sizes are small or irregular. It supports training stabilization, enhances convergence, and strengthens deep learning models' capacity for generalization.

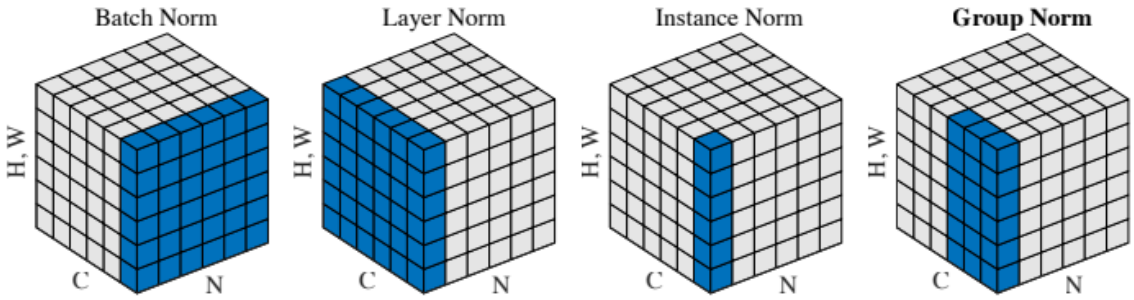


Figure 2.2: Normalization Methods

In Figure 2.2, Batch Normalization, Layer Normalization, Instance Normalization, Group Normalization are shown. Each subplot shows a feature map tensor. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

CHAPTER 3

METHODOLOGY

According to our literature survey, fire detection with deep learning subject is selected to study Res-Net, the deep learning model focused on in this section, will be explained in detail with the formulas of the normalization methods we apply on ResNet. Normalization methods which we selected to follow are batch, instance and Batch-Instance Normalization. These concepts will be explained under subheadings.

3.1 Res-Net

Many improvements have been made by using deep convolutional neural networks for image classification. These networks have produced state-of-the-art outcomes for image classification and recognition. One of these state of art networks is ResNet. ResNet is a type of neural network was introduced by Kaiming She, Xiangyu Zhang, Shaoqing Ren, and Jian. This neural network aims to ease the training of deep networks [45]. The ResNet network is based on the VGG-19 architecture. ResNet has a 34-layer flat network topology that is simpler and requires fewer filters than VGG networks. The architecture is then changed into a residual network by including shortcut connections in this flat network. In Figure 3.1 VGG19 model, a flat network and a residiual network are shown [46]. The model has fewer filters and lower complexity than VGG networks. Figure 3.1 shows a model with a depth of 34, but ResNet has versions with depths of 18, 34, 50, 101, and 152.

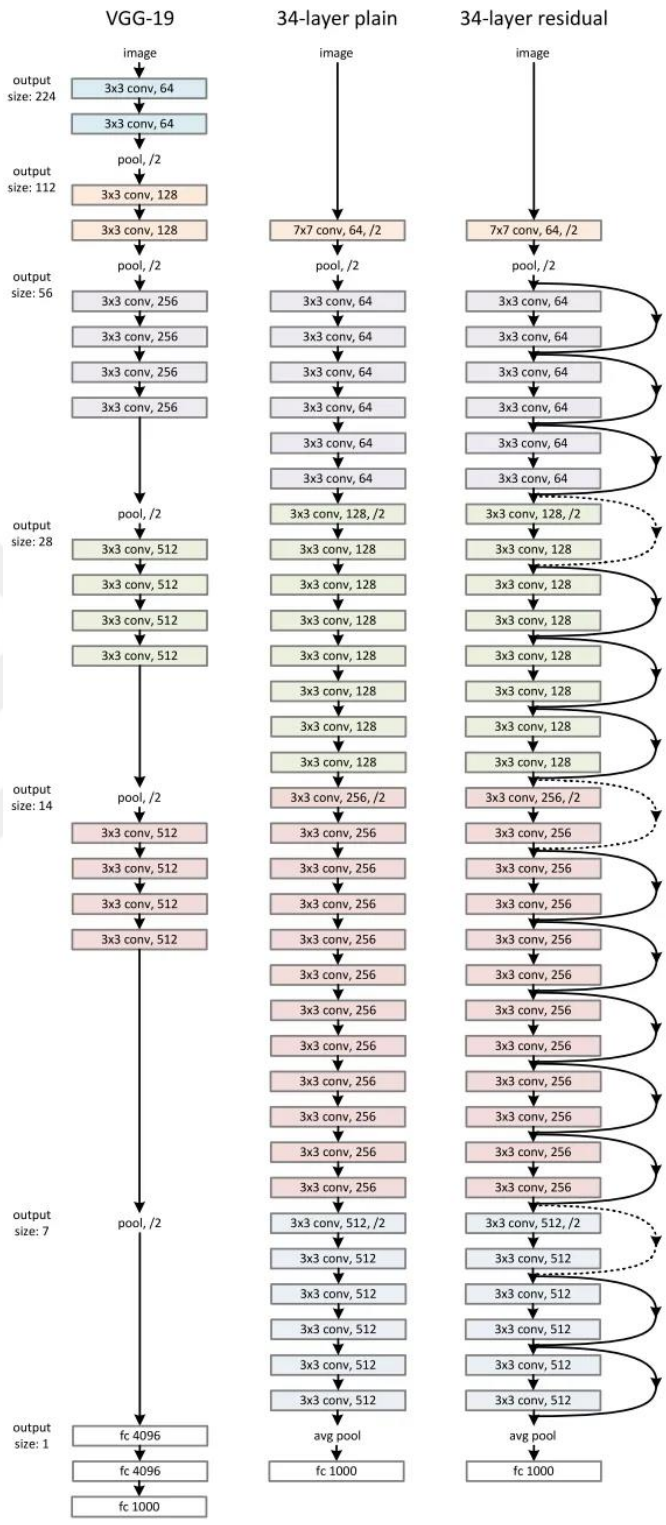


Figure 3.1: VGG-19 model, a plain network, a residual network

Training extremely deep neural networks is a task that ResNet addresses. The fundamental innovation of this approach is residual learning, in which the network acquires the ability to predict residual mappings rather than the output directly. Skip connections help to deal with the vanishing gradient problem that deep networks face by enabling the network to learn the residual between the input and output [47]. The residual block is the fundamental component of ResNet and consists of two convolutional layers with Batch Normalization and ReLU activation algorithms. An identity mapping is produced by the skip connection, which adds the block's input to the output. When necessary, a 1x1 convolutional layer modifies dimensions to correspond. ResNet creates deeper architectures represented by the total number of layers by stacking these blocks. ResNet-50, for example, has fifty layers. After convolutional layers, feature maps are reduced to vectors by global average pooling, which is followed by fully connected layers and a softmax classifier. Stochastic gradient descent with mini-batches is commonly used in ResNet, together with optimization strategies like as weight decay and learning rate schedules. With state-of-the-art performance on a variety of applications and datasets, ResNet's capacity to train very deep networks efficiently has made it an essential tool in computer vision [45].

3.2 Batch-Instance Normalization

In order to take full advantage on the benefits of both approaches, Batch-Instance Normalization, or BIN, is a strategy that combines batch normalization (BN) and instance normalization (IN). It was introduced to increase generalization performance and address some of the shortcomings of batch normalization, particularly in small batch sizes settings [6].

Activations are normalized throughout the batch dimension using batch normalization. Each sample in the batch is normalized using the mean and standard deviation that are computed for the entire batch. Activations are normalized across the channel dimension using instance normalization. Each feature map is essentially normalized independently since the mean and standard deviation are computed independently for every sample and per channel. Batch-Instance Normalization combines the ideas of both batch normalization and instance normalization. It operates as follows:

Determine the mean and standard deviation for every channel throughout the batch dimension (much like BN).

Using the mean and standard deviation computed over the batch dimension, normalize each sample in the batch.

Next, (just as with IN) normalize each feature map (channel) across the spatial dimensions.

Lastly, employ learnable parameters to shift and scale the normalized activations (as in BN and IN).

The batch normalization operation is defined as follows x is an input tensor of shape (N,C,H,W), where N is the batch size, C is the number of channels, H is the height, and W is the width: Compute the mean μ and variance σ^2 of the input tensor across the batch dimension:

$$\mu_C^{(B)} = \frac{1}{NHW} \sum_N \sum_H \sum_W x_{nchw} \tag{15}$$

$$\sigma_c^{2(B)} = \frac{1}{NHW} \sum_N \sum_H \sum_W (x_{nchw} - \mu_C^{(B)})^2 \tag{16}$$

Normalize the input tensor using the mean and variance:

$$\bar{\mathrm{x}}_{nchw}^{(B)} = \frac{x_{nchw} - \mu_C^{(B)}}{\sqrt{\sigma_C^{2(B)} + \epsilon}} \tag{17}$$

Batch-normalized response is $\bar{\mathrm{x}}_{nchw}^{(B)}$. Instance Normalization normalizes per-instance feature;

$$\mu_{nC}^{(I)} = \frac{1}{HW} \sum_H \sum_W x_{nchw} \tag{18}$$

$$\sigma_c^{2(I)} = \frac{1}{HW}\sum_H\sum_W (x_{nchw} - \mu_C^{(I)})^2$$

(19)

Normalize the input tensor using the mean and variance:

$$\bar{x}_{nchw}^{(I)} = \frac{x_{nchw} - \mu_{nC}^{(I)}}{\sqrt{\sigma_C^{2(I)} + \epsilon}}$$

(20)

Instance-normalized response is $\bar{x}_{nchw}^{(I)}$. Aim of this approach is adaptively balancing $\bar{x}_{nchw}^{(B)}$ and $\bar{x}_{nchw}^{(I)}$ for each channel. Scale and shift the normalized tensor using learnable parameters γ (scale) and β (shift):

$$y = \left(\rho . x^{(B)} + (1 - \rho) . x^{(I)}\right) . y + \beta$$

(21)

Batch-Instance Normalization offers a middle ground between batch normalization and instance normalization, making it particularly useful in scenarios with small batch sizes where batch normalization may not perform optimally. It can improve model generalization and stability by providing normalization both across batches and within instances. Formulas are given for applying Batch-Instance Normalization. Equation (21) is very similar to the single exponential smoothing method which is a statistics method. In the following section, single exponential smoothing is explained.

3.3 Single Exponential Smoothing

Exponential smoothing, initially proposed by Robert Goodell Brown in 1956 without referencing prior work, and later expanded by Charles C. Holt in 1957, presents a highly effective method for smoothing time series data through the application of an exponential

window function. The key parameter influencing the exponential smoothing computation is termed the smoothing factor or constant [48].

In contrast to the simple moving average, where historical observations are given equal weights, exponential smoothing employs exponential functions to assign progressively diminishing weights to past data points. Exponential smoothing plays a significant role in machine learning, particularly in time series forecasting and data analysis. This approach is used in various applications such as forecasting, noise reduction, and adaptive learning. The equation of single exponential smoothing is given below.

$$s_t = \alpha x_t + (1 - \alpha) s_{t-1} \quad (22)$$

s_t = smoothed statistic (22)

s_{t-1} = previous smoothed statistic (22)

α = smoothing factor of data; $0 < \alpha < 1$ (22)

t = time period (22)

In Batch-Instance Normalization, the scaling factor s_t for a given channel or feature map at time *t* is calculated as a weighted average of the current instance's activation x_t and the previous scaling factor s_{t-1}. The weight assigned to x_t is controlled by the parameter α, while the weight assigned to s_{t-1} is $1-\alpha$.This adaptive scaling factor allows the normalization process to adapt to changes in the input data distribution, improving the stability and effectiveness of the normalization layer. Batch-Instance Normalization is a network layer. It is inserted between a hidden layer and the next hidden layer. Its task is normalizing the outputs from previous hidden layer and passing normalized values to the next hidden layer. It is shown in Figure 3.2.

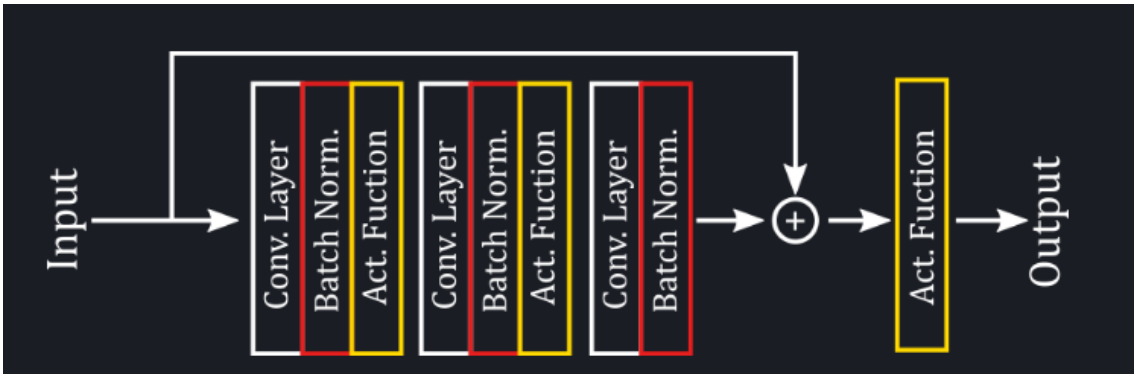


Figure 3.2: Example of a residual block [49]

In Figure 3.3, inside of a Batch-Instance Normalization layer is shown. First, mean and variance are taken for both batch and instance. Then each result is normalized separately. The normalized values are combined in the scale and shift phase, and the result of this operation is transferred to the next layer.

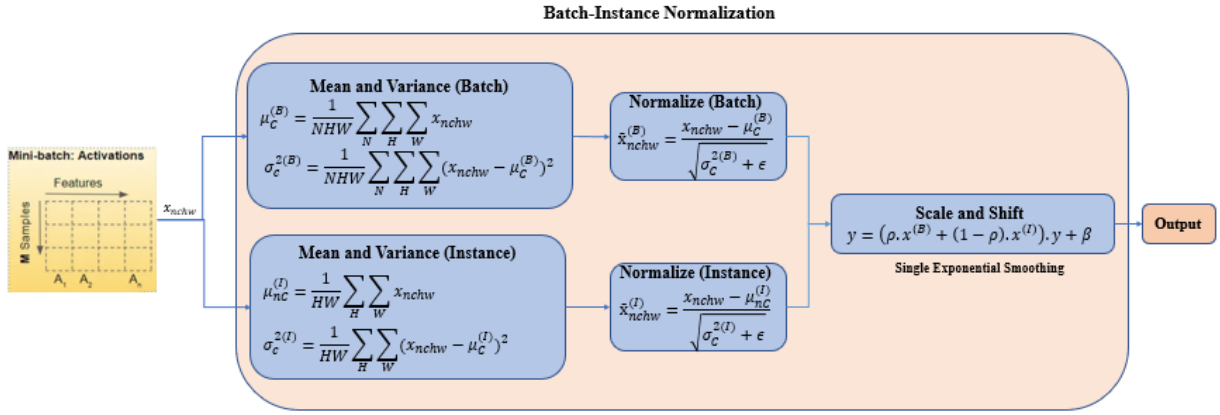


Figure 3.3: Batch-Instance Normalization Operation

3.4 Double Exponential Smoothing

This approach is alternatively known as Holt's trend-corrected or second-order exponential smoothing. It is applied for predicting time series. The fundamental concept behind double exponential smoothing is to incorporate a term that considers the potential presence of a trend in the series. This slope component is itself updated through exponential smoothing [50]. This methodology is implemented for Batch-Instance Normalization.

$$s_t = \alpha x_t + (1 - \alpha)(s_{t-1} + b_{t-1})$$

(23)

$$\beta_t = \beta(s_t - s_{t-1}) + (1 - \beta)b_{t-1}$$

(24)

s_t = smoothed statistic (23) (24)

s_{t-1} = previous smoothed statistic (23) (24)

α = smoothing factor of data; $0 < \alpha < 1$ (23)

t = time period (23) (24)

b_t = best estimate of trend at time t (23)

β = trend smoothing factor; $0 < \beta < 1$ (24)

In the following section, the forest fire dataset, data augmentation, optimizers used, comparison criteria, experimental conditions are mentioned. Experiment are shown with different graphs.

CHAPTER 4

EXPERIMENTAL RESULTS

4.1 Forest Fire Dataset

The dataset is taken from the Kaggle. According to the collector, the dataset was created by combining and merging various other smaller datasets. The Dataset includes 4609 images, 2120 Fire and 2499 Non-Fire images [51]. The dataset is divided into Test Data and Train Data categories. Test Data consists of 1830 images, it includes 838 Fire and 992 Non-Fire images. Train Data consists 2789 images, it includes 1282 Fire and 1507 Non-Fire images. Train Data size is increased from 2789 to 11324 through data augmentation. After the data augmentation, Train/Test Distribution reached %86 / %14. The details of data augmentation are provided in the following section. Sample images are shown in Figure 4.1 and Figure 4.2.

Figure 4.1: Sample Fire Images [51]

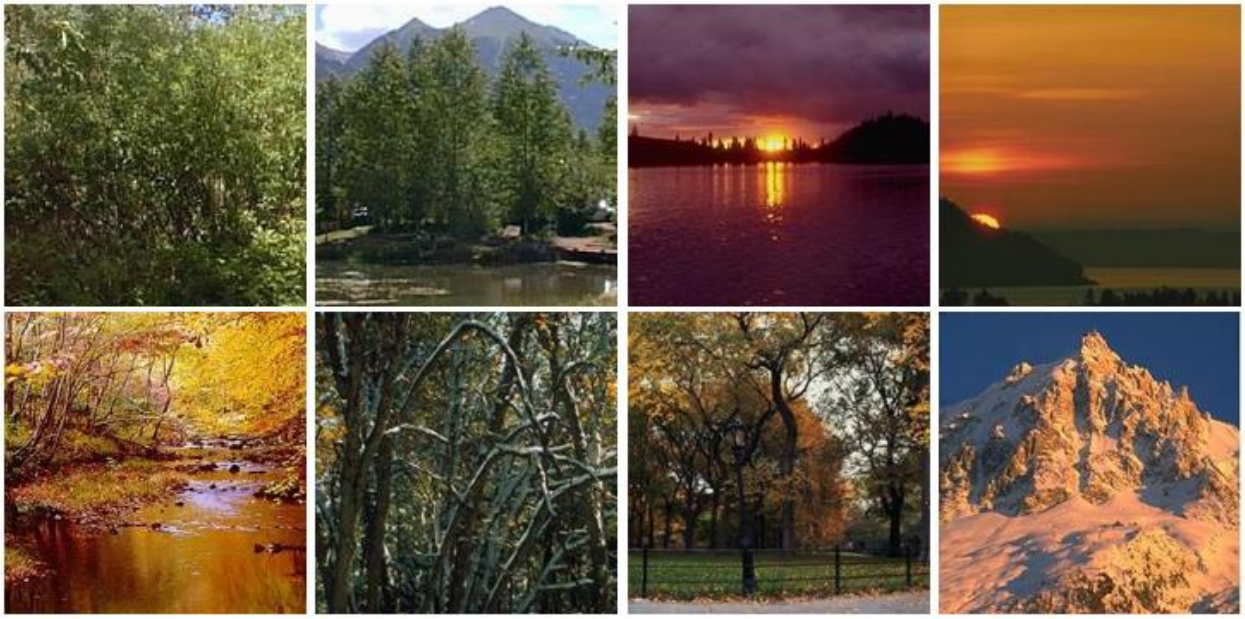


Figure 4.2: Sample Non Fire Images [51]

4.2 Data Augmentation

Data augmentation is a technique used to increase the size of a dataset by applying different methods to the existing dataset. It's particularly useful in machine learning, especially in tasks like image classification, object detection, and natural language processing, where having a large and diverse dataset is crucial for training robust models [52].

Common Data Augmentation methods are mentioned below.

Rotation: Images can be rotated in a clockwise or counterclockwise direction to add variation to the dataset.

Shearing: Shearing is the process of moving a portion of an image along a specific axis. It can simulate distortions caused by camera angle or perspective changes.

Zooming (Scaling): Zooming, or scaling, involves resizing the image, either by zooming in (enlarging) or zooming out (shrinking).

Cropping: Cropping involves removing a part of the image, typically from the edges. It helps the model become invariant to irrelevant background information and focus on the most relevant parts of the image.

Flipping: Flipping the image horizontally or vertically helps the model learn features that are invariant to left-right or up-down orientation changes.

Changing the Brightness Level: Adjusting the intensity of the pixels in the image helps the model become robust to variations in lighting conditions [53].

In this study, all tests have been done with Google Colab T4 GPU. Python programming language is used for development. Pytorch, tensorflow and keras libraries are used. Keras image processing library is used to increase size of dataset. The parameters were 40-degree rotation, 0.2 shear, 0.2 zoom, horizontal flip, brightness (0.5,1.5). Training dataset is increased from 3000 to 11324 images by using the mentioned parameters. In Figure 4.3, sample augmented images are shown.

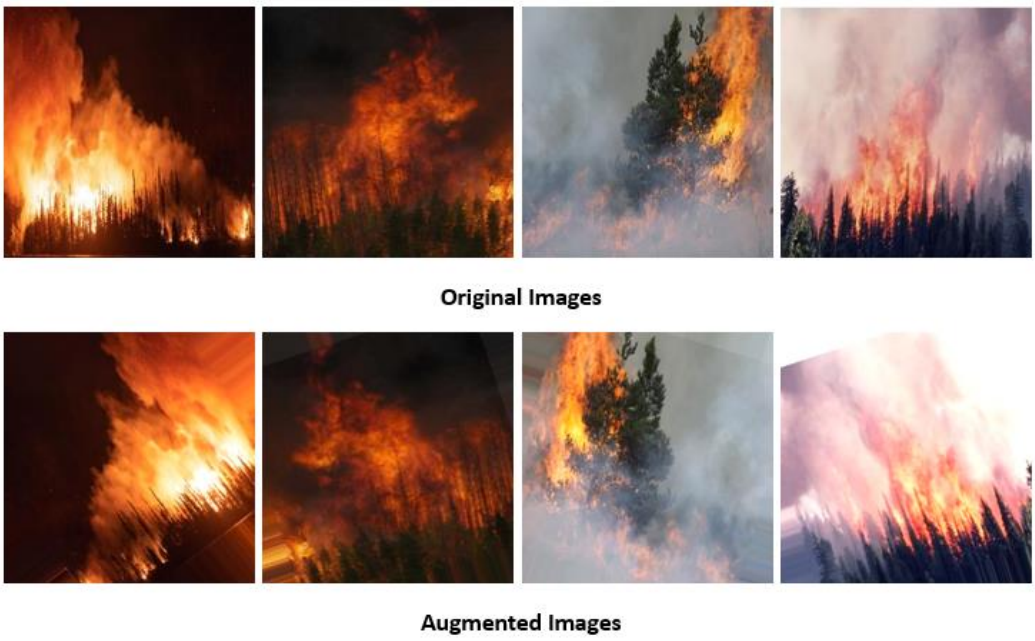


Figure 4.3: Sample Augmented Images [51]

4.3 Optimizers

In machine learning, an optimizer refers to an algorithm or method used to minimize the loss function or objective function during the training of a model. The main purpose of an optimizer is to adjust the weights and biases of the model iteratively, so that the model's predictions become more accurate over time. Most optimizers rely on the concept of gradient descent. This approach involves updating the model's parameters based on the gradients of the loss function with respect to those parameters. Stochastic Gradient Descent (SGD), RMSprop, Adam, Adagrad, and Adadelta are a few examples of common optimizers. These optimizers may differ in terms of computational efficiency, convergence properties, and adaptability. Adaptive learning rates are a characteristic of several optimizers. They help to enhance convergence and stability in the training process by dynamically adjusting the learning rate during. Optimizers are essential to the machine learning process since they decide how to update the model parameters in order to minimize the loss function and enhance the prediction performance of the model [31].

4.4 Comparison Criteria

Evaluating methods are used to measure the performance of neural networks. In this study, accuracy, precision, F1 score, sensitivity, specificity, confusion matrix are used as metrics. These evaluation metrics are explained below.

True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN) terms are elements of the confusion matrix, which is a tabular representation of actual versus predicted class labels.

True Positives (TP): The number of correctly predicted positive instances.

True Negatives (TN): The number of correctly predicted negative instances.

False Positives (FP): The number of incorrectly predicted positive instances.

False Negatives (FN): The number of incorrectly predicted negative instances.

Accuracy: Accuracy is a widely used metric to evaluate classification models. It measures the overall correctness of predictions made by the model. Accuracy is calculated as the ratio of true postives and true negatives to the total number of instances. While accuracy is intuitive, it might not be the best metric for imbalanced datasets [54].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision: Precision measures the correctness of positive predictions made by the model. It indicates the proportion of true positives among all positive predictions made. Precision is crucial in applications where false positives are costly [55].

$$Precision = \frac{TP}{TP + FP}$$

Sensitivity (Recall): Sensitivity measures the ability of the model to correctly identify positive instances. It's the ratio of true positives to the total number of actual positive instances [55].

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

F1 Score: The F1 score is the harmonic mean of precision and recall. A higher score signifies better model performance; the best possible score is 1, and the range is 0 to 1. When working with datasets that are unbalanced and have a skewed class distribution, it is especially helpful [55].

$$F1\ Score = 2\ \times \frac{Precision\ \times\ Recall}{Precision + Recall}$$

Specificity: Specificity measures the ability of the model to correctly identify negative instances. It's the ratio of true negative predictions to the total number of actual negative instances [56].

$$Specificity = \frac{TN}{TN + FP}$$

These metrics and the confusion matrix provide a comprehensive understanding of the performance of machine learning models, especially in classification tasks. They are essential for assessing model effectiveness and making informed decisions about model improvements.

4.5 Experiment Conditions

In this study, the effects of Batch-Instance Normalization with different approaches are demonstrated. Batch-Instance Normalization with single and double exponential smoothing, Batch Normalization, Instance Normalization were compared. Models are created with different normalization layers for comparison.

ResNet-18, ResNet-34, ResNet-50, ResNet-101 models are used to detect forest fires and we used Forest Fire Dataset which we mentioned before. The networks are trained with different optimizers like SGD, RMSprop, Adam, AdamW, Adadelta, Adagrad, Adamax, Nadam. Batch size of 128, learning rate of 0.1, epochs of 30 are some of the parameters for training.

For each normalization type, the effect of different optimizers is observed. In Batch-Instance Normalization with single and double exponential smoothing case, the effects of gate values are also observed.

ResNet Hypermeters used in experiments are explained below.

Depth (Number of Layers): The depth parameter is set to 18, 34, 50, 101 for different experiments.

Number of Classes: The number of classes are set to 2. These are Fire and No Fire.

Normalization Type: The normalization type parameter is set to batch normalization, instance normalization and Batch-Instance Normalization for different experiments.

Filter Size: The filter size is set to 3x3.

Stride: The stride for convolutional operations is set to 1 by default within the conv3x3 function.

Padding: Padding is set to '1', indicating 'same' padding, within the conv3x3 function.

Learning Rate: The Learning rate is set to 0.1.

Optimizer: The optimizer is set to SGD, RMSprop, Adam, AdamW, Adadelta, Adagrad, Adamax, Nadam for different experiments.

Batch Size: The batch size is set to 128.

4.6 Experiments

4.6.1 Experiment 1

In experiment 1, Different optimizers are tested to determine the best optimizer. Optimizers are tested with Batch-Instance Normalization. The comparison of optimizers is shown below.

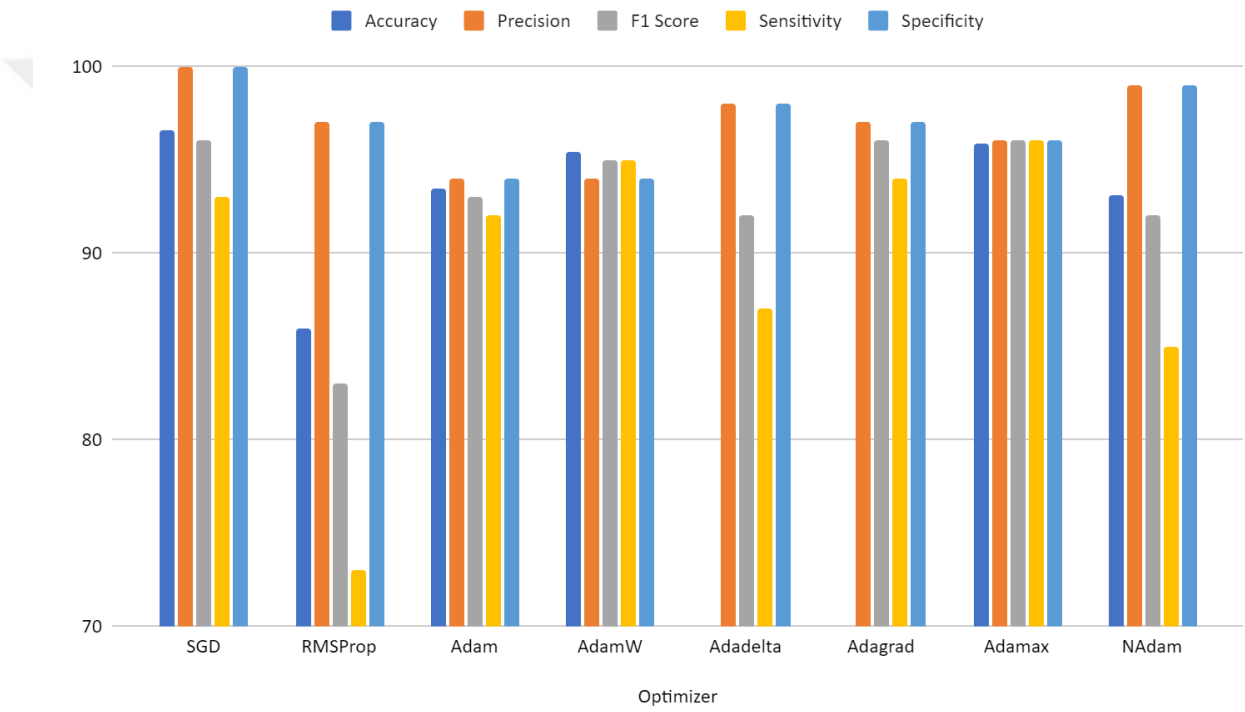


Figure 4.4: Different optimizers with Batch-Instance Normalization

The studies examined several batch-instance single normalization configurations using varying ResNet depths, training epoch counts, and optimization techniques. According to the Figure 4.4, With batch-instance single normalization and stochastic gradient descent (SGD) optimization produced the best results, with an accuracy of 96.56%, perfect precision (100%) and a strong F1 score of 96%. Strong performance was shown by this arrangement, which produced great sensitivity (93%) and specificity (100%), with a high number of true positives (TP) and true negatives (TN) and negligible false positives (FP) and false negatives (FN). Other optimizers also investigated including RMSProp, Adam,

AdamW, Adadelta, Adagrad, Adamax, and NAdam. Although the accuracy scores obtained in these studies were commendable, ranging from 85.96% to 95.85%, there were variations in precision, F1 score, and other performance metrics among the different configurations. Notably, although some configurations, like RMSProp optimization, showed lesser precision and F1 score despite excellent accuracy, others, like AdamW optimization, revealed high precision and F1 score (both at 94%). Out of all the investigated configurations, SGD performs the best overall, demonstrating the usefulness of batch-instance single normalization with SGD optimization for the specified task and dataset. According to the test about optimizers, SGD optimizer is used in the rest of the tests.

4.6.2 Experiment 2

88 different situations were tested in this experiment. In line with the information in the table, the cases were examined in different circumstances. Normalization types and the ResNet depth is changed in each case. Normalization types are “No Normalization”, “Batch Normalization”, “Instance Normalization”, “Batch-Instance Normalization with Single Exponential Smoothing” and “Batch-Instance Normalization with Double Exponential Smoothing”. Tested ResNet models are ResNet-14, ResNet-34, ResNet-50 and ResNet-101. Graphs are drawn for each evaluation metric. At first, graphs are drawn for batch, instance normalization with different depth of ResNet.

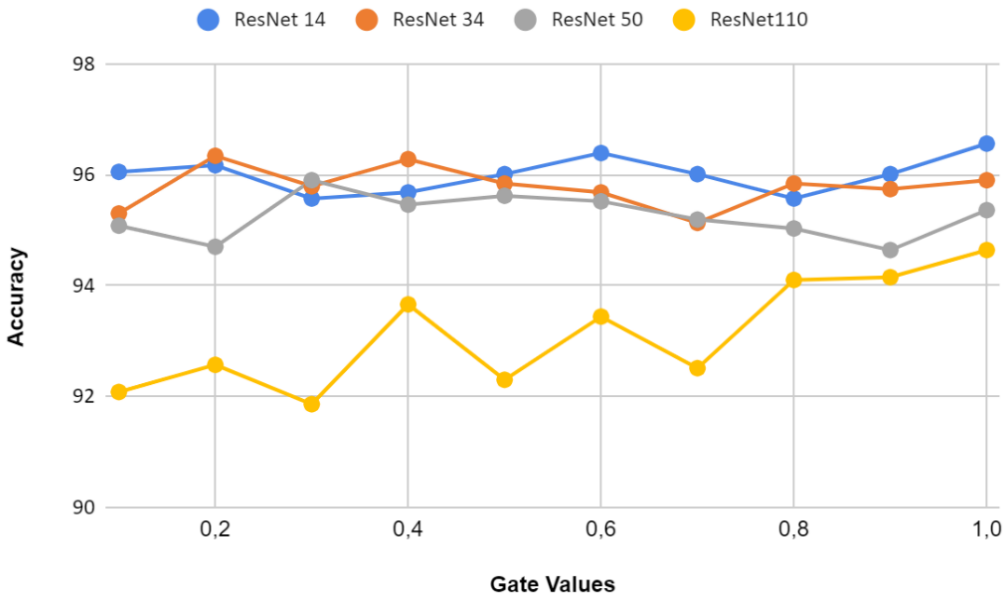


Figure 4.5: Batch-Instance Normalization Accuracy

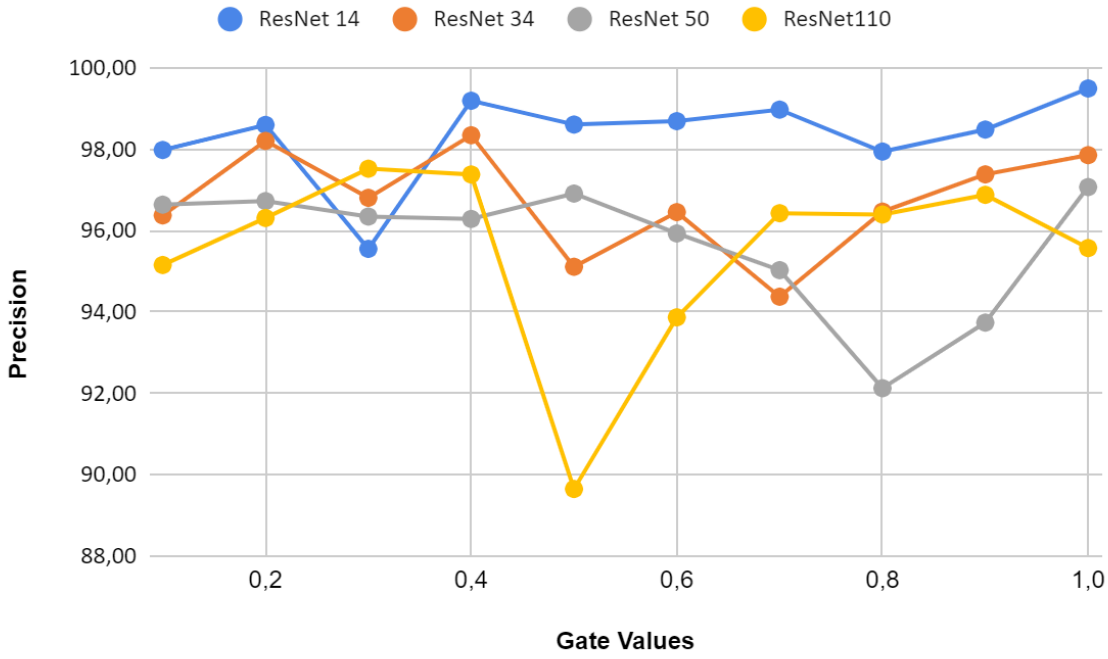


Figure 4.6: Batch-Instance Normalization Precision

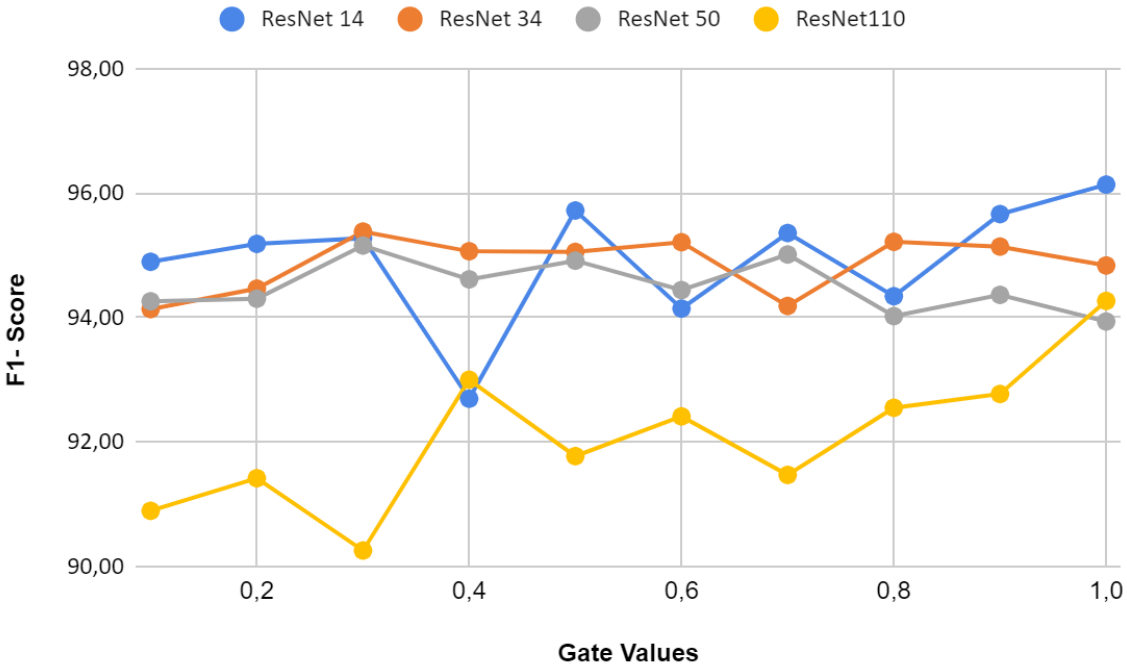


Figure 4.7: Batch-Instance Normalization F1-Score

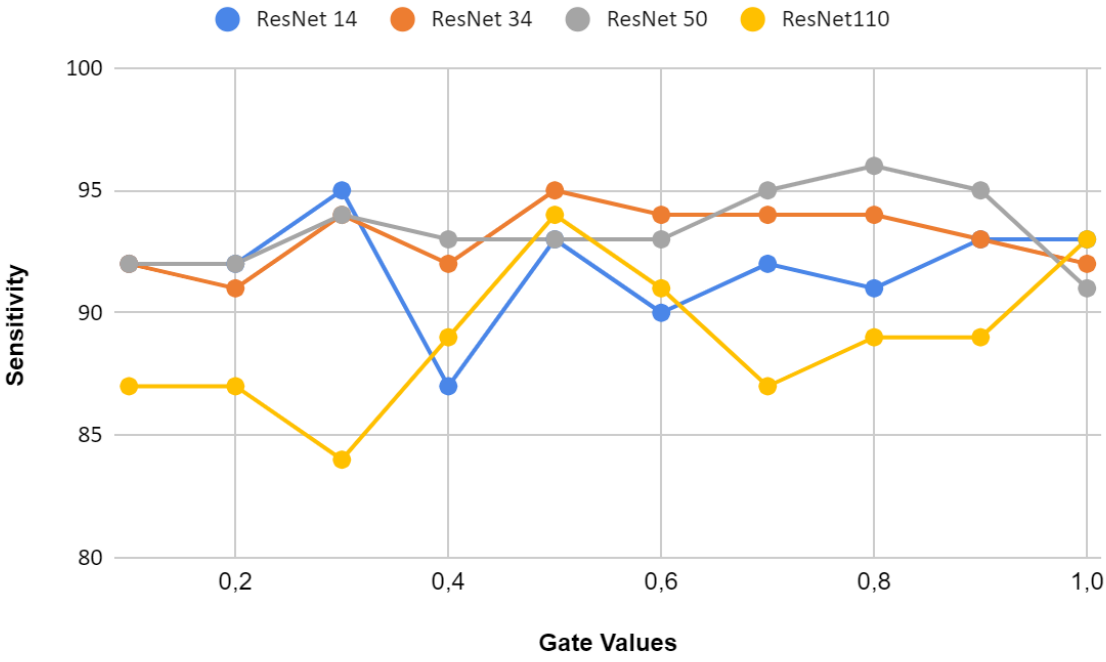


Figure 4.8: Batch-Instance Normalization Recall(Sensitivity)

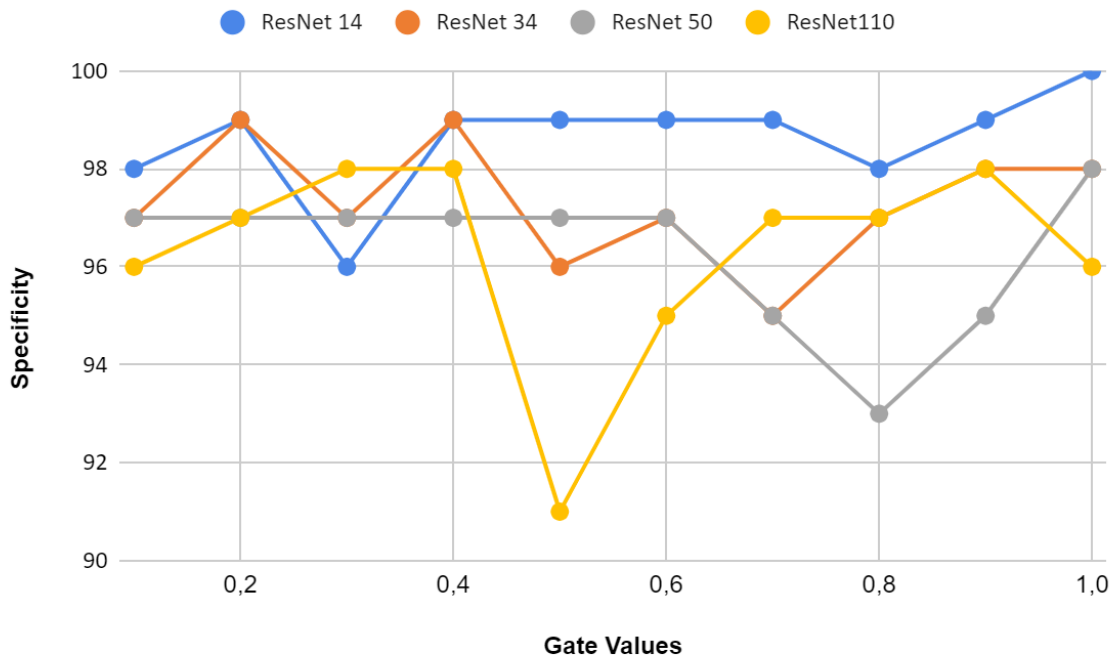


Figure 4.9: Batch-Instance Normalization Specificity

The Figures 4.5-4.9 display the findings from multiple experiments carried out on a dataset, most frequently associated with a classification problem. These studies examine a range of configurations, including alpha and beta values, optimization algorithms, ResNet depths, and the number of training epochs. Various normalizing strategies, such as batch-instance single and double normalization, are also explored. Accuracy, precision, F1 score, true positives (TP), true negatives (TN), false positives (FP), false negatives (FN), sensitivity, and specificity are among the performance indicators that are used to evaluate each experiment. Batch-instance single normalization with an alpha value of 1 performs better than other alpha values, it is showing good F1 score, accuracy, and precision.

4.6.3 Experiment 3

In Experiment 3, without Normalization, Batch Normalization, Instance Normalization, Batch-Instance Normalization with single/double exponential smoothing are tested on different Res-Net models and compared. Overall comparison of all normalization methods are shown in Figures 4.10-4.14.

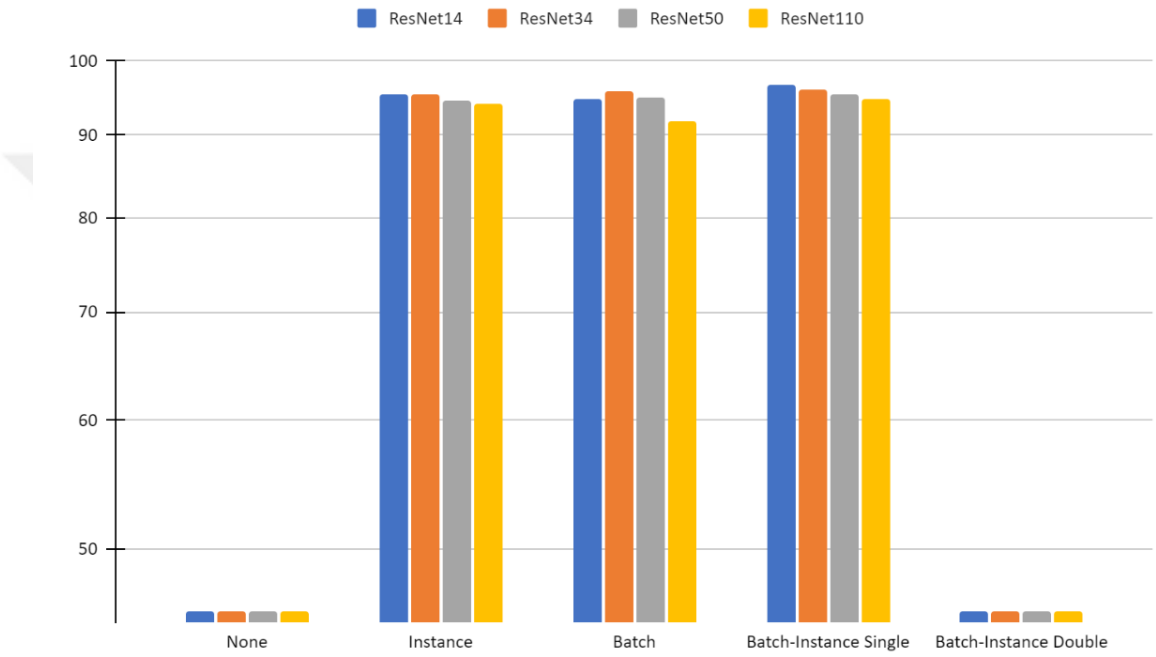


Figure 4.10: Normalization Methods Accuracy

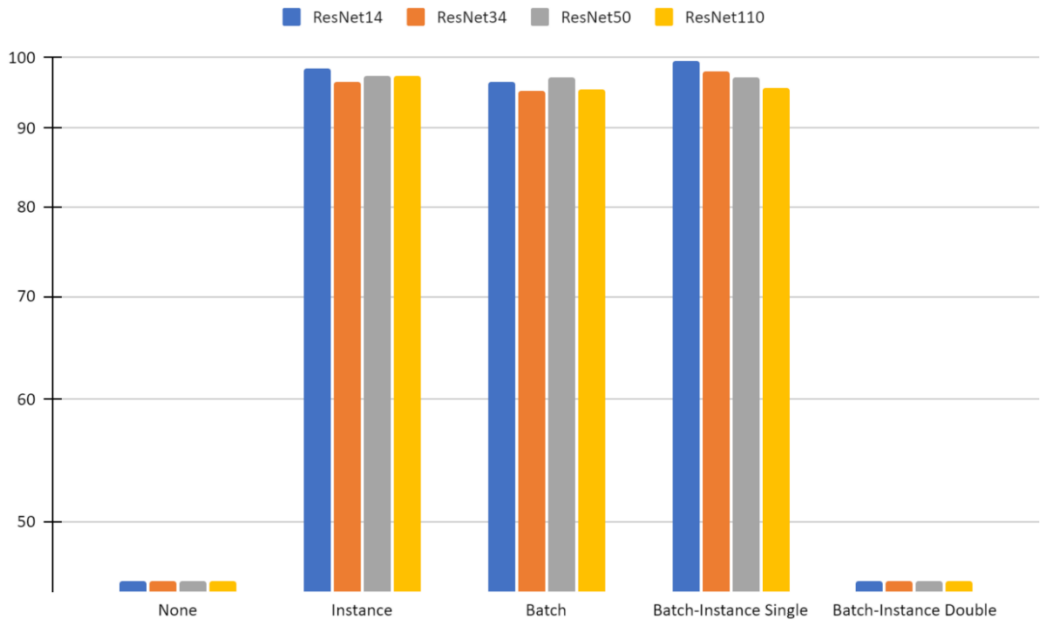


Figure 4.11: Normalization Methods Precision

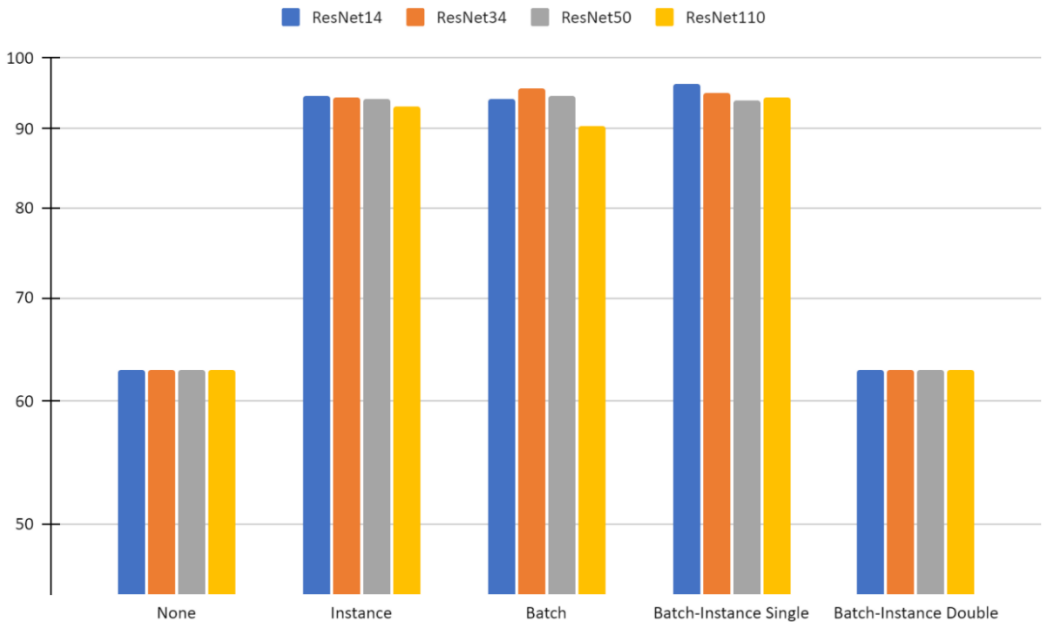


Figure 4.12: Normalization Methods F1-Score

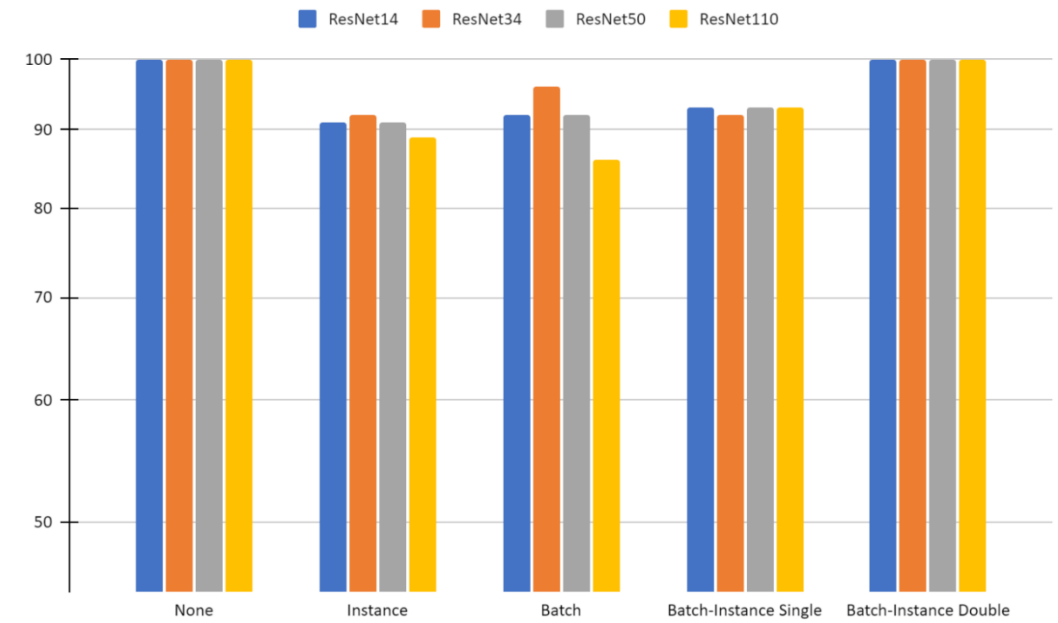


Figure 4.13: Normalization Methods Sensitivity

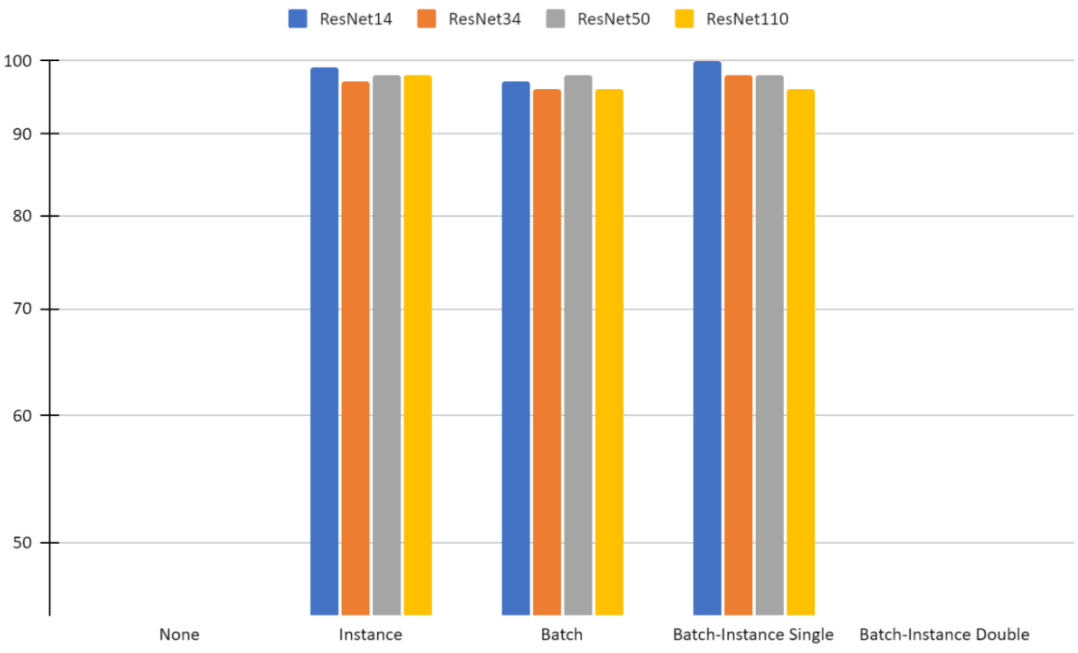


Figure 4.14: Normalization Methods Specificity

Across a range of optimization strategies and experimental conditions, both batch and instance normalization techniques show competitive average performance. An accuracy score of approximately 93.46% is obtained on average by batch normalization. In contrast, the average accuracy of instance normalization is approximately 93.71%. According to the graphs best accuracy were achieved in batch normalization with ResNet 34 which is 95.73. In double exponential smoothing and without normalization, accuracy score of %45,73 is achieved in every case. Analysis reveals that batch-instance single normalization with an alpha value of 1 performs well in all experiments, showing good F1 score, accuracy, and precision. With an F1 score of 96.14%, accuracy of 96.56%, and precision of 99.49%, Batch-Instance Normalization stands out above the others. These findings demonstrate how well batch-instance single normalization performs when effectively classifying instances, especially for tasks involving the dataset that is presented.

4.7 Detection Time

The times spent in the training and testing phases were measured. These measurements have been done with Batch-Instance Normalization and ResNet-14. In the training phase, each epoch is completed in 38 seconds. The model tests 1830 images in 6 seconds so it detects a frame in 3.3ms. In real-time applications, processing time for each frame should be less than the frame rates. Common target frame-rate range is 24-60 FPS in real-time applications [57]. Maximum acceptable time for per frame is 16.67 millisecond in 60 FPS. Detecting time in this study is less than the maximum acceptable time in 60 FPS. Based on this comparison, real-time forest fire detection can be involved in the future work.

4.8 Discussion

The results of our extensive experiments demonstrate the profound impact of the Batch-Instance Normalization and various normalization approaches on the Res-Net model for wildfire detection purposes. We aimed to determine the most effective strategy for wildfire detection using a 32x32-input ResNet-14, ResNet-34, ResNet-50, ResNet101 model trained on the Wildfire Dataset, along with comparisons with single and double exponential smoothing, Batch Normalization, and Instance Normalization. We used 88

different test scenarios which includes different normalization approaches and different depths of ResNet. The most appropriate optimizer is selected according tests of different optimizers. SGD produced the best results, with an accuracy of 96.56%, perfect precision (100%) and a strong F1 score of 96%. After this test, SGD is used in rest of the experiments. In the experiments, An accuracy score of approximately 93.46% is obtained on average by batch normalization and the average accuracy of instance normalization is approximately 93.71%. The best accuracy were achieved in batch normalization with ResNet 34 which is 95.73. In experiments of without normalization and Batch-Instance Normalization %45,79 accuracy is recorded. In experiments of batch-instance single normalization with an alpha value of 1 achieves F1 score of 96.14%, accuracy of 96.56%, and precision of 99.49%, batch-instance single normalization stands out above the others. The conclusion, and the one we target in this research, is that Batch-Instance Normalization improves the accuracy better than other approaches with 96.56% accuracy. In conclusion, combining Batch Instance Normalization with single exponential smoothing has demonstrated a promising and effective solution for wildfire detection, demonstrating the potential for widespread use in similar applications.

CHAPTER 5

CONCLUSION

Wildfires have long-term devastating effects on ecosystems, such as destroying vegetation dynamics, greenhouse gas emissions, loss of wildlife habit, and destruction of land covers. Today, many traditional fire detection methods are used, but these methods are insufficient to convey timely and accurate information. In this regard, the latest technology needs to be investigated for more effective possible solutions. As an effective possible solution, we propose to combine the Batch-Instance Normalization approach with ResNet, a deep learning model for forest fire detection. Batch normalization and instance normalization are connected to each other by gate parameters. In this study, forest fire dataset is used for training the model. The Dataset includes 4609 images, 2120 Fire and 2499 Non-Fire images. The dataset is divided into Test Data and Train Data categories. Test Data consists of 1830 images, it includes 838 Fire and 992 Non-Fire images. Train Data consists 2789 images, it includes 1282 Fire and 1507 Non-Fire images. Train Data size is increased 2789 to 11324 with data augmentation. After the data augmentation, Train/Test Distribution reached %86 / %14. In this study, accuracy, precision, F1 score, sensitivity, specificity, confusion matrix are used as metrics. As seen in the experiments, Batch-Instance Normalization exhibits superior performance than other normalization methods such as batch and instance. Our experiments verify the batch-instance success claimed in the article titled "Batch-Instance Normalization for Adaptively Style-Invariant Neural Networks", which introduces this approach for the CIFAR dataset. This study demonstrated significant performance by reaching an F1 score of 96.14%, accuracy of 96.56%, and precision of 99.49% in the forest fire data set we used. A minimum 1%, accuracy difference, %0.6 F1 score difference, %1.05 precision difference were obtained from other approaches. While achieving this accuracy, eight different optimizers were

tried, and the most appropriate optimizer was selected according to the performances of optimizers and different depths of ResNet were tried. Apart from the experiments for optimizer selection, the double exponential smoothing approach was also tried instead of the single exponential smoothing statistical approach used in the Batch-Instance Normalization, but it did not give better results than single exponential smoothing. In addition to proving the positive effects of the batch-ins norm, this study makes a significant contribution to this field by demonstrating its applicability in the context of forest fire detection. The observed results highlight the potential of this normalization type of approach. This study is one of the important examples of the use of Batch-Instance Normalization. This approach will become more widespread in the future, at least for object recognition and classification studies. In the future, enhancing early detection capabilities could be achieved by the integration of multimodal data, IoT-powered real-time monitoring, and UAVs fitted with fire detection algorithms.

REFERENCES

[1] I. Brunner and D. Godbold, "Tree roots in a changing world," *Journal of Forest Research,* vol. 12, pp. 78-82, 04/02 2007.

[2] G. Ball, P. Regier, R. González-Pinzón, J. Reale, and D. Van Horn, "Wildfires increasingly impact western US fluvial networks," *Nature Communications,* vol. 12, 04/30 2021.

[3] F. Carta, C. Zidda, M. Putzu, D. Loru, M. Anedda, and D. Giusto, "Advancements in Forest Fire Prevention: A Comprehensive Survey," (in eng), *Sensors (Basel),* vol. 23, no. 14, Jul 24 2023.

[4] D. Sabah, "Türkiye's Marmaris forest goes green after devastating wildfires," Daily Sabah, Accessed [09.01.2024], Available: https://www.dailysabah.com/turkiye/turkiyes-marmaris-forest-goes-green-after-devastating-wildfires/news?gallery_image=undefined#big.

[5] J. Chai, H. Zeng, A. Li, and E. W. T. Ngai, "Deep learning in computer vision: A critical review of emerging techniques and application scenarios," *Machine Learning with Applications,* vol. 6, p. 100134, 2021/12/15/ 2021.

[6] H. Nam and H.-E. Kim, "Batch-Instance Normalization for Adaptively Style-Invariant Neural Networks," *ArXiv,* vol. abs/1805.07925, 2018.

[7] P. Barmpoutis, P. Papaioannou, K. Dimitropoulos, and N. Grammalidis, "A Review on Early Forest Fire Detection Systems Using Optical Remote Sensing," *Sensors,* vol. 20, no. 22.

[8] G. Subbiah, C. Abhishek, and C. Akshayanat, "Machine Vision Based Fire Detection Techniques: A Survey," *Fire Technology,* vol. 57, 11/27 2020.

[9] M. Xin and Y. Wang, "Research on image classification model based on deep convolution neural network," *EURASIP Journal on Image and Video Processing,* vol. 2019, no. 1, p. 40, 2019/02/11 2019.

[10] D. Wu, Z. Chunjiong, L. Ji, R. Ran, H. Wu, and Y. Xu, "Forest Fire Recognition Based on Feature Extraction from Multi-View Images," *Traitement du Signal,* vol. 38, pp. 775-783, 06/30 2021.

[11] X. B. Qiu *et al.*, "Fire Detection Algorithm Combined with Image Processing and Flame Emission Spectroscopy," (in English), *Fire Technology,* vol. 54, no. 5, pp. 1249-1263, Sep 2018.

[12] D. Dzigal, A. Akagic, E. Buza, A. Brdjanin, and N. Dardagan, "Forest Fire Detection based on Color Spaces Combination," in *2019 11th International Conference on Electrical and Electronics Engineering (ELECO)*, 28-30 Nov. 2019 2019, pp. 595-599.

[13] A. Khalil, S. U. Rahman, F. Alam, I. Ahmad, and I. Khalil, "Fire Detection Using Multi Color Space and Background Modeling," *Fire Technology,* vol. 57, no. 3, pp. 1221-1239, 2021/05/01 2021.

[14] K. Patel, A. Kar, S. Jha, and M. Khan, "Machine vision system: A tool for quality inspection of food and agricultural products," *Journal of food science and technology,* vol. 49, pp. 123-41, 04/01 2012.

[15] O. Cosido, *Hybridization of Convergent Photogrammetry, Computer Vision, and Artificial Intelligence for Digital Documentation of Cultural Heritage - A Case Study: The Magdalena Palace*. 2014.

[16] A. Gaur, A. Singh, A. Kumar, A. Kumar, and K. Kapoor, "Video Flame and Smoke Based Fire Detection Algorithms: A Literature Review," *Fire Technology,* vol. 56, no. 5, pp. 1943-1980, 2020/09/01 2020.

[17] H. Wu, D. Wu, and J. Zhao, "An intelligent fire detection approach through cameras based on computer vision methods," *Process Safety and Environmental Protection,* vol. 127, pp. 245-256, 2019/07/01/ 2019.

[18] A. Khondaker, A. Khandaker, and J. Uddin, "Computer Vision-based Early Fire Detection Using Enhanced Chromatic Segmentation and Optical Flow Analysis Technique," *The International Arab Journal of Information Technology,* vol. 17, pp. 947-953, 11/01 2020.

[19] Y. He *et al.*, "Smart detection of indoor occupant thermal state via infrared thermography, computer vision, and machine learning," *Building and Environment,* vol. 228, p. 109811, 2023/01/15/ 2023.

[20] M. Mazur-Milecka, N. Głowacka, M. Kaczmarek, A. Bujnowski, M. Kaszyński, and J. Rumiński, "Smart city and fire detection using thermal imaging," in *2021 14th International Conference on Human System Interaction (HSI)*, 8-10 July 2021 2021, pp. 1-7.

[21] A. Bouguettaya, H. Zarzour, A. M. Taberkit, and A. Kechida, "A review on early wildfire detection from unmanned aerial vehicles using deep learning-based computer vision algorithms," *Signal Processing,* vol. 190, p. 108309, 2022/01/01/ 2022.

[22] Y. Li, "Research and Application of Deep Learning in Image Recognition," in *2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA)*, 21-23 Jan. 2022 2022, pp. 994-999.

[23] Y. Li, "Research and Application of Deep Learning in Image Recognition," in 2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA), Jan. 21-23, 2022, pp. 994-999.

[24] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Computer Science Review*, vol. 40, p. 100379, 2021/05/01/ 2021.

[25] S. Saponara, A. Elhanashi, and A. Gagliardi, "Real-time video fire/smoke detection based on CNN in antifire surveillance systems," *Journal of Real-Time Image Processing*, vol. 18, pp. 1-12, 06/01 2021.

[26] G. Lin, Y. Zhang, G. Xu, and Q. Zhang, "Smoke Detection on Video Sequences Using 3D Convolutional Neural Networks," *Fire Technology*, vol. 55, no. 5, pp. 1827-1847, 2019/09/01 2019.

[27] J. Florath and S. Keller, "Supervised Machine Learning Approaches on Multispectral Remote Sensing Data for a Combined Detection of Fire and Burned Area," *Remote Sensing*, vol. 14, p. 657, 01/29 2022.

[28] P. Barmpoutis, T. Stathaki, K. Dimitropoulos, and N. Grammalidis, "Early Fire Detection Based on Aerial 360-Degree Sensors, Deep Convolution Neural Networks and Exploitation of Fire Dynamic Textures," (in English), *Remote Sensing*, vol. 12, no. 19, Oct 2020, Art no. 3177.

[29] R. K. Mohammed, "A real-time forest fire and smoke detection system using deep learning," *International Journal of Nonlinear Analysis and Applications*, vol. 13, no. 1, pp. 2053-2063, 2022.

[30] C. M. Bishop, Pattern Recognition and Machine Learning. Springer-Verlag, New York, NY, USA, 2006.

[31] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, Cambridge, MA, USA, 2016.

[32] C. Molnar, Interpretable Machine Learning: A Guide for Making Black Box Models Explainable, 2nd ed. Springer, New York, NY, USA, 2022.

[33] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2e ed. New York, NY, USA: Springer, 2006.

[34] R. Tibshirani, T. Hastie, and J. H. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction : with 200 Full-color Illustrations. Springer, New York, NY, USA, 2001.

[35] A. C. Muller and S. Guido, Introduction to Machine Learning with Python: A Guide for Data Scientists. O'Reilly Media, Incorporated, Beijing, China, 2018.

[36] F. Provost and T. Fawcett, *Data science for business*. Beijing: O'Reilly (in English), 2013.

[37] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Applied Soft Computing*, vol. 97, p. 105524, 2020/12/01/ 2020.

[38] V. Thakkar, S. Tewary, and C. Chakraborty, "Batch Normalization in Convolutional Neural Networks — A comparative study with CIFAR-10 data," in *2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT)*, 12-13 Jan. 2018 2018, pp. 1-5.

[39] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in Proceedings of the 32nd International Conference on Machine Learning (ICML), Feb. 10, 2015.

[40] J. Ba, J. Kiros, and G. Hinton, "Layer Normalization," arXiv:1607.06450, Jul. 21, 2016.

[41] X. Huang and S. Belongie, "Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 22-29 October 2017 2017, pp. 1510-1519.

[42] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance Normalization: The Missing Ingredient for Fast Stylization," arXiv:1607.08022, Jul. 27, 2016.

[43] Y. Wu and K. He, "Group Normalization," *International Journal of Computer Vision*, vol. 128, no. 3, pp. 742-755, 2020/03/01 2020.

[44] A. Gunawan, X. Yin, and K. Zhang, "Understanding and Improving Group Normalization," arXiv:2201.06038, Jan. 11, 2022.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 27-30 June 2016 2016, pp. 770-778.

[46] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv 1409.1556*, 09/04 2014.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Identity Mappings in Deep Residual Networks," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 27-30, 2016, pp. 630-645.

[48] E. Ostertagova and O. Ostertag, *The Simple Exponential Smoothing Model*. 2011.

[49] "Batch Normalization and ResNets," Accessed [20.02.2024], Available: https://www.neuralception.com/objectdetection-batchnorm/.

[50] T. Booranawong and A. Booranawong, "Double exponential smoothing and Holt-Winters methods with optimal initial values and weighting factors for forecasting lime, Thai chili and lemongrass prices in Thailand," *Engineering and Applied Science Research,* vol. 45, pp. 32-38, 02/28 2018.

[51] "Forest Fire Images," Accessed [10.11.2023], Available: https://www.kaggle.com/datasets/mohnishsaiprasad/forest-fire-images.

[52] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data,* vol. 6, no. 1, p. 60, 2019/07/06 2019.

[53] P. Simard, D. Steinkraus, and J. Platt, *Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis*. 2003, pp. 958-962.

[54] T. G. Dietterich, "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms," *Neural Computation,* vol. 10, no. 7, pp. 1895-1923, 1998.

[55] D. Powers, "Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation," *Mach. Learn. Technol.,* vol. 2, 01/01 2008.

[56] S. Adhikari, S.-L. Normand, J. Bloom, D. Shahian, and S. Rose, "Revisiting performance metrics for prediction with rare outcomes," *Statistical Methods in Medical Research,* vol. 30, p. 096228022110387, 09/01 2021.

[57] J. Lee and K.-i. Hwang, "YOLO with adaptive frame control for real-time object detection applications," *Multimedia Tools and Applications,* vol. 81, 09/18 2021.