الجمهورية الجزائرية الديمقراطية الشعبية

People's Democratic Republic of Algeria

وزارة التعليم العالي والبحث العلمي

Ministry of Higher Education and Scientific Research

جامعة زيان عاشور بالجلفة

Ziane Achour University of Djelfa

Department: Electronics and communication

كلية العلوم والتكنولوجيا

Faculty of Sciences and Technology

Order N° : . …. / 2026

Defense authorization N° ……..…/2026

DOCTORAL THESIS

3rd Cycle Doctoral (D-LMD)

Presented by

SIHAM KHIRANI

With a view to obtain the doctoral degree in 3rd Cycle Doctoral (D-LMD)

Branch: Telecommunication

Specialty: communication and wireless networks

Topic

Wildfire detection using computer vision techniques

Supported, on 02/06/ 2026, before the jury composed of:

Last and first name	Grade	Institution of affiliation	Designation
Pr. Elhadi Mehallel	Prof	university of Djelfa	President
Dr.Abdelkerim Souahlia	MCA	university of Djelfa	Supervisor
Pr.Abdelhalim Rabehi	Prof	university of Djelfa	Co-Supervisor
Dr.Ali Teta	MCA	university of Djelfa	Examiner
Dr.Hamza Kheddar	MCA	university of Medea	Examiner

Djelfa University, FST - 2026

Acknowledgments

First and foremost, all praise is due to Allah, abundant and blessed. Praise be to Allah, the source of all great gifts, who does not let a tear fall from my eyes without wiping it with the most beautiful moments; who mends every break in my heart and every crack in my soul with His finest blessings; who does not let my hands be humbled in prayer without filling them with what I desire and more; and who, every time, directs my feelings toward success and purity instead of resentment and revenge. Praise be to Allah until praise reaches its ultimate limit.

I would like to express my deepest gratitude to my supervisor, Dr. Abdelkerim Souahlia, for his guidance, encouragement, and unwavering support throughout this research. His expertise, insightful feedback, and patience were instrumental in shaping this work. I am also grateful to my co-supervisor, Dr. Rabehi Abdelhalim, for his honest, straightforward feedback and constructive criticisms.

I would also like to acknowledge the support of the Laboratory of Telecommunications and Smart Systems, Faculty of Sciences and Technology, University of Djelfa, for providing the resources and facilities necessary to carry out this work.

I am grateful to the members of my thesis committee for their constructive comments, valuable suggestions, and dedication, which helped improve the quality of this research.

Finally, I extend my sincere thanks to my colleagues and friends for their collaboration, insightful discussions, and constant encouragement during the challenges of this journey. Special thanks go to Chaouli Zahra.

Dedication

I dedicate this work to my beloved family, whose unwavering support, patience, and encouragement have been my source of strength throughout this journey.

To my mother, because of her I am here, and to my father, who witnessed only my first steps when I started walking and never saw the other steps of my life. To My sisters and my brothers.

To my husband, for his endless love and understanding, and to my daughters, whose smiles inspire me every day.

To all those who challenged me, doubted me, or caused me difficulties. Their words and actions became a source of motivation, driving me to discover my true potential and realize, by the grace of God, that I am capable.

Abstract

Forests are essential components of the Earth's ecosystem, providing biodiversity, regulating the climate, and supplying critical natural resources. However, forest fires are an increasing global threat, intensified by climate change, and pose significant risks to ecosystems, human life, and property. Traditional detection methods are often limited in speed, accuracy, and cost-effectiveness, emphasizing the need for real-time and automated solutions. This thesis proposes a methodology for forest fire detection using transfer learning (TL) with pre-trained convolutional neural networks (CNNs). Six leading architectures were experimentally evaluated to examine the impact of different configurations, identify the optimal combination for accurate detection, and compare performance with state-of-the-art methods. Additionally, a low-complexity CNN architecture was designed, enhanced with selected improvement techniques, and deployed on resource-constrained devices such as the Raspberry Pi 5. Experimental results demonstrate that TL models achieve high detection performance, whereas the lightweight CNN is more suitable for real-time deployment, offering a practical solution for wildfire monitoring on embedded platforms. These approaches provide a solid foundation for developing effective early warning systems and disaster management strategies.

Keywords

Forest fire detection, Deep learning, Convolutional Neural Networks, Low-complexity CNN, Transfer Learning, Raspberry Pi 5.

الملخص

تُعد الغابات من المكونات الأساسية للنظام البيئي للأرض، حيث توفر التنوع البيولوجي، تنظم المناخ، وتزود بالموارد الطبيعية الحيوية. ومع ذلك، تُشكّل حرائق الغابات تهديدًا عاما متزايدًا. تتفاقم الحرائق بفعل التغيرات المناخية، وتشكل مخاطر كبيرة على النظم البيئية وحياة الإنسان والممتلكات. تعاني طرق الكشف التقليدية عن الحرائق من محدودية في السرعة والدقة والجدوى الاقتصادية، مما يبرز الحاجة إلى حلول آلية وفورية للكشف المبكر . تقترح هذه الأطروحة منهجية للكشف عن حرائق الغابات باستخدام التعلّم بالنقل للشبكات العصبونية الالتفافية المدرَّبة مسبقًا . تم تقييم ستة من أبرز النماذج تجريبيًا على مجموعتي بيانات معياريتين لدراسة تأثير التكوينات المختلفة، وتحديد أفضل تركيبة للكشف الأمثل، ومقارنة الأداء مع الأساليب المعتمدة حديثا. بالإضافة إلى ذلك، تم تصميم شبكة عصبونية منخفضة التعقيد، وتحسينها باستخدام تقنيات تعزيز مختارة، ونشرها على منصات ذات موارد محدودة مثلRaspberry Pi 5 . تُظهر النتائج التجريبية أن نماذج التعلّم بالنقل تحقق أداءً عاليًا، في حين تُعد الشبكة منخفضة التعقيد أكثر ملاءمة للتطبيق الفوري، مما يوفر حلًا عمليًا لمراقبة حرائق الغابات على الأنظمة المدمجة. وتوفر هذه المقاربة أساسًا قويًا لتطوير أنظمة إنذار مبكر فعّالة واستراتيجيات إدارة الكوارث.

الكلمات المفتاحية

اكتشاف حرائق الغابات، التعلم العميق، الشبكات العصبية الالتفافية، شبكة منخفضة التعقيد، التعلم بالنقل، راسبيري باي 5.

Table of contents

List of figures

List of tables

List of acronyms

ACNet	Custom Attention Connected Network
Adam	Adaptive Moment Estimation
AI	Artificial Intelligence
AUC	Area Under The Curve
AVHRR	Advanced Very High-Resolution Radiometer
CNNs	Convolutional Neural Networks
CO_2	Carbon Dioxide
ConvL	Convolutional Layer
CPU	Central Processing Unit
Def-Pooling	Deformation-Constrained Pooling
DL	Deep Learning
DT	Decision Trees
FCL	Fully Connected Layer
FLOPS	Floating Point Operations
GAP	Global Average Pooling
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
KNN	k-nearest Neighbors
LR	Logistic Regression
ML	Machine Learning
MODIS	Moderate Resolution Imaging Spectroradiometer
MSE	Mean Squared Error
NB	Naïve Bayes
NIN	Network in Network
NO_2	Nitrogen Dioxide
O_3	Ozone

PM	Particulate Matter
RAISR	Rapid And Accurate Image Super Resolution
RAM	Random Access Memory
RBFNs	Radial Basis Function Networks
ReLU	Rectified Linear Unit
RF	Random Forest
RGB	Red Green Blue
RMSprop	Root Mean Square Propagation
ROC	Receiver Operating Characteristic
SE	Squeeze And Excitation Blocks
SGDM	Stochastic Gradient Descent With Momentum
SHAP	Shapley Additive Explanations
SPP	Spatial Pyramid Pooling
SVM	Support Vector Machine
TF	TensorFlow
TFLite	TensorFlow Lite
TL	Transfer Learning
UAV	Unmanned Aerial Vehicle
YCbCr	Luminance–Chrominance Color Spac

1. Introduction

1.1. Overview

Forests serve as one of the most critical components of the Earth's environmental system, playing a foundational role in promoting ecological balance and planetary health. Specifically, forest ecosystems are integral to the richness of terrestrial biodiversity, often harboring a wider variety of species than most other land environments. They span more than 30% of the planet's landmass. One of their most vital contributions is the sequestration of carbon, forests absorb and store large amounts of carbon dioxide from the atmosphere, which directly contributes to lowering CO_2 levels and helps to slow the pace of global warming and mitigate the broader impacts of climate change [1]. This natural process supports international climate goals by reducing greenhouse gas concentrations and buffering the planet against temperature extremes. Beyond their role in carbon management, forests perform several other vital ecological functions. They help regulate the hydrological cycle by facilitating rainfall distribution and groundwater replenishment. Their root systems stabilize soil structures, preventing erosion and reducing the risk of landslides, particularly in hilly or mountainous terrains. Moreover, forests act as protective buffers against natural disasters such as floods and droughts. Ecologically, forests are biodiversity hotspots. They provide shelter and living conditions for approximately 80% of all known terrestrial species of flora and fauna, making them essential for the conservation of wildlife and the maintenance of biological diversity on a global scale. These rich ecosystems not only support animal and plant life but also provide essential resources for human survival, such as oxygen for respiration, food for sustenance, and wood for fuel, construction, and manufacturing [2].

However, these crucial ecosystems are increasingly threatened by forest fires, which can rapidly destroy biodiversity and disrupt environmental balance. Often, forest fires go undetected until they have already expanded across large regions, making suppression efforts extremely challenging or even unfeasible. The environmental and atmospheric consequences are profound, as forest fires are responsible for nearly 30% of global carbon dioxide (CO_2) emissions [3]. These fires also influence local climatic conditions, contribute significantly to global warming, and may drive rare plant and animal species to extinction.

In recent years, numerous large-scale forest fires have resulted in devastating ecological and economic damage across continents worldwide.

The consequences of wildfires can be categorized into two primary types: short-term and long-term effects, each of which requires ongoing monitoring. Short-term impacts refer to the immediate effects that occur during or shortly after a wildfire. These include property damage, injuries, displacement of wildlife from their habitats, vegetation loss, deterioration of air quality, and the significant costs associated with firefighting efforts [4][5]. Additionally, wildfires contribute to acute short-term effects by releasing harmful gases such as nitrogen dioxide (NO_2) and ozone (O_3), along with producing dangerously high concentrations of fine particulate matter (PM2.5), which can severely affect respiratory health, leading to increased morbidity and mortality rates. While these immediate impacts are devastating, long-term impacts of wildfires encompass ecological changes, including shifts in the composition of plant and animal species, soil erosion, land degradation, and a decline in water quality. These fires can compromise the land's capacity to retain water, raising the risk of flash floods [6]. The aftermath of a wildfire can lead to irreversible alterations in the environment, disrupting the natural water cycle and damaging ecosystems, which may take years, if not decades, to recover. Furthermore, wildfires have social and psychological impacts that extend beyond their immediate environmental effects. The devastation caused by wildfires often leaves communities grappling with physical destruction, loss of livelihoods, and emotional trauma. Those affected by the fires may experience long-term psychological distress, especially if they have lost homes or loved ones in the disaster.

Given the severe environmental, economic, and social consequences of wildfires, significant research efforts have been devoted to developing effective wildfire detection and monitoring techniques.

However, despite these efforts, forest fire detection remains a challenging task. Variations in lighting conditions, smoke dispersion patterns, background complexity, weather conditions, and similarities between fire-like objects and actual flames often lead to false alarms or missed detections. Moreover, many existing solutions require high

computational resources, making real-time deployment on low-cost edge devices difficult. These challenges indicate that further research is needed to develop accurate, lightweight, and real-time wildfire detection systems suitable for practical field deployment.

1.2. Motivation

The increasing frequency and severity of wildfires have made early and accurate fire detection a critical requirement. The limitations of traditional monitoring approaches highlight the need for intelligent, real-time, and cost-effective wildfire detection systems. This section outlines the key motivations for adopting DL–based solutions to address these challenges.

1.2.1. Increasing threat of wildfires due to climate change

Building on the global concern raised by climate change, studies have shown a steady rise in wildfire intensity and frequency worldwide. A study published in Nature Ecology & Evolution reported that since 2003, the occurrence of extreme wildfires has more than doubled, with six of the most severe fire seasons recorded in the past seven years [7]. These fires not only devastate vast forested areas but also release substantial quantities of greenhouse gases, thereby further exacerbating global warming.

Rising global temperatures and prolonged periods of drought have created drier and more combustible environments, which significantly heighten the risk of forest fires [2][8]. These fires can spread rapidly and uncontrollably, leading to the large-scale destruction of vegetation, the loss of wildlife habitats, and severe disruptions to ecosystems. Additionally, forest fires have profound social and economic consequences, particularly for communities

that rely on forest resources for their livelihoods, including indigenous populations, rural dwellers, and those working in forestry and agriculture. This trend is further illustrated in Fig. 1-1, which presents the wildfire-burned land area in Mediterranean countries in 2024.

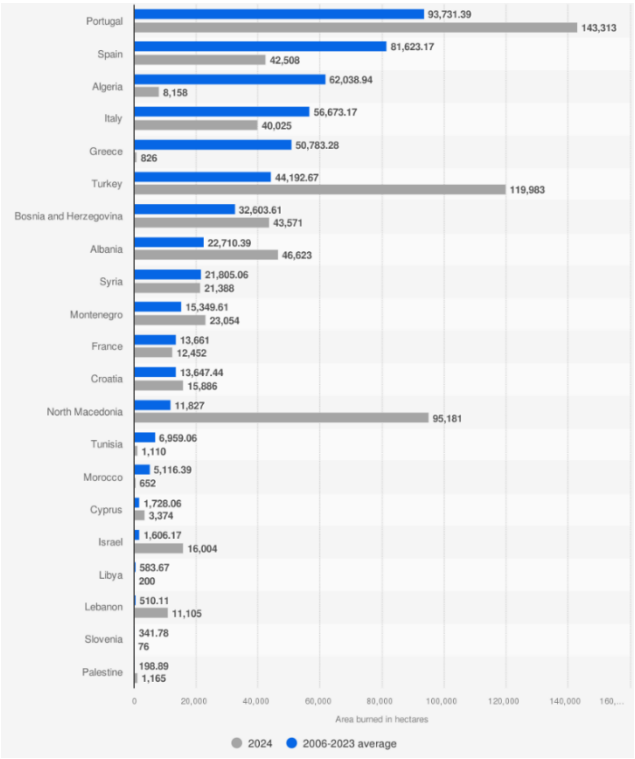


Fig. 1-1.Wildfire-Burned land area in Mediterranean countries, 2024 [9] .

Algeria, as part of this climate-sensitive region, has witnessed a series of severe wildfires over the past five years, reflecting this broader trend:

- **2020:** Algeria experienced significant forest fires that impacted various regions across the country. Approximately 44,000 hectares of forested land were destroyed by these fires, posing serious threats to biodiversity and the environment. Several individuals were arrested on suspicion of arson in connection with these incidents [10].

- **2021**: The northern Kabylia region experienced some of the most destructive wildfires, affecting 13 provinces. These fires led to the tragic loss of at least 90 lives, including 28 members of the People's National Army, who participated in firefighting operations and rescued over 100 individuals in Bejaïa and Tizi-Ouzou [11].
- **2022:** Wildfires resulted in at least 38 fatalities and caused substantial human and material damage. The provinces of El Tarf and Sétif were among the most severely affected [12].

- **2023:** Fires raging across Algeria claimed the lives of over 30 people, including 10 soldiers, and forced the evacuation of hundreds of residents along the Mediterranean coastal region [13].

On average, Algeria loses approximately 20,000 hectares of forest land annually to wildfires. The country's forests, covering about 4.1 million hectares, are predominantly composed of Aleppo pine (68%) and cork oak (21%), both of which are highly flammable. The increasing frequency and severity of wildfires pose significant challenges to Algeria's biodiversity, economy, and public health.Fig.1-2. illustrates the annual number of wildfires and the corresponding burned areas in Algeria from 2006 to 2024 [14].

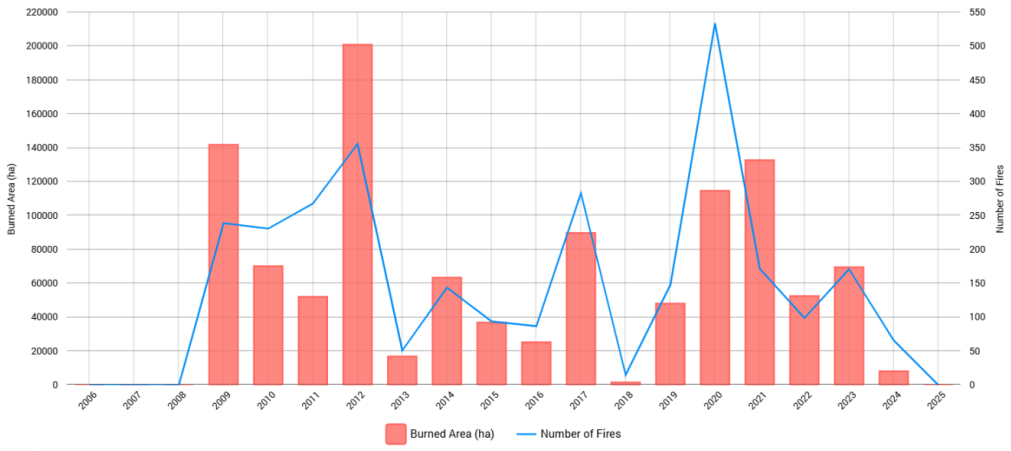


Fig. 1-2.Number of wildfires and total burned area in Algeria between 2006 and 2024 [14].

1.2.2. The limitations of traditional detection methods

Early detection of forest fires is crucial for minimizing the spread of wildfires, reducing environmental damage, and lowering firefighting costs. Rapid and precise alerts enable more efficient responses, significantly limiting the potential devastation. Historically, conventional fire monitoring relied heavily on human observation, where individuals were stationed in lookout towers to detect and report fire incidents [15]. However, human-based monitoring is prone to inaccuracies due to fatigue, environmental conditions, and limited field of view.

To overcome these limitations, technological advancements introduced mechanical tools such as wireless sensors for detecting smoke, temperature changes, and air transparency [16]. Although effective in small areas, sensor-based systems are constrained by challenges like limited coverage, high installation density requirements, battery limitations, and high costs, making them impractical for large-scale forest monitoring. Similarly, satellite-based systems offer the ability to monitor vast territories but suffer from low image resolution, long scanning intervals, and sensitivity to weather conditions like cloud cover[17] .

The rapid development of digital cameras, wireless communication, and video processing techniques has shifted the focus toward computer vision-based fire detection systems [18]. Image-based approaches offer significant benefits, such as early detection, improved accuracy, flexible deployment, and the ability to monitor extensive areas efficiently through the analysis of visual data. Typically, image-based fire detection involves three main stages: preprocessing, feature extraction, and classification. Traditional methods depend on manually crafted features, requiring expert knowledge and intensive labor. Moreover, these handcrafted features often struggle to distinguish fire from visually similar phenomena, leading to reduced accuracy and poor generalization.

1.2.3. The need for real-time and cost-effective detection

Real-time detection of forest fires is not just a technological goal, it is a critical requirement for saving lives, minimizing environmental losses, and optimizing resource deployment during firefighting operations. Fires can escalate in minutes, especially under dry and windy conditions, making any delay in detection potentially catastrophic. Therefore, systems capable of delivering instant or near-instant alerts are essential for effective wildfire management. However, achieving real-time performance comes with considerable challenges. High computational costs, limited internet access in remote areas, and hardware constraints on edge devices can all hinder the deployment of real-time solutions. Furthermore, the cost of deploying and maintaining high-tech systems such as thermal cameras, satellite imagery platforms, or dense sensor networks is often prohibitively high, particularly for developing countries or regions with vast forested areas.

Given the growing severity and frequency of forest fires, there is an urgent need for intelligent, automated systems capable of detecting fires in their early stages with minimal false alarms. DL-based approaches offer a scalable, cost-effective, and accurate solution to this problem, particularly when integrated with real-time image feeds from surveillance cameras, drones, or other visual sensors. In recent years, CNNs have revolutionized image-based fire detection by enabling automatic feature extraction directly from data, enhancing classification accuracy, and improving system robustness. CNNs have demonstrated outstanding success in various domains such as visual search [19], autonomous driving [20], and medical diagnosis [21]. Their ability to learn complex, hierarchical features makes them highly suitable for wildfire detection tasks. However, training deep CNNs from scratch is a challenging and time-consuming process, requiring careful tuning of hyperparameters such as the number of layers, filter sizes, learning rates, and optimizers.

Moreover, large and complex CNN models can impose significant computational burdens, leading to slower inference times, an undesirable drawback for real-time forest fire detection. As a result, TL has emerged as an efficient alternative, allowing researchers to leverage pre-trained models such as SqueezeNet, GoogLeNet, ResNet, MobileNet, and others. TL not only reduces training time but also enhances performance, particularly when labeled data is scarce [22], [23].

1.3. The objectives of this research

The overarching objective of this research is to enhance the effectiveness and efficiency of DL-based approaches for forest fire detection through systematic evaluation, architectural innovation, and practical deployment. To achieve this aim, the study is guided by the following specific objectives:

- **Comprehensive Evaluation of Pre-trained CNN Models:** To systematically evaluate several widely used pre-trained CNN architectures for the task of forest fire detection. This objective focuses on analyzing the influence of key hyperparameters, including learning rate, batch size, and optimizer type, on model performance, with the aim of identifying the most effective configurations and understanding network behavior across different dataset sizes.

- **Design of a Low-Complexity CNN Architecture:** To design and develop a novel lightweight CNN specifically optimized for forest fire detection. The proposed architecture seeks to minimize computational cost while maintaining high classification accuracy, emphasizing simplicity and efficiency to enable real-time performance and suitability for deployment on embedded or resource-constrained devices.

- **Practical Deployment and Validation:** To implement and evaluate the proposed CNN on a resource-constrained edge device, thereby examining its real-world applicability in low-power, real-time environments. This objective aims to validate the feasibility and effectiveness of the proposed model for intelligent, on-device wildfire detection under limited computational resources.

1.4. Contributions

To determine the optimal performance of various pre-trained networks for wildfire detection, an extensive series of experiments was conducted on two benchmark datasets: DeepFire and Forest Fire Images dataset. Six widely used pre-trained models, namely AlexNet, GoogLeNet, SqueezeNet, ResNet18, EfficientNetB0, and MobileNetV2, were evaluated by systematically adjusting key hyperparameters such as learning rate, batch size, and optimizer. These models achieved generally strong results, although their classification accuracy and processing time varied depending on the selected hyperparameter configurations.

Beyond these pre-trained models, this study demonstrates that a carefully designed lightweight CNN can achieve comparable or even superior accuracy with substantially lower computational cost. For this purpose, a low-complexity CNN network architecture was proposed, consisting of three convolutional layers and one fully connected layer with two output neurons. The model achieved high accuracy on the DeepFire dataset and demonstrated strong generalization capability on the larger Forest Fire Images dataset.

In addition, several common enhancement techniques were examined, including Global Average Pooling (GAP), Dropout, Squeeze and Excitation blocks (SE), and L2 regularization, the proposed CNN was also deployed on a Raspberry Pi 5 device. Its performance was compared with that of other pre-trained networks and models that

included improvement techniques, demonstrating the feasibility and effectiveness of the proposed approach for on-device wildfire detection.

Overall, this comprehensive study emphasizes the strong potential of low-complexity CNN architectures as efficient and reliable alternatives to heavy TL and hybrid models, especially in situations where computational resources are limited or where real-time performance is essential.

1.5. Thesis structure

The ramain of the thesis is structured as follows:

Chapter 2: presents an in-depth review of the essential concepts, core technologies, and previous research related to DL approaches for forest fire detection. It establishes the theoretical background required for this study by first introducing the fundamental principles of CNNs and TL, then gradually addressing their specific applications in wildfire detection.

Chapter 3: provides a detailed description of the experimental framework based on TL for forest fire detection. It presents some practical related choices such as the datasets used and the evaluation metrics. The chapter also discusses the evaluation of six pre-trained CNN models including AlexNet, GoogLeNet, SqueezeNet, ResNet18, EfficientNetB0, and MobileNetV2, the selection of critical hyperparameters, and the performance metrics adopted for assessment. In addition, it presents the hardware and software resources utilized and includes a comprehensive comparison of the results to evaluate the performance and reliability of the tested networks.

Chapter 4: presents the design, training, and evaluation of a custom low-complexity CNN for forest fire detection. It assesses the model's performance on two benchmark datasets and examines the impact of several enhancement techniques, including Global Average Pooling (GAP), Dropout, Squeeze-and-Excitation blocks (SE), and L2 regularization, on its generalization and robustness. The chapter also evaluates the deployment of the proposed solution on a Raspberry Pi 5 under resource-constrained conditions. Finally, a

comparative analysis highlights the proposed architecture's ability to achieve an effective balance between accuracy, computational efficiency, and deployment feasibility.

Chapter 5: concludes the thesis by summarizing the main findings and contributions of this research. It highlights the significance of DL in improving the effectiveness of forest fire detection systems and emphasizes the advantages of lightweight models for real-time applications. The chapter also provides practical recommendations for deploying such

models in embedded systems and outlines potential directions for future research aimed at further enhancing intelligent and scalable wildfire monitoring solutions.

1.6. Publications from the Thesis

Some of the material presented in this thesis were published in the following papers:

1.6.1. Journal publications

Khirani, S., Souahlia, A., Rabehi, A., Bourennane, M., Guermoui, M., Tibermacine, I., Rabehi, A. (2026). Advanced evaluation of pre-trained CNN models for accurate forest fire detection. *Natural Hazards* **122**, 234 (2026). https://doi.org/10.1007/s11069-026-07976-3

Khirani, S., Souahlia, A., Rabehi, A., Bourennane, M., Latreche, B., Dhelim, S. An Optimized Lightweight Convolutional Neural Network Architecture for Real-Time Forest Fire Detection in Resource-Constrained Edge Devices. Submitted to Journal of Real-Time Image Processing.

1.6.2. International conference publications

Khirani, S., Souahlia, A., & Rabehi, A. (2023, November). Forest Fire Detection Using a Low Complex Convolutional Neural Network. In 2023 2nd International Conference on Electronics, Energy and Measurement (IC2EM) (Vol. 1, pp. 1-6). IEEE. doi: 10.1109/IC2EM59347.2023.10453317.

2. Literature review

2.1. Introduction

This chapter presents a comprehensive review of the fundamental concepts and methodological components that underpin this research. It begins with an overview of DL principles, with particular emphasis on CNN and TL, as well as the main training parameters and optimization strategies involved in model development. In addition, the chapter includes a detailed review of related work in the field of forest fire detection, highlighting recent advances, existing challenges, and research gaps. Together, these elements establish the theoretical and technical foundation necessary to understand the proposed methodology and its contribution to improving the performance and efficiency of forest fire detection systems.

2.2. Background

2.2.1. Convolutional neural networks

CNNs are a type of DL model composed of multiple layers of neurons organized in a hierarchical structure. Their primary objective is to learn meaningful representations of input data by extracting increasingly complex features at each layer. CNNs are particularly effective for computer vision tasks due to their ability to autonomously learn important features directly from data [24]. This feature extraction process identifies and organizes key attributes within the input, enabling the network to learn efficiently. Unlike traditional ML methods that require manual feature engineering [25], CNNs offer a significant advantage by automatically learning task-specific features [26].

The term "convolutional" in CNNs originates from the convolutional layers that form the core of the architecture. A convolution operation is a mathematical process that identifies commonalities between two different data sets. Within CNNs, these operations are executed using filters that move, or "slide", across the input during training, producing what is known as a feature map. After each convolution, the output passes through an activation function, and the resulting feature maps are then combined to form the final output of the convolutional layer [27].

As illustrated in Fig.2-1, a typical CNN architecture begins with an input layer where data is represented numerically, such as pixel intensity values in an image. The input is then passed through a series of convolutional layers, where filters (also called kernels) slide across the input matrix to detect local patterns like edges, textures, and colors. After each convolution operation, a non-linear activation function, commonly ReLU (Rectified Linear Unit), is applied to enhance learning capacity.

Following the convolutional layers, pooling layers are introduced to downsample the feature maps, which reduces spatial dimensions, lowers computational cost, and helps prevent overfitting. This process retains the most important features while discarding redundant information. After multiple rounds of convolution and pooling, the resulting feature maps are flattened into a one-dimensional vector and passed to one or more fully connected (dense) layers, which perform higher-level reasoning. The final output layer provides the model's prediction, such as class probabilities in classification tasks.

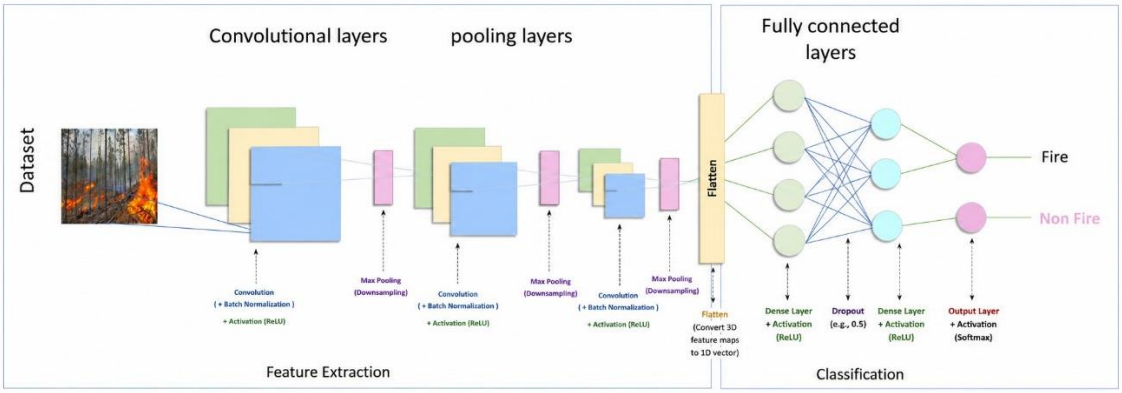


Fig. 2-1.CNN architecture.

2.2.1.1. The convolutional layers

In convolutional layers, CNNs apply different kernels across both the original input image and the resulting intermediate feature maps, producing a variety of new feature maps as illustrated in Fig.2-2 The convolution operation provides several key advantages [28]: it reduces the number of parameters through a weight-sharing mechanism within the same feature map, captures relationships between neighboring pixels through local

connectivity, and achieves a level of spatial invariance to object positioning. Because of these benefits, several influential studies have opted to replace fully connected layers with convolutional ones to speed up the learning process [29][30]. A notable strategy in this context is the Network in Network (NIN) method [31], which substitutes traditional convolutional layers with small multilayer perceptrons that contain fully connected layers and nonlinear activation functions. This approach replaces linear filters with nonlinear neural networks and has demonstrated effective results in image classification tasks.

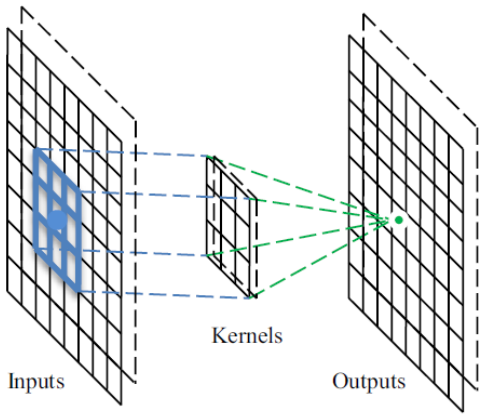


Fig. 2-2.Functionality of the convolutional layer [32].

2.2.1.2. Batch normalization (BN)

normalizes the activations of a layer using the mean and variance of each mini-batch. This process stabilizes and accelerates training, allows higher learning rates, and reduces sensitivity to weight initialization. BN also provides a regularization effect that can help reduce overfitting[33].

2.2.1.3. The rectified linear unit (ReLU)

The Rectified Linear Unit (ReLU) is a nonlinear activation function (1) defined as:

$$f(x)=\max(0,x) \quad (1)$$

It sets negative values to zero while keeping positive values unchanged. ReLU is computationally efficient and helps alleviate the vanishing gradient problem, making it the most widely used activation function in CNNs [34].

2.2.1.4. Pooling layers

Pooling layers are typically applied after convolutional layers in CNNs to reduce the spatial dimensions of feature maps and decrease the number of network parameters. These layers contribute to translation invariance because they compute over local neighborhoods of the input, thus preserving spatial hierarchies. The most common types are max pooling and average pooling, which summarize local regions in the feature map. For example, as shown in Fig.2-3 , applying a 2×2 max pooling filter with a stride of 2 to an 8×8 input reduces it to a 4×4 output. Max pooling selects the highest value in each region, while average pooling computes the mean [35][36]. Due to its advantages in feature selection and generalization, max pooling is widely used in modern CNN implementations [37][34].

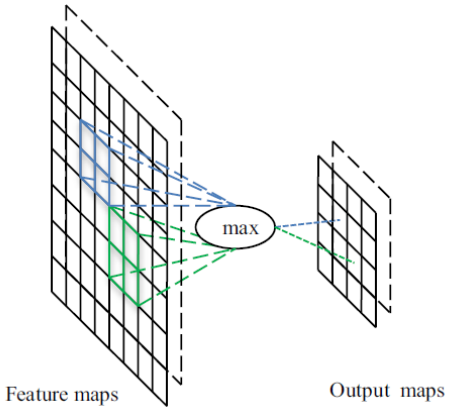


Fig. 2-3. Functionality of the pooling layer [32].

In addition to these conventional methods, several advanced pooling strategies have been proposed to address specific limitations or enhance network flexibility. These include stochastic pooling, spatial pyramid pooling, and deformation-constrained pooling, each serving distinct purposes within the architecture.

- **Stochastic pooling** : introduces randomness into the pooling process by selecting an activation from each pooling region based on a probability distribution. Unlike deterministic pooling methods like max or average pooling, stochastic pooling helps prevent overfitting by mimicking the effect of having multiple slightly varied inputs. This probabilistic selection allows for better generalization to unseen data and is especially useful in preventing the network from memorizing training samples [28][38].

- **Spatial Pyramid Pooling (SPP):** allows CNNs to handle input images of arbitrary sizes by generating fixed-length feature representations. Unlike traditional CNN architectures that require fixed-size inputs, SPP divides the input feature map into bins of various sizes and applies pooling within each bin. This results in a flexible representation that captures spatial information across multiple scales and enables the model to work effectively with images of different dimensions and aspect ratios [39] .
- **Deformation-Constrained Pooling (Def-Pooling):** is designed to improve the handling of geometric variations and deformations in visual patterns. Traditional pooling methods, while helpful for general deformation tolerance, do not explicitly learn spatial constraints. Def-pooling addresses this by modeling the deformation of object parts and integrating this information into the pooling process. It can be used at various abstraction levels within the CNN to enhance its ability to recognize deformed or misaligned objects [40].

2.2.1.5. The flatten layer

converts multidimensional feature maps into a one-dimensional feature vector before they are passed to fully connected layers. Although it contains no trainable parameters, it enables the transition from feature extraction to classification stages in CNNs [41].

2.2.1.6. Fully-connected layers

After the Flatten layer converts the multidimensional feature maps into a one-dimensional feature vector, one or more fully connected (FC) layers are typically employed, as illustrated in Fig. 2-4. These layers resemble traditional artificial neural networks, where each neuron is connected to all neurons in the preceding layer. Their primary function is to integrate the extracted features and learn high-level representations that facilitate the final classification process. The resulting fixed-length feature vector can be used either to classify images into predefined categories [34] or to serve as a feature representation for subsequent tasks [42]. Although it is uncommon to modify the structure of fully-connected layers, TL offers a notable exception in this approach, models trained on datasets like ImageNet [34] retain the earlier learned parameters but substitute the final fully-connected

layer with one or more new ones to suit new recognition tasks. One of the primary drawbacks of fully connected layers is their high parameter count, which significantly increases the computational cost during training. As a solution, some architectures aim to reduce or eliminate these layers. For instance, GoogLeNet [29] achieves a deeper and broader network architecture while maintaining a fixed computational cost by replacing fully connected layers with sparsely connected ones.

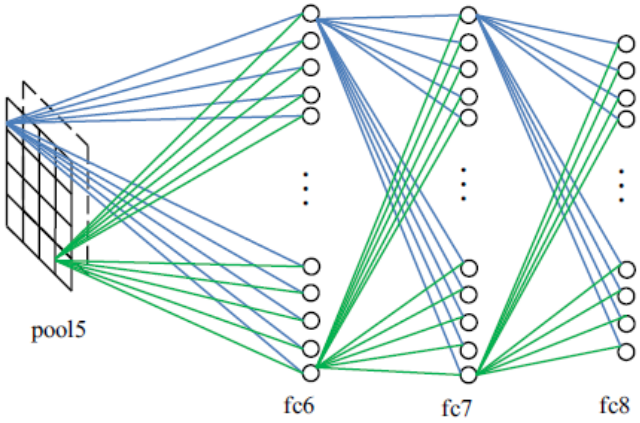


Fig. 2-4.Functionality of the fully-connected layer [32].

- **Key Concepts in CNNs**

When working with CNNs, it is important to understand several fundamental concepts [43]:

- **Filters**: Filters act as feature detectors. They help identify visual elements in an image such as edges or lines. By applying convolutional operations, filters generate feature maps that highlight structural details like object boundaries [44].

- **Local Perception**: CNNs do not require each neuron to process the entire image. Instead, each neuron focuses on a local receptive field, enabling the network to capture

spatial hierarchies of features. This approach allows CNNs to model local dependencies effectively [44].

• **Sub-sampling**: This refers to reducing the spatial dimensions of the feature maps typically achieved through pooling layers. Sub-sampling helps decrease the computational burden and provides translation invariance, meaning the model becomes less sensitive to slight changes in the position of features [44].

- **Fully Connected Layers**: Located at the final stage of the network, these layers receive the most relevant high-level features extracted by earlier layers and use them to make classification decisions by assigning class scores to inputs. When selecting a CNN model, there are several distinct architectural designs available. The choice of architecture can significantly influence the outcomes, depending on the specific problem being addressed, even when the same training and validation datasets are used.

2.2.1.7. Dropout

is a regularization technique that randomly deactivates a fraction of neurons during training. This prevents overfitting by reducing the dependence between neurons and encourages the network to learn more robust features. During testing, all neurons remain active[34].

2.2.1.8. Softmax layer

The Softmax layer is typically used as the final layer in a CNN for multi-class classification tasks. It transforms the output scores of the previous layer into a probability distribution, where each value represents the likelihood of the input belonging to a specific class. The predicted class is the one with the highest probability. By producing normalized probabilities that sum to one, Softmax facilitates the interpretation of network predictions and improves classification performance [45].

The Softmax function (2) is given by:

$$P(y = i) = \frac{e^{Zi}}{\sum_{j=1}^{k} e^{zj}} \qquad (2)$$

In summary, CNNs are highly effective for learning both low-level and high-level features in a hierarchical manner, making them powerful for applications such as image recognition, object detection, and image classification.

2.2.2. Transfer learning

Transfer Learning (TL) is a machine learning technique that improves performance on a target task by leveraging knowledge acquired from a related source task [46]. It has been widely adopted in fields such as computer vision, natural language processing, and speech recognition, particularly when labeled data for the target task are limited. By reusing previously learned knowledge, TL helps reduce overfitting and improves model generalization. One of the earliest forms of TL is feature-based transfer learning, where features learned from a source task are transferred to a target task [47]. For example, features extracted from models pre-trained on the ImageNet dataset can be reused for tasks such as object detection. Another common approach is fine-tuning, in which a pre-trained model is further trained on a smaller, task-specific dataset to improve performance. This strategy is widely used in both computer vision and natural language processing applications. Transfer learning can be categorized into three main types: Inductive Transfer Learning, where the source and target tasks differ and labeled target data are available; Transductive Transfer Learning, where the tasks are the same but the data domains differ; and Unsupervised Transfer Learning, where neither domain contains labeled data. In addition, Multi-Task Learning enables a model to learn several related tasks simultaneously by sharing common representations across tasks. Another classification of TL includes feature-based, instance-based, and parameter-based approaches. Instance-based methods transfer selected data samples between domains, while parameter-based methods transfer learned model weights and adapt them to the target task through fine-tuning. These approaches allow models to benefit from previously acquired knowledge while reducing training requirements.TL has become particularly valuable in forest fire detection. Instead of training deep learning models from scratch, pre-trained models are adapted using labeled fire and non-fire images. This enables the reuse of general visual features such as edges, textures, and shapes, which are then refined for the specific task. As a result, TL improves detection accuracy, reduces computational costs, and requires fewer labeled samples than conventional deep learning methods.Furthermore, to address the computational challenges associated with large-scale datasets, distributed learning approaches have been proposed. By partitioning datasets into smaller subsets and processing them independently, these

methods reduce the complexity of kernel-based computations and improve scalability without compromising performance.Overall, Transfer Learning has become a fundamental technique in modern machine learning, enabling efficient knowledge transfer across domains while reducing data and computational requirements. Its effectiveness and flexibility make it especially suitable for real-world applications where resources and labeled data are limited (Fig.2-5).

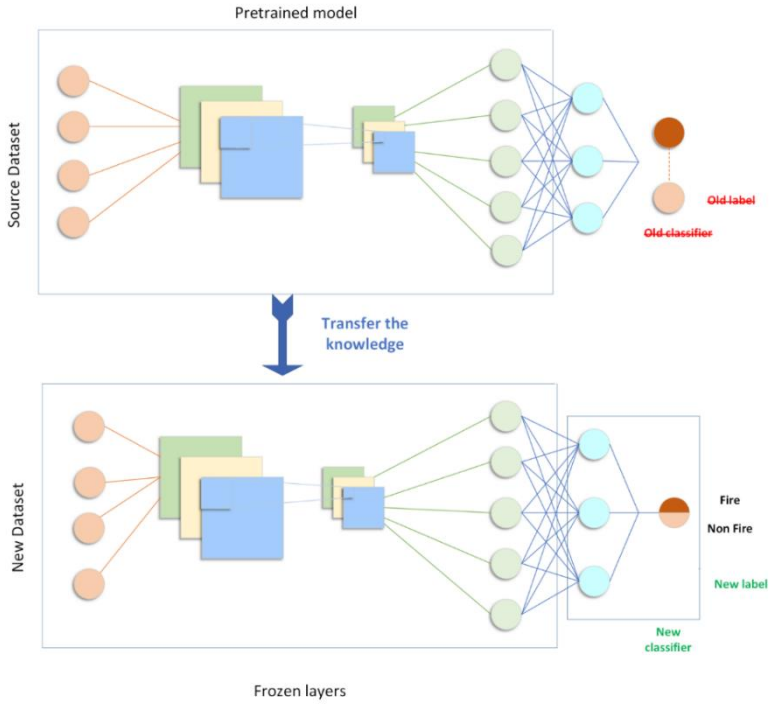


Fig. 2-5.Transfer learning technique.

2.2.3. Selected CNN architectures

In this work, we focus on developing a DL model for wildfire detection using TL techniques from computer vision architectures including AlexNet, GoogLeNet, SqueezeNet, ResNet18, EfficientNetB0 and MobileNetV2.

2.2.3.1. AlexNet

AlexNet represents a landmark development in the field of DL and CNNs. Introduced in 2012 by Krizhevsky, Sutskever, and Hinton, it played a pivotal role in popularizing deep neural networks by winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) that year [34].

The architecture consists of eight layers: five convolutional layers that extract spatial features from input images, followed by three fully connected layers responsible for classification. It employs Rectified Linear Unit (ReLU) activations to accelerate training and incorporates dropout regularization to reduce overfitting. AlexNet's success marked a turning point in the practical application DL to real-world image analysis tasks.

2.2.3.2. GoogLeNet

GoogLeNet, also referred to as Inception v1, was developed by Szegedy et al. at Google Research and introduced in 2014. It gained prominence by winning the ILSVRC competition that year [29]. The architecture is composed of 22 layers and is particularly notable for its introduction of inception modules. These modules allow the network to perform convolutions at multiple spatial scales in parallel, enabling it to capture both fine-grained and high-level features efficiently. Additionally, GoogLeNet replaces fully connected layers with global average pooling, significantly reducing the number of parameters and thereby improving computational efficiency without compromising accuracy.

2.2.3.3. SqueezeNet

SqueezeNet was proposed in 2016 by Iandola et al., with the goal of achieving AlexNet-level classification accuracy using a model with drastically fewer parameters [48]. The key innovation in SqueezeNet lies in its use of "fire modules," which consist of a squeeze layer (1×1 convolutions) followed by an expand layer (a combination of 1×1 and 3×3 convolutions). This compact design allows the model to deliver strong performance with a reduced memory footprint, making it well-suited for applications on embedded systems or mobile devices with constrained resources.

2.2.3.4. ResNet

short for Residual Network, was introduced by He et al. (2016) to address the vanishing gradient problem in deep neural networks. It employs skip (residual) connections that allow gradients to flow directly through identity mappings, enabling the successful training of extremely deep models. ResNet architectures vary in depth (e.g., ResNet18, ResNet50,

ResNet101), and are widely used for image classification and feature extraction due to their high accuracy and stable convergence [49].

2.2.3.5. MobileNet

MobileNet, proposed by Howard et al. (2017), is a family of lightweight deep convolutional neural networks optimized for mobile and embedded vision applications. It utilizes depthwise separable convolutions, which decompose standard convolutions into depthwise and pointwise operations, dramatically reducing the number of parameters and computational cost while maintaining competitive accuracy. Variants such as MobileNetV2 and MobileNetV3 further improve efficiency using inverted residuals and squeeze-and-excitation modules [50].

2.2.3.6. EfficientNet

EfficientNet represents a modern family of CNN architectures introduced in 2019, aimed at enhancing accuracy while reducing computational complexity. Its core innovation is the compound scaling method, which uniformly adjusts the network's depth, width, and input resolution using a single scaling coefficient. Unlike traditional approaches that scale one or two dimensions independently, EfficientNet's compound strategy ensures a more balanced and effective use of model capacity. This enables the architecture to deliver state-of-the-art performance with significantly fewer parameters and lower computational costs than earlier CNN models. EfficientNetB0 serves as the baseline model in this family and exemplifies the effectiveness of the compound scaling approach [51].

2.2.4. Training parameters

The efficiency and effectiveness of training DL models depend on the careful selection and configuration of key parameters: optimizers, learning rate, batch size, and loss function. These parameters play critical roles in guiding the model towards optimal performance.

2.2.4.1. The Optimizer

Optimizers update the model's weights to minimize the loss function. Three commonly used optimizers are:

- **RMSprop** adjusts the learning rate for each parameter, helping to stabilize the training process, especially when gradients fluctuate [52].
- **SGD**: updates weights based on the gradient of the loss function but can be slow to converge, particularly with complex datasets [53]. Momentum variants of SGD help accelerate convergence in consistent directions.
- **Adam**: combines RMSprop and momentum, adapting learning rates for each parameter and leading to faster convergence and better stability, making it ideal for noisy or sparse gradients [54].

2.2.4.2. The Learning Rate

The learning rate controls how large the adjustments are during the optimization process. When properly tuned, it balances quick convergence with stable training. If the learning rate is too high, it can make the training unstable, while a learning rate that is too low results in very slow convergence.[55].

2.2.4.3. The Batch Size

Batch size is the number of samples the model processes before updating its weights. Larger batch sizes tend to produce more stable gradient estimates but require greater memory resources. In contrast, smaller batch sizes add more variability to the gradients, which can help improve generalization and make each training iteration faster [56].

2.2.4.4. The Loss Function

The loss function measures how different the model’s predictions are from the actual target values. It is essential for guiding the optimization of the model’s parameters. Common examples include Mean Squared Error (MSE) for regression problems and Cross-Entropy Loss for classification tasks. [57].

Together, these parameters guide the optimization process and significantly influence the model's final performance.

2.2.5. Improvement techniques

2.2.5.1. Dropout

Dropout is a regularization technique applied in neural networks to minimize overfitting. During training, it randomly deactivates a certain percentage of neurons by setting their outputs to zero. By temporarily removing these neurons, the network is prevented from depending too heavily on particular connections. As a result, the model learns more distributed and robust feature representations, which improves its ability to generalize to unseen data.[58].

2.2.5.2. L2 regularization

also known as weight decay, helps prevent overfitting by adding a penalty term proportional to the square of the magnitude of the weights to the loss function. This encourages smaller weight values and promotes smoother model behavior, improving generalization on unseen data [59].

2.2.5.3. Squeeze-and-excitation (se) blocks

Squeeze-and-Excitation (SE) blocks are architectural components that improve the representational capacity of a convolutional neural network (CNN) by explicitly capturing and modeling the relationships between different feature channels. They perform two main operations: "squeeze," which captures global spatial information using global average pooling, and "excitation," which adaptively recalibrates channel-wise feature responses through learned weights [60].

2.2.5.4. Global average pooling (GAP)

Global Average Pooling (GAP) is a downsampling operation that replaces traditional fully connected layers by averaging each feature map across its spatial dimensions. This reduces the number of parameters, mitigates overfitting, and preserves spatial correspondence between feature maps and output categories [31].

2.3. Related works

Rapid fire detection and effective risk monitoring play a vital role in minimizing response times, reducing potential damage, and lowering associated costs. Numerous fire-fighting strategies have been proposed in previous studies [61],[62]. This section provides an overview of recent state-of-the-art approaches related to the technique proposed in this work.

Traditional fire detection methods have relied on human observers to visually monitor and identify signs of fire outbreaks. However, this approach is limited by factors such as operator fatigue, the availability of personnel over extended periods, and the difficulty of maintaining consistent coverage across vast or remote areas [15]. As technology has advanced, more sophisticated mechanical solutions have emerged to aid fire detection. A notable example is the deployment of wireless sensor networks, which detect smoke by analyzing airborne particles and monitoring changes in temperature [16]. However, these systems have limitations, such as their limited spatial range and reliance on battery power, which can be quickly depleted, reducing their long-term effectiveness.Another approach involves satellite-based surveillance systems, which offer the ability to observe large land areas from orbit. While satellite systems have the advantage of wide-scale fire monitoring, they face significant drawbacks, including low spatial resolution, which can delay the detection of smaller fires. Additionally, the high financial costs of satellite imaging and data processing, along with challenges posed by environmental conditions such as cloud cover and smoke, can compromise their effectiveness [17].

In recent years, the use of image-based fire detection techniques has grown significantly, largely due to rapid advancements in digital camera technology and the development of more sophisticated image processing methods. These improvements have enabled faster and more accurate analysis of visual data, making camera-based systems a promising approach for early fire detection and monitoring. These systems offer advantages such as quick detection, greater accuracy, and flexibility in installation across various environments. Despite these benefits, challenges remain, including the impact of fluctuating environmental factors (e.g., weather, lighting) on detection accuracy. Additionally,

distinguishing between real fires and other phenomena resembling fires, such as industrial smoke or sunlight glare, remains a significant challenge [63].Numerous studies have investigated satellite imagery for forest fire detection. Guangmeng and Mei [64] utilized data obtained from the Moderate Resolution Imaging Spectroradiometer (MODIS) sensors mounted on NASA's Terra and Aqua satellites. Although MODIS is widely employed in environmental monitoring applications, its relatively low spatial resolution and sensitivity to cloud cover limit its effectiveness in detecting small-scale fires. Similarly, Li et al. [65] used imagery from the Advanced Very High Resolution Radiometer (AVHRR) for fire detection. While AVHRR benefits from infrared and near-infrared bands that enhance fire identification, it often struggles to differentiate actual fires from other high-temperature sources such as industrial facilities. Moreover, satellite-based systems generally suffer from low temporal resolution, as images are typically updated once every one or two days, making them unsuitable for real-time fire detection [66].

To overcome these limitations, researchers have increasingly explored vision-based approaches using ground cameras and digital imaging systems. Recent advances in computer vision have significantly improved image processing techniques for forest fire detection [67][68][69]. These approaches analyze visual characteristics within images to detect fire events rapidly, providing real-time capabilities and cost-effective alternatives to satellite monitoring. Several studies have focused specifically on flame detection by classifying fire-related pixels using RGB color space models [70][71][72][73]. One study [15] proposed combining YCbCr and RGB color spaces to identify fire-prone regions based on color characteristics. In addition, temporal and spatial wavelet analysis techniques have been employed to improve detection accuracy by extracting multi-level local features from image sequences [74],[75]. Despite their potential, these methods can be affected by environmental variations such as lighting changes and image quality, and deploying such models typically requires high computational power and large datasets.Other studies have explored fire detection using video sequences [76]. Unlike still-image analysis, video-based methods must consider temporal dynamics and motion across frames, which increases algorithmic complexity. While they provide richer information for detecting fire behavior

over time, these methods can be more computationally intensive and may require advanced techniques to reduce false positives and ensure reliable detection.

The application of ML in forest fire detection has garnered increasing attention in recent years due to the critical need for timely and accurate identification of fire events. A notable contribution in this domain is the study by Foggia et al. [77], who developed a fire detection system specifically designed for real-time video surveillance. Their method integrates an ensemble of expert modules, each focused on analyzing distinct visual features typically associated with fire, such as flame color, geometric characteristics of fire regions, and dynamic motion patterns indicative of fire behavior. By combining these heterogeneous cues, the system enhances its ability to detect fires with improved robustness and accuracy, particularly in structured environments where visual signatures are clearly distinguishable. This multi-feature strategy mitigates the limitations of relying on a single visual attribute, allowing for better generalization across varied video scenes. Despite its strengths, the system is not without limitations. A primary drawback lies in its computational complexity, as the simultaneous operation of multiple feature extraction algorithms requires substantial processing power. This makes the approach less feasible for deployment in resource-constrained environments, such as remote forest locations or on edge devices, where real-time performance is essential. Additionally, the success of this ensemble approach hinges on the precise calibration and synchronization of its individual modules. Any inconsistencies in input data or environmental fluctuations, such as varying lighting conditions or camera motion, can disrupt synchronization and compromise detection accuracy. This can result in increased false positives, missed detections, or inconsistent outputs, thereby reducing the system's overall reliability and suitability for real-world wildfire monitoring applications. In another study, researchers in [78], proposed a forest fire management system utilizing unmanned aerial vehicles (UAVs) equipped with artificial intelligence (AI) and advanced sensors for real-time fire detection and response. Their system incorporates computer vision algorithms and applies a range of traditional ML classifiers, including k-nearest neighbors (KNN), naïve Bayes (NB), random forest (RF), support vector machine (SVM), and logistic regression (LR), to classify and identify fire incidents captured by UAV-mounted cameras. Further expanding on this line of research, Spoorthy

et al.[79] examined the effectiveness of several ML algorithms for predicting forest fire risk in specific geographic regions. Their methodology involved training classifiers on labeled datasets from historically fire-prone areas to uncover patterns associated with fire outbreaks. The study compared four widely used models: SVM, Decision Trees (DT), KNN, and Random Forest. The results demonstrated that SVM achieved the highest predictive accuracy at approximately 92%, followed by Random Forest (85%), Decision Trees (78%), and KNN (72%). The superior performance of SVM was attributed to its capability to separate data points effectively using hyperplanes, making it well-suited for complex classification tasks. The use of localized data further enabled the models to learn region-specific patterns, enhancing their predictive capabilities. However, while the results were promising, the study presented several limitations. The focus was primarily on forecasting fire-prone zones based on historical data, rather than detecting active fires in real time. Consequently, although the models were effective in risk prediction, they lacked the operational capability for immediate fire detection. Moreover, the ML methods explored in this study are relatively basic compared to modern DL and advanced computer vision approaches, which offer greater scalability, adaptability, and resilience in dynamic environments. In large-scale deployment scenarios, where environmental variability and the rapid evolution of fire conditions pose substantial challenges, classical ML models may struggle to maintain high accuracy and responsiveness. As such, more sophisticated methods are required to meet the complex demands of contemporary wildfire detection and management systems.

In recent years, CNNs have gained significant attention in forest fire detection research due to their capacity to automatically learn complex and high-level features from raw image data. This eliminates the need for manual feature extraction and enhances detection accuracy and reliability. CNNs have achieved remarkable results in various domains, including object recognition [19], autonomous driving [20], and medical diagnostics [21], which has encouraged their application to fire detection tasks. Their adaptive learning process enables CNNs to identify visual fire characteristics, such as flame color, shape, and texture,even under challenging conditions like occlusion or variable lighting. This adaptability makes them promising tools for advancing automated forest fire detection systems.

Several researchers have explored DL approaches for early wildfire detection. Muhammad et al. [80] introduced a framework based on fine-tuned CNNs to improve detection accuracy. Likewise, other studies [81][82][83] employed DL across diverse landscapes and reported promising outcomes. However, these approaches face notable challenges. A primary issue is the scarcity of balanced and representative datasets focused specifically on forest fire imagery. Many datasets contain an imbalance between fire and non-fire images or feature non-forest fire scenarios, such as urban fires. This can result in biased learning and poor generalization in real-world forest conditions. Furthermore, DL models typically require high computational resources and extended training time, limiting their feasibility in real-time and low-resource environments like remote forest areas. The limited variability and context in training data, often skewed toward video-based samples and hinder the robustness of these models.To address such challenges, Ahmad et al. [1] proposed FireXNet, a lightweight DL model optimized for wildfire detection. It integrates the SHAP (SHapley Additive exPlanations) method to improve model transparency and interpretability. Tested on a dedicated wildfire dataset, FireXNet achieved an accuracy of 98.42%, outperforming several well-known models including VGG16, InceptionResNetV2, InceptionV3, DenseNet201, and MobileNetV2. This result highlights both the efficiency and effectiveness of the proposed model. In support of DL research in this domain, Shamsoshoara et al. [84] developed the FLAME dataset for wildfire image segmentation and classification. Using a modified Xception model with fully connected layers and optimized hyperparameters, they achieved a classification accuracy of 76%, establishing a valuable baseline for future studies in aerial wildfire detection. Namburu et al. [85] introduced X-MobileNet, an improved version of MobileNetV2 adapted through TL for forest fire detection. They preserved pre-trained ImageNet weights to serve as a general feature extractor and incorporated a Global Average Pooling (GAP) layer to enhance classification efficiency while reducing overfitting.This architecture achieved strong generalization performance, which makes it well suited for real-time applications, especially in environments with limited computational resources. Similarly, Treneska et al. [86] evaluated the performance of five pre-trained CNN architectures including VGG16, VGG19, ResNet50, InceptionV3, and Xception ,on the FLAME dataset using a TL approach. After replacing the final layers with a custom

GAP-based classifier, ResNet50 demonstrated the highest accuracy (88%) following fine-tuning, confirming its potential for wildfire detection tasks.

In recent years, TL has emerged as a powerful method for addressing various computer vision challenges, including wildfire detection [22][23][87]. This technique involves utilizing pre-trained DL models, such as VGG19, InceptionV3, and ResNet50, that have been trained on large-scale datasets like ImageNet. These models have learned to extract generalized visual features, which can then be fine-tuned for specific tasks. By using pre-trained models, researchers can reduce the need for large labeled datasets and minimize computational costs associated with training deep models from scratch. This has led to faster development and more efficient deployment of robust models for wildfire detection and other vision-based applications. In addition, within the same study, Ali Khan et al. [78] introduced the DeepFire dataset along with a deep TL benchmark to improve forest fire detection systems. They demonstrated the effectiveness of TL in enhancing detection accuracy, especially in situations where large annotated datasets are scarce. By employing the pre-trained VGG19 model, the researchers leveraged features learned from large datasets like ImageNet to classify forest fire images. They improved the model's classification performance by adding fully connected layers and achieved a classification accuracy of 95% on the DeepFire dataset, proving the value of adapting pre-trained models for wildfire detection. Yandouzi et al. [88] developed a wildfire detection system using 11 prominent pre-trained CNN architectures, including VGG16/19, InceptionV3, ResNet50/V2, InceptionResNetV2, Xception, DenseNet, MobileNet/V2, and NASNetMobile. The Forest Fire Images dataset, consisting of 4,661 images, was employed, and data augmentation techniques were used to improve model robustness. By replacing the original classifier layers with custom layers for binary classification, the method achieved an impressive detection accuracy of 99.9%. However, this approach was limited to static image classification and did not explore real-time or video-based applications. Islam et al. [89] proposed a novel DL framework for wildfire classification, combining EfficientNetB7, a powerful pre-trained CNN, with a custom Attention Connected Network (ACNet). The hybrid architecture processes both aerial and ground-level imagery, capturing comprehensive fire-related features. The attention mechanism enables the model to focus on key areas of each image

stream, enhancing its ability to identify wildfire patterns in complex scenes. The integration of EfficientNetB7 ensures strong feature extraction, while ACNet refines attention-driven processing. Additionally, Bayesian optimization was used for fine-tuning hyperparameters, achieving state-of-the-art results with classification accuracies of 97.45% on the FLAME dataset and 95.97% on the DeepFire dataset. Wang et al. [90]. proposed the Reduce-VGGNet model for wildfire image classification, building upon the widely used VGG-16 network architecture. Their approach involves transferring the weight parameters from the first 13 convolutional layers of VGG-16 and replacing the original fully connected layers with two simpler layers (1024 and 2 neurons) and a Softmax classifier. This modification reduces the number of parameters, resulting in decreased training time while still benefiting from the pre-trained features of VGG-16 through TL. The model demonstrates improved efficiency, particularly for datasets with fewer images, like the DeepFire dataset. However, the reduced complexity of the fully connected layers may limit the model's capacity to capture more complex patterns, potentially leading to underfitting, especially if the dataset is not diverse or large enough. Furthermore, by fine-tuning only the fully connected layers, the model may not fully adapt to the unique features of wildfire images. While this approach offers a computationally efficient solution for wildfire classification, its scalability to more complex datasets or tasks remains uncertain, and further exploration into fine-tuning additional layers or experimenting with more complex architectures could enhance its performance and generalization capabilities. Khan and Khan [91] proposed FFireNet, a DL method for forest fire classification using TL with MobileNetV2 as the backbone. They froze the convolutional layers and added new fully connected layers to classify fire and no-fire images, achieving high accuracy (98.42%) and recall (99.47%). However, their approach is limited by the use of frozen features, which may restrict adaptation to fire-specific patterns, and it focuses only on image classification without addressing fire localization or real-time detection challenges. Prakash et al. [92]. present an AI-based forest fire detection system that leverages vision-based techniques for the early identification and classification of fire events. Their approach employs TL using the Inception-v3 convolutional neural network to distinguish between fire and non-fire images from the DeepFire dataset. To enhance detection accuracy, the system integrates a hybrid framework combining Radial Basis Function Networks (RBFNs) with Rapid and Accurate Image

Super Resolution (RAISR), which improves image quality and feature representation. This combination significantly boosts the classification performance, with the RBFN-RAISR model achieving a reported accuracy of 97.55%. Ghali et al. [93] introduced a hybrid DL model combining EfficientNet-B5 and DenseNet-201 to perform wildfire classification using aerial imagery. The fusion of EfficientNet's compound scaling and DenseNet's feature reuse capabilities improved the model's ability to extract features at multiple scales, achieving a classification accuracy of 85.12%. This approach outperformed previous methods, particularly in detecting small and less prominent fire regions.

2.4. Conclusion

This chapter has reviewed the fundamental principles of Deep Learning, including CNN architectures, Transfer Learning strategies, and key optimization techniques, alongside a critical analysis of recent advances in forest fire detection systems.While these advancements demonstrate the potential of combining pre-trained CNNs, custom attention mechanisms, and optimization strategies, several challenges remain. The performance of some models is highly dependent on the careful selection of optimization parameters and the quality and diversity of the training data. Moreover, limited dataset size, lack of cross-dataset validation, and hardware constraints often restrict the generalizability and real-time applicability of the proposed solutions. These limitations highlight the need for more robust and computationally efficient approaches capable of maintaining high detection accuracy while ensuring scalability and practical deployment in real-world wildfire monitoring systems.

In light of these challenges, the present study proposes a comprehensive framework based on TL and lightweight model design to enhance detection performance while addressing generalization and efficiency concerns. The following chapter details the proposed methodology and experimental validation.

3. Forest fire detection based on transfer learning

3.1. Introduction

The approach used to detect forest fires using TL is presented in this chapter. For the binary classification of fire and no-fire images, it begins by introducing the suggested TL approach, which entails optimizing a number of cutting-edge CNNs, such as AlexNet, GoogLeNet, ResNet18, SqueezeNet, Efficient-NetB0, and MobileNetV2. To find the best configuration for each network, the models were modified by altering their final layers and assessed under various hyper-parameter configurations. The chapter then describes the experimental framework, including the two datasets used (Deep-Fire and Forest Fire Images), the specific TL procedures applied to each pre-trained model, the evaluation metrics adopted to assess performance, and the hardware and software environment used for implementation. The chapter concludes by presenting and discussing the experimental findings and offering a comparison of the assessed models in terms of computational efficiency, accuracy, and generalizability. Through a fair and repeatable evaluation, the goal is to determine the most efficient and dependable configuration for forest fire detection.

3.2. Proposed transfer learning technique

TL is an efficient approach for modifying deep learning models, previously trained on large datasets, to accommodate new tasks with little labeled data. Instead, then training a deep CNN from the ground up, transfer learning utilizes pre-existing visual representations, including edges, textures, forms, and object components. These representations exhibit significant transferability across many image recognition challenges. Consequently, TL significantly reduces training duration, mitigates overfitting, and enhances generalization performance while utilizing relatively limited datasets.
This study utilizes transfer learning (TL) to adapt six prominent CNN architectures, including AlexNet, GoogLeNet, ResNet18, SqueezeNet, EfficientNetB0, and MobileNetV2, for forest fire detection. These designs were chosen because they have different levels of depth, structural design, and computational complexity. This lets us fully evaluate the trade-off between classification performance and computational efficiency.To guarantee robustness and generality across various data distributions, the proposed technique was assessed using two publically accessible datasets: the DeepFire

dataset and the Forest Fire Images dataset. Utilizing datasets with diverse characteristics aids in validating the robustness of the suggested methodology and mitigates potential dataset-specific bias. The comprehensive workflow of the suggested technique is depicted in Fig. 3-1 and comprises multiple sequential stages, which are elaborated upon in detail below.

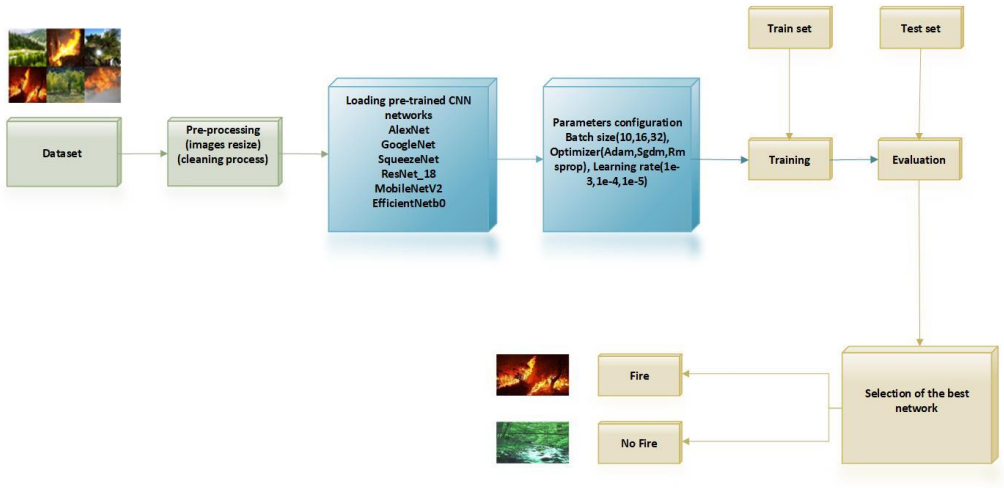


Fig. 3-1. Processing Pipeline of the Proposed Technique.

3.2.1. Image preprocessing

The first stage focuses on preparing the raw images to satisfy the input specifications of the selected CNN architectures. This step is essential to ensure consistent data formatting and reliable training. All images are resized to fixed spatial dimensions according to the input layer specifications of each model. Specifically, images are resized to 224 × 224 pixels for GoogLeNet, ResNet18, SqueezeNet, EfficientNetB0, and MobileNetV2, while a resolution of 227 × 227 pixels is used for AlexNet. This resizing process ensures compatibility with the pretrained networks while preserving essential visual information. In addition, the Forest Fire Images dataset was inspected to identify corrupted or invalid files. Images with zero byte size that cannot be loaded or processed were removed in order to avoid training interruptions and potential bias. Further details regarding this data cleaning procedure are provided in Subsection 3.3.2.2.

3.2.2. Transfer learning and model adaptation

All six CNN architectures were originally trained on the ImageNet dataset, which contains more than one million natural images distributed across one thousand categories. Pretraining on ImageNet enables these networks to learn rich hierarchical feature representations that are highly effective for a wide range of visual recognition tasks. In the proposed approach, the pretrained convolutional and feature extraction layers of each model are retained in order to benefit from their learned representations. The original final classification layers are removed and replaced with custom fully connected layers designed specifically for the binary classification task of forest fire detection. The new output layer consists of two neurons corresponding to the fire and no fire classes, followed by either a softmax or sigmoid activation function depending on the network architecture. This adaptation strategy allows the models to specialize in forest fire detection while maintaining the advantages of pretrained knowledge.

3.2.3. Hyperparameter optimization strategy

To optimize model performance and analyze the influence of training parameters, a systematic exploration of key hyperparameters was conducted. Three main hyperparameters were considered. Batch sizes of 10, 16, and 32 were evaluated in order to balance computational efficiency and training stability. Smaller batch sizes may enhance the model's generalization ability, though they require more time to train, whereas larger batch sizes reduce computational overhead but may lead to less stable convergence. The selected values are appropriate for limited dataset sizes and moderate computational resources. Three optimizers were examined, including Adam, stochastic gradient descent with momentum, and RMSprop. These optimizers were chosen due to their widespread use and proven effectiveness in deep CNN training. Adam combines adaptive learning rates with momentum, stochastic gradient descent with momentum provides stable convergence and strong generalization, and RMSprop updates learning rates adaptively using information from recent gradients. Learning rates of 0.001, 0.0001, and 0.00001 were tested, as the learning rate directly affects convergence speed and training stability. These values are frequently used in fine-tuning pre-trained models and offer a convenient range for assessing performance

behavior. The combination of these hyperparameters produced a total of 27 configurations for each model, reflecting three choices of batch size, optimizer, and learning rate.

3.2.4. Training and evaluation protocol

For each dataset, the predefined training and testing subsets provided by the dataset creators were used. The training sets were employed to fine-tune the models and update network weights, while the test sets were used to evaluate the generalization performance of the models on unseen data. During training, each configuration was monitored using validation performance in order to assess convergence behavior and classification effectiveness. In the evaluation phase, the trained models classified input images into either fire or no fire categories. Performance was assessed using standard classification metrics derived from the confusion matrix, including accuracy and related measures, as described in the evaluation section (section 3.3.3).

3.2.5. Model selection and comparative analysis

Based on the evaluation results, the best performing configuration was identified for each CNN architecture. These optimal configurations were then compared across all models to analyze both classification performance and computational efficiency. This final comparison enabled the selection of the most suitable architecture for real world deployment in forest fire detection systems, where accuracy, robustness, and resource constraints are critical considerations.

3.3. Experimental setup

This subsection outlines the experimental setup used to evaluate the performance of pretrained convolutional neural network models for forest fire detection.

3.3.1. The transfer learning setup

This section describes the TL strategy adopted in this study, including the fine-tuning configuration used to adapt pretrained convolutional neural networks to the forest fire detection task. The pretrained models were initialized with weights learned from large-

scale image datasets and then fine-tuned on the target datasets to improve task-specific performance.

In our study, we focused on three core hyperparameters: batch size**,** learning rate, and optimizer, due to their well-documented influence on model convergence, training stability, and overall efficiency. These hyperparameters are common to virtually all DL pipelines and are typically the first to be tuned when adapting pretrained models to new tasks. The selected values were guided by empirical best practices in CNN-based image classification and fine-tuning:

• The batch sizes we have chosen to implement are 10, 16, and 32, reflecting small to moderate regimes, which can be associated with limited datasets and computation constraints.

• The learning rates of 0.00001, 0.0001, and 0.001 have been commonly recommended for fine-tuning a pre-trained model, reflecting a useful range to observe the trend of the performance.

• We implemented Adam, SGDM, and RMSprop as optimizers, reflecting both their widespread use and their varied convergence characteristics.

3.3.2. Dataset description

In order to assess the performance of the proposed TL under different conditions and image sources, two diverse datasets were utilized. Each dataset offers unique characteristics in terms of fire scenarios and image quality, ensuring a robust evaluation of the models' generalizability.

3.3.2.1. DeepFire dataset

This work employs a publicly accessible benchmark dataset particularly created for forest fire detection tasks [78]. There are 1,900 annotated photographs in total: 950 of them are of fire and 950 of them are not fire. The information was gathered from a variety of natural settings, including plains, mountains, and bodies of water. It talks about several kinds of forest fires, like crown fires and surface fires. It also covers visually complicated

situations where the environment is affected, including when the sun rises and sets, when the light changes, and when there are dried yellow or brown leaves that look like fire or smoke. These traits make the dataset especially good for testing how well a model works in the actual world. To make sure the evaluation is fair, the dataset is separated into training and test sets ahead of time. For training, there are 1,520 photographs (760 from each class), and for testing, there are 380 images (190 from each class). This dataset is split 50% into the fire class and 50% into the non-fire class. This makes it perfect for creating models that can learn general features to tell the difference between fire and non-fire situations. Some sample photos from both groups can be shown in Fig. 3-2.

(a) (b)

Fig. 3-2 DeepFire dataset samples: (a) No-fire images, (b) fire images.

3.3.2.2. Forest Fire Images dataset

The Forest Fire Images dataset, which was retrieved from the Kaggle platform [94], originally comprised 5,050 images, with an equal number of fire and no-fire images. However, following the removal of corrupted images (0-byte images) during the data cleaning process, the dataset comprised 4,661 images, including 2,136 fire and 2,525 no-fire images. Of these, 4,611 images were allocated for the training process, including 2,111 fire and 2,500 no-fire images. The remaining 50 images, including 25 fire and 25 no-fire images, were allocated for testing. The dataset comprises wildfire images obtained from

different global locations. The wildfire images include different scenarios of fire, such as surface fire and crown fire, in different environments, including mountains, valleys, and water. This is beneficial for model training, as the model can recognize fire in different conditions. The samples of fire and no-fire images are shown in Fig. 3-3. The fire images account for approximately 46% of the total dataset, while the no-fire images account for approximately 54%. The fire images are fewer than the no-fire images. However, the dataset is sufficiently balanced for the evaluation of the model performance under real-world class imbalance conditions.

(a) (b)

Fig. 3-3.Forest Fire Images dataset samples: (a) No-fire images, (b) fire images.

3.3.3. Evaluation metrics

The reliable evaluation of model performance during training and testing is crucial for identifying the most effective configuration. In this research, a set of quantitative evaluation metrics has been adopted to measure the model's capability in correctly distinguishing between images containing fire and those without. These metrics are accuracy, Precision, Recall, and F1 score.

These performance indicators are derived from four fundamental classification outcomes that result from comparing predicted labels with ground truth labels [95]. These include:

- True Positives (TP): Fire images correctly identified as fire.
- True Negatives (TN): Non-fire images correctly identified as non-fire.
- False Positives (FP): Non-fire images misclassified as fire.
- False Negatives (FN): Fire images misclassified as non-fire.

A Confusion Matrix (CM) is used to make it easier to understand these results. This matrix gives a clear picture of how well the model did in predicting the two target classes, which helps us understand the strengths and shortcomings of the classification findings [96] . The evaluation metrics show how well the model is doing by looking at the distribution of TP, TN, FP, and FN in the matrix. This is especially important in situations like detecting forest fires, where both missed detections (FN) and false alarms (FP) can have serious effects.

3.3.3.1. Accuracy

Accuracy evaluates the classifier's overall performance by determining the ratio of correctly categorized cases (both fire and non-fire) to the total number of predictions made. This metric is frequently employed in classification tasks and is computed using Equation (3):

$$\text{Accuracy} = \frac{\text{TP+TN}}{TP+TN+FP+FN} \quad (3)$$

3.3.3.2. Precision

Precision measures how accurate positive predictions are by finding the percentage of true positives among all cases that were anticipated to be positive (i.e., fire). Negative accuracy also means the number of accurately anticipated no-fire cases divided by the number of all expected negatives .Equation (4) shows how to define precision [97]:

$$\text{Precision} = \frac{TP}{TP+FP} \quad (4)$$

3.3.3.3. Recall

Recall, also referred to as sensitivity, evaluates the model's efficacy in identifying positive samples. It quantifies the proportion of accurately identified fire photos relative to the total occurrences of fire cases. The calculation is performed using Equation (5) [97]:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (5)$$

3.3.3.4. F1-Score

The F1-score offers a comprehensive assessment by combining precision and recall into one measure. This is especially advantageous for managing unbalanced datasets or where both false positives and false negatives are crucial, such as in fire detection.The F1-score is determined using the harmonic mean, as seen in Equation (6) [97]:

$$\text{F1-score} = 2 \times \frac{Precision \times Recall}{Prescision + Recall} \quad (6)$$

3.3.3.5. The confusion matrix

Serves as a visual instrument that organizes prediction outputs, offering a comprehensive overview of the model's efficacy by categorizing results into true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) [96].

3.3.3.6. Receiver operating characteristic (ROC) curve

The ROC curve is a conventional assessment instrument for binary classification tasks. It evaluates the diagnostic efficacy of a classifier by illustrating the correlation between the True Positive Rate (TPR) and the False Positive Rate (FPR) across various threshold values [98]. This graphic facilitates the assessment of model performance across many thresholds, offering a more thorough perspective than depending on a singular categorization limit.

3.3.3.7. Area under the curve (AUC)

AUC measures the efficacy of a model in distinguishing between positive and negative classes. The metric spans from 0 to 1, with 1 signifying flawless classification, 0.5 representing random predictions, and values beneath 0.5 indicating inadequate performance [99]. Elevated AUC scores indicate superior classification capability and improved differentiation between fire and non-fire instances, which is essential in fire detection operations.

The AUC-ROC metric is very useful for datasets that aren't balanced because it checks how well the model works without taking into account the decision threshold or class distribution [100]. Its strong features make it a useful addition to other performance metrics like accuracy, precision, recall, and F1-score.

3.3.3.8. Number of parameters

In deep learning, the phrase "number of parameters" denotes the aggregate of learnable variables within a neural network, mostly encompassing weights and bias terms that are modified throughout the training process [101]. The overall parameter count influences the memory requirements and the duration of the training process. Neural networks with many parameters typically take longer to train compared to models with fewer parameters [102]. Consequently, neural networks with fewer parameters offer greater training efficiency and are preferable for applications where computational cost is an important factor.

3.3.3.9. Floating point operations

Floating Point Operations (FLOPs), as a measure, defines the total number of arithmetic operations, including additions, multiplications, and divisions, that a computational process performs while executing the program with floating-point data types. Within the domains of deep learning and neural networks, FLOPs are acknowledged as a platform-agnostic metric that assesses the computing demands of a model. The FLOPs

measure allows for estimating the theoretical costs that might be involved in the process of training and inference, It has also been acknowledged as an effective instrument for evaluating the viability of a model with constrained computational resources [103].

3.3.4. Hardware and software specifications

All trials were conducted on a Lenovo ThinkPad laptop, running MATLAB R2022b, along with the following hardware specifications: Intel Core i7-6600U @ 2.6 GHz processor, 20 GB RAM, and Windows 10 Pro OS.

3.4. Results and discussion

This section presents and discusses the experimental results obtained from the evaluation of six pre-trained CNN models that are widely used: AlexNet, ResNet-18, SqueezeNet, GoogLeNet, MobileNetV2, and EfficientNet-B0. The models were tested on both datasets. In the experiments, the impact of three significant hyperparameters, namely batch size, learning rate, and optimizer, was investigated to determine the best combination of hyperparameters, which can produce the best results. Once the best architecture of the CNN model and hyperparameters are determined, the outcomes are contrasted with those obtained from other advanced deep learning approaches.

3.4.1. The Influence of CNN hyper parameters on DeepFire dataset

This sub-section focuses on analyzing how the architectural choices of six convolutional neural network models affect their performance in detecting forest fires within the DeepFire dataset. We performed a comprehensive set of trials to examine the effects of various optimizers, batch sizes, and learning rates, in order to find the setups that lead to the highest accuracy in forest fire detection. For each of the CNN models, the evaluation of each architecture focused on the optimal set of parameters from a set of three optimizers, namely Adam, RMSprop, and stochastic gradient descent with momentum, combined with learning rates of 0.001, 0.0001, and 0.00001, and a range of batch sizes of 10, 16, and 32. Figures 3.4 through 3.9 present the experimental results, illustrating how different batch sizes and learning rates affect the test accuracy of each CNN model.

3.4.1.1. AlexNet architecture

As shown in Fig. 3-4, when the Adam optimizer is used, a maximum accuracy of 99.21% is achieved when the batch size is 10 and the learning rate is 0.00001. However, when the learning rate is increased to 0.001, a sharp fall in accuracy is seen, and it reduces to 50%. This fall in accuracy is because, with a high learning rate, the steps taken in the direction of the gradient vector are large, which might cause the solution to diverge. As a result, the performance and accuracy of the model get affected. This shows that the Adam optimizer is highly responsive to the learning rate. From the above figures, it is also seen that the batch sizes exert minimal influence on the model's performance, as the trends in

the accuracy for different batch sizes are almost the same. When the RMSprop optimizer is used, a maximum accuracy of 99.74% is achieved when the batch size is 16 and the learning rate is 0.00001, which is the best performance compared to other optimizers. However, when the learning rate is increased to 0.001, the accuracy reduces sharply and reaches 87.63%. As seen above, the batch sizes have a negligible effect on the performance of the model, as the trends in the accuracy for different batch sizes remain almost the same. The SGDM optimizer has a maximum accuracy of 98.68% when the batch sizes and learning rates are 16 and 0.0001, or 32 and 0.00001, respectively. However, when the learning rate is increased to 0.001, the accuracy falls sharply and reaches 50%, which again shows that high learning rates have a negative impact on the performance and accuracy of the model, as seen with other optimizers. The best configuration for the AlexNet model is when the RMSprop optimizer, a batch size of 16, and a learning rate of 0.00001 are used. The other optimizers have varying levels of sensitivity with respect to hyperparameter tuning.

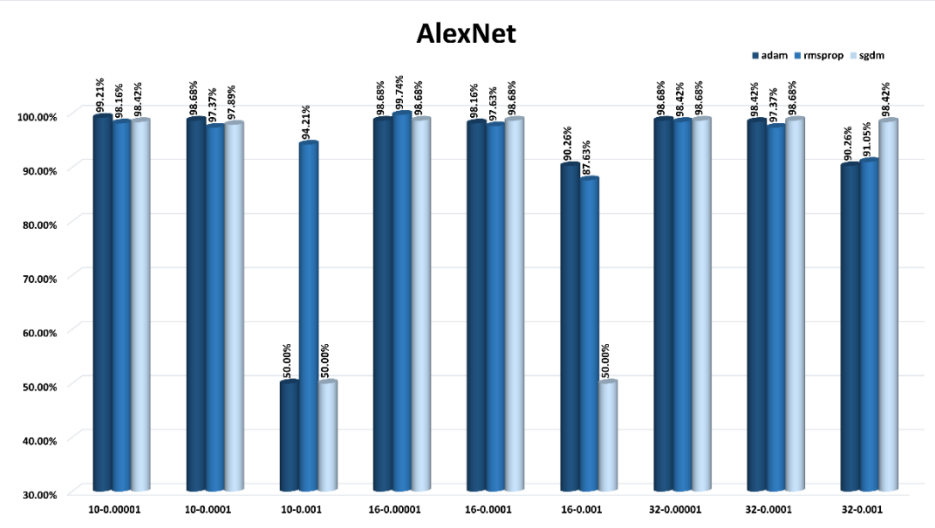


Fig. 3-4. Accuracy of AlexNet with Adam, SGDM, and RMSprop optimizers for multiple batch sizes and learning rates.

3.4.1.2. ResNet18 architecture

As illustrated in Fig. 3-5, the best accuracy of 99.47% is obtained by employing the Adam optimizer with a batch size of 10 and learning rates of 0.00001 or 0.0001. However, if the learning rate is increased to 0.001, the accuracy is reduced to 95.79%. It can also be seen that the best accuracy of 99.21% is obtained by employing the Adam optimizer with a batch size of 16 and a learning rate of 0.0001. Furthermore, the best accuracy of 98.68% is obtained by employing the Adam optimizer with a batch size of 32 and the same learning rate of 0.0001. The RMSprop optimizer has also shown good results, with the best accuracy of 99.47% being obtained by employing the RMSprop optimizer with a batch size of 16 or 32 and the same learning rate of 0.00001. The best accuracy of 99.47% is also obtained by employing the RMSprop optimizer with a batch size of 10 and the same learning rate of 0.0001. However, the accuracy is reduced substantially to 92.89% if the batch size is 16 and the learning rate is increased to 0.001. The best accuracy of 99.47% is obtained by employing the SGDM optimizer with a batch size of 32 and the same learning rate of 0.001. However, the accuracy is reduced substantially to 95.00% if the learning rate is reduced to 0.00001. In conclusion, the best configuration for the ResNet18 model is the use of the RMSprop optimizer with a batch size of 32 and the same learning rate of 0.00001, which has produced the best accuracy while ensuring the efficiency of the model.

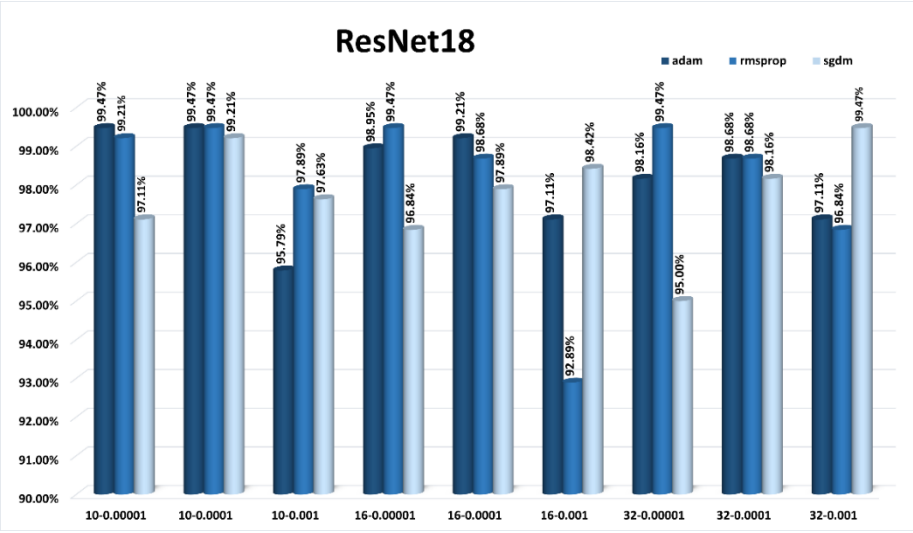


Fig. 3-5. Accuracy of ResNet18 with Adam, SGDM, and RMSprop optimizers for multiple batch sizes and learning rates.

3.4.1.3. GoogleNet architecture

Fig. 3-6 shows that Adam optimizer achieved its maximum accuracy of 99.47% when the batch size was 10 and the learning rate was 0.00001. Similar results were achieved when the batch size was 16 and the learning rate was 0.00001, and when the batch size was 32 and the learning rates were 0.00001 and 0.0001, with an accuracy of 99.21%. However, when the learning rate was elevated to 0.001and the batch size was 10, the accuracy fell to 97.11%. The RMSprop optimizer achieved the maximum accuracy of 99.47% when the batch size was 10 and the learning rate was 0.0001, and when the batch size was 16 and the learning rate was 0.00001. High accuracy was maintained at 99.21% when the batch size was 10 and the learning rate was 0.00001, and when the batch size was 32 and the learning rate was 0.0001. However, when the learning rate was elevated to 0.001 and the batch size was 10, the accuracy fell to 96.32%. The maximum accuracy for SGDM optimizer was 99.21% when the batch size was 32 and the learning rate was 0.0001. However, when the learning rate fell to 0.00001, the accuracy fell to 97.63%, irrespective of the batch sizes. This shows that SGDM optimizer is slightly more variant than the other two, with more dramatic changes. From the results, it is clear that when the batch size is 16 and the learning rate is 0.00001, the RMSprop optimizer with the GoogLeNet model has the optimal configuration since it has the highest accuracy of 99.47% and the shortest training time.

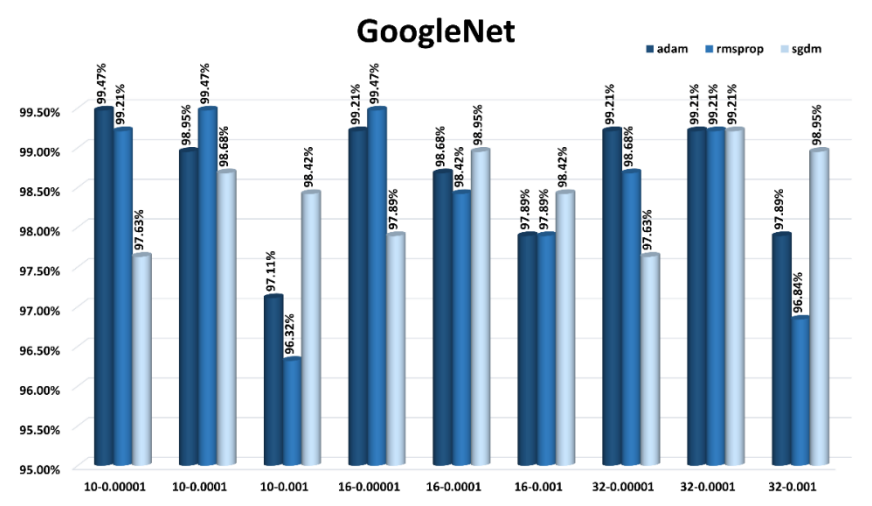


Fig. 3-6. Accuracy of GoogLeNet with Adam, SGDM, and RMSprop optimizers for multiple batch sizes and learning rates.

3.4.1.4. SqueezeNet architecture

As illustrated in Fig. 3-7, the Adam optimizer resulted in SqueezeNet achieving the maximum accuracy of 98.95% for a batch size of 10 and a learning rate of 0.00001 or 0.0001. There was a reduction in accuracy to 97.37% for a learning rate of 0.001 or a batch size of 32 and a learning rate of 0.00001. However, the Adam optimizer proved to be highly stable, especially for low learning rates, particularly for a batch size of 10. The RMSprop optimizer proved to be highly efficient, resulting in a maximum accuracy of 99.21% for a variety of configurations, including a batch size of 16 and a learning rate of 0.00001, and a batch size of 32 and a learning rate of 0.0001. The RMSprop optimizer proved to be highly stable, except for a significant reduction in accuracy to 87.89% for a batch size of 10 and a learning rate of 0.001. For the case of the SGDM optimizer, the highest accuracy of 99.21% was achieved with a batch size of 32 and a learning rate of 0.001. This optimizer also performed well with different settings, with the accuracy remaining close to 98.95% for a batch size of 10 and learning rates 0.00001 and 0.0001, as well as for a batch size of 16 and learning rate 0.0001. There was, however, a slight decline in accuracy to 97.11% for a batch size of 32 and learning rate 0.00001. From the results, it can be concluded that the RMSprop optimizer was the most effective optimizer for the SqueezeNet network, especially for a batch size of 32 and a learning rate of 0.0001. Though the accuracy for this network is slightly lower compared to that achieved by AlexNet and GoogLeNet, it remains high for different batch sizes and learning rates, with minimal reductions from 98.95% to 97.37%. This is quite different from the other networks, for which the accuracy can be as low as 50% if the chosen hyperparameter settings are inappropriate.

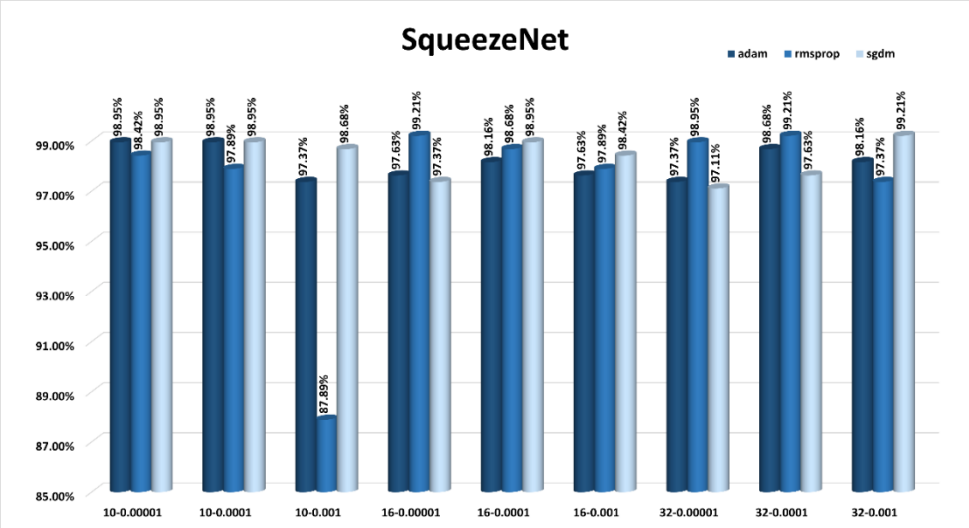


Fig. 3-7. Accuracy of SqueezeNet with Adam, SGDM, and RMSprop optimizers for multiple batch sizes and learning rates.

3.4.1.5. MobileNetV2 architecture

As depicted in Fig. 3-8, the Adam optimizer performed well, with the highest accuracy of 99.47% observed with two different setups, i.e., batch size of 10 and 16 with learning rates of 0.00001 and 0.0001, respectively. When the learning rate was set to 0.001 with a batch size of 16, the accuracy fell to 95.26%, proving the sensitivity of Adam to high learning rates. The RMSprop optimizer was observed to have the highest accuracy of 99.47%, with three different setups, i.e., batch sizes of 10, 16, and 32 with learning rates of 0.0001 and 0.00001, respectively. When the learning rate was set to 0.001 with a batch size of 10, the accuracy fell to 97.63%, proving the robustness and stability of the RMSprop optimizer with lower learning rates. The maximum accuracy of 99.21% was observed with several setups of the SGDM optimizer, including batch sizes of 16 and 32 with a learning rate of 0.001. The minimum accuracy of 92.89% was observed with a batch size of 32 and 0.00001 as the learning rate, proving the sensitivity of the SGDM optimizer to lower learning rates and batch sizes. From the observations, it is clear that lower to mid-range learning rates of 0.00001 and 0.0001 offer the best results, with batch size playing an insignificant role in affecting the accuracy. Among the three, the RMSprop optimizer with a learning rate of 0.00001 is found to be the most reliable, with the highest accuracy observed with different setups.

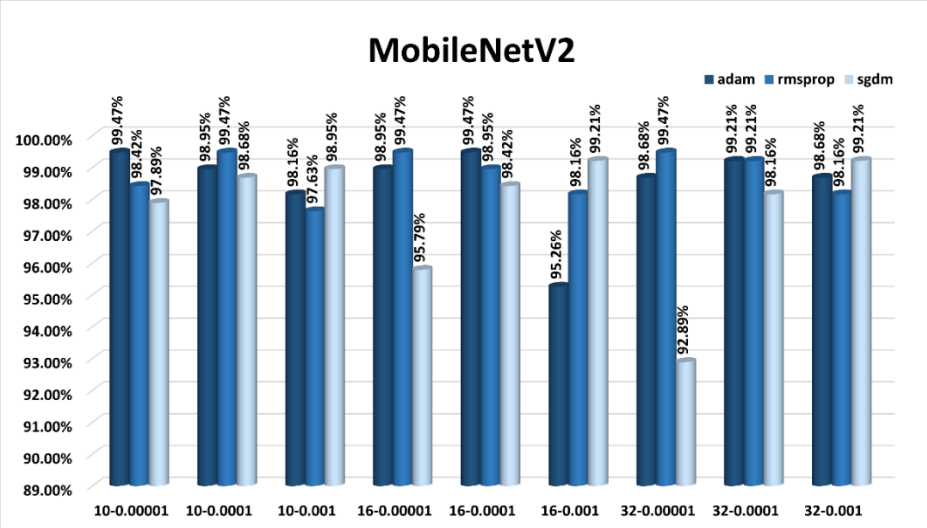


Fig. 3-8. Accuracy of MobileNetV2 with Adam, SGDM, and RMSprop optimizers for multiple batch sizes and learning rates.

3.4.1.6. EfficientNetb0 architecture

As depicted in Fig. 3-9, the accuracy of the Adam optimizer was found to have significant variability. The maximum accuracy obtained was 93.42% for a batch size of 32 with a learning rate of 0.0001. The second maximum accuracy obtained was 90.26% for a batch size of 16 with the same learning rate. The accuracy reduced for higher learning rates. The lowest accuracy obtained was 50.00% and 50.79% for batch sizes 10 and 16, respectively, with the same learning rate of 0.001. This shows the instability of the Adam optimizer for high learning rates. The accuracy of the RMSprop optimizer was found to have minimal variability. The maximum accuracy obtained was 96.32% for a batch size of 10 with the same learning rate of 0.0001. The second maximum accuracy obtained was 95.26% for a batch size of 16 with the same learning rate. The accuracy was found to be consistently high for a batch size of 32 with the same learning rate. However, the accuracy reduced to 50.00% for a batch size of 16 with the same high learning rate of 0.001. In the case of the SGDM optimizer, the highest accuracy of 99.47% was obtained with a batch size of 16 and a learning rate of 0.001. This was found to be the optimal configuration for the model based on the high accuracy it obtained and the lesser time it took for training. In addition, the model also showed high accuracy with other configurations, obtaining 98.95% with a batch

size of 10 and 98.42% with a batch size of 32, both with a learning rate of 0.001. When the learning rate was adjusted to 0.00001, the model showed a considerable decline in accuracy, obtaining only 76.05% and 82.63% for batch sizes of 16 and 32, respectively. This shows that the model, specifically the SGDM optimizer with a learning rate of 0.001, shows the highest accuracy compared to the other optimizers and configurations, making it the most optimal for the EfficientNetB0 model. Though the batch size has some effect on the model, the optimizer and learning rate show more impact on the model, with lower and mid-range learning rates showing better results compared to the lowest learning rate, which was 0.00001.

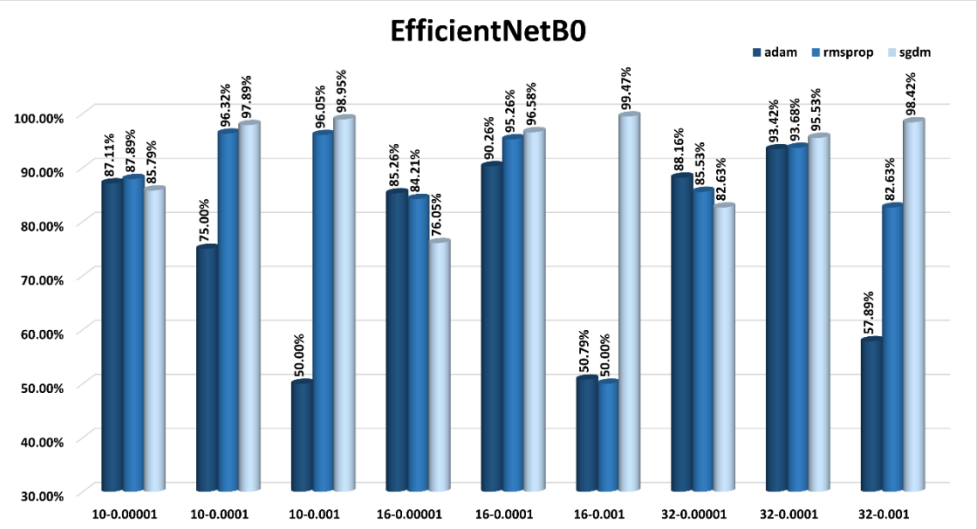


Fig. 3-9. Accuracy of EfficientNetB0 with Adam, SGDM, and RMSprop optimizers for multiple batch sizes and learning rates.

From the results, it is clear that the choice of the optimizer and learning rates significantly impacts the accuracy of CNNs in forest fire detection models. In all the models, RMSprop outperformed other optimizers in terms of accuracy. Specifically, RMSprop achieved the best accuracy of 99.74% for AlexNet, 99.47% for ResNet18, 99.47% for GoogLeNet, 99.21% for SqueezeNet, and 99.47% for MobileNetV2, thus proving to be a reliable optimizer for a wide range of CNN models in forest fire detection models. Although the batch size has a marginal impact on the accuracy of the models, larger batch sizes resulted in slightly improved performance in most models. On the other hand, the

Adam optimizer was highly affected by learning rates, especially 0.001, resulting in a significant reduction in accuracy, as seen in AlexNet and EfficientNetB0 models. In contrast, the SGD with momentum (SGDM) optimizer performs satisfactorily in most models, although there is a significant impact of larger batch sizes and learning rates.

From the results, it is clear that RMSprop, especially combined with a learning rate of 0.00001 or 0.0001, is the best optimizer in terms of accuracy, as most of the models achieved 99% or higher accuracy, thus indicating a strong potential in forest fire detection models.

In order to further explore the relationships between the key hyperparameters, Figure 3.10 shows a heat map illustrating the combined effect of the learning rate, batch size, and the performance of the CNN models for the six CNN architectures on the DeepFire dataset, using the RMSprop optimizer because of its excellent performance. Each CNN model was trained using a learning rate of 0.001, 0.0001, and 0.00001, along with a batch size of 10, 16, and 32. The analysis shows that the combination of a learning rate of 0.0001 and a batch size of 10 yields high accuracy for all the CNN models. However, slightly higher batch sizes, such as 16, often yield the best performance when the learning rate is kept low. For example, the AlexNet CNN yields the best accuracy of 99.74% when the learning rate is kept at 0.00001 and the batch size at 16. These results indicate that the best performance is achieved by using low to moderate values of the learning rate along with smaller batch sizes. However, slightly higher batch sizes yield a moderate reduction in the accuracy of the CNN models. These results emphasize the fact that the choice of the learning rate and batch size should be a good balance to achieve the best performance.

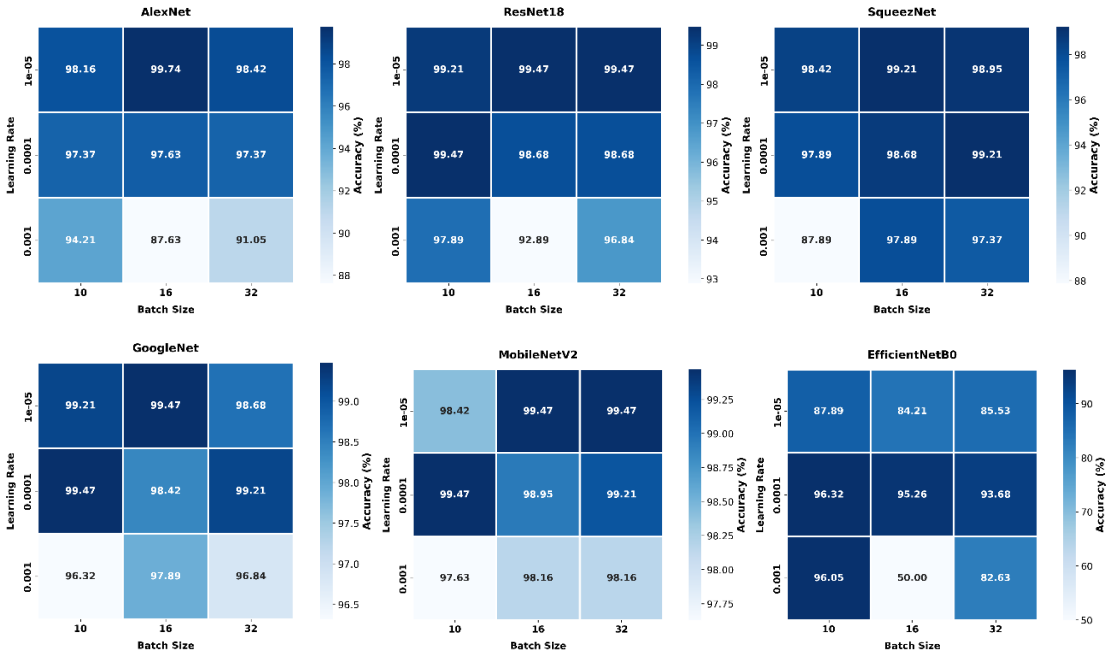


Fig. 3-10 Impact of Learning Rate and Batch Size on DeepFire Dataset Performance.

Table 3-1 shows the best configurations that were found for each of the pre-trained models that were looked at. This was accomplished by a comprehensive series of 27 experiments, done only on the DeepFire dataset, utilizing diverse combinations of hyperparameters. Due to the high computational time and resources needed, the search for the optimal combination of the aforementioned hyperparameters was not conducted for the larger Forest Fire Images dataset. In order to assess the ability for the selected architectures to generalize and be robust, the results obtained were extended for use with both datasets for the analysis that follows. In the following sections, a comprehensive analysis of the training dynamics and results for the six top architectures for both datasets will be presented, along with a comparison with the results obtained with the existing state-of-the-art DL techniques.

Table 3-1.Summary of the configuration of the top performing architectures.

Model	Optimizer	Learning rate	Batch size
AlexNet	RMSprop	0.00001	16
ResNet18	RMSprop	0.00001	32
SqueezeNet	RMSprop	0.0001	32
GoogLeNet	RMSprop	0.00001	16
MobileNetV2	RMSprop	0.00001	32
EfficientNetB0	SGDM	0.001	16

3.4.2. Training progress of the six top architecture

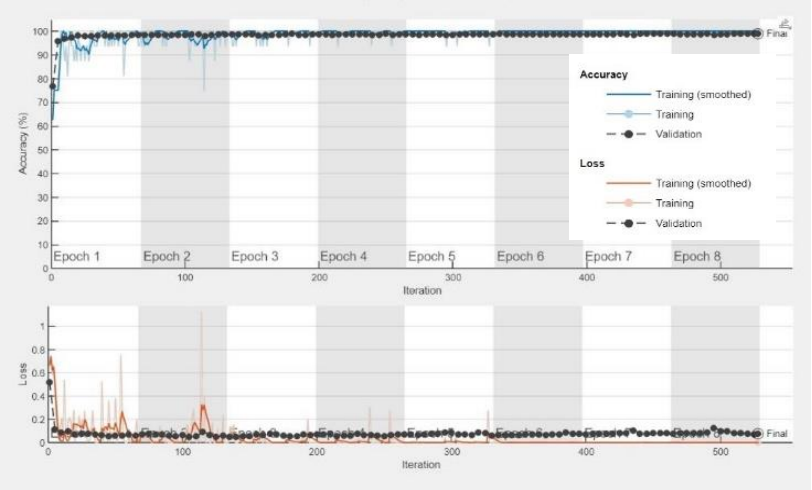
This subsection presents an analysis of the training progress for the optimal configuration of each network. Figures 3-11 to 3-16 illustrate the accuracy and loss trajectories over 8 training epochs for the two datasets during the training phase. These graphs provide important insights into the models' convergence behavior and generalization capabilities, helping to identify potential signs of overfitting or underfitting.

3.4.2.1. AlexNet

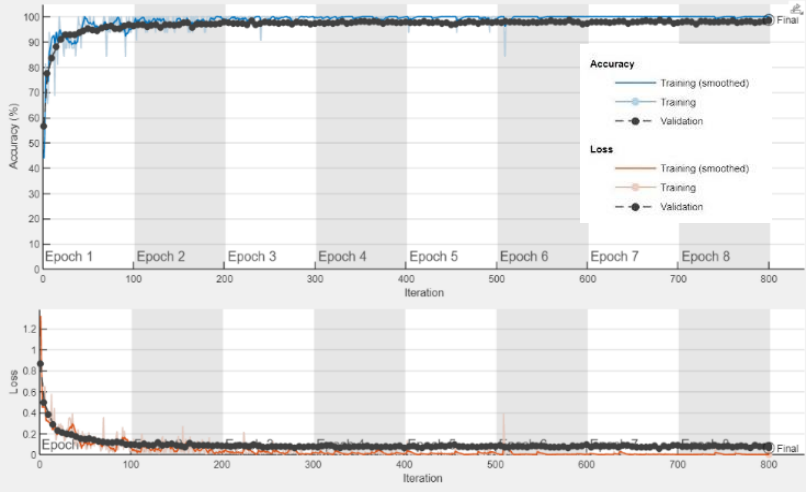
As shown in Fig. 3-11(a), the accuracy curve for the training process of AlexNet on the DeepFire dataset shows a sharp increasing trend at the beginning, quickly reaching a high accuracy level. After that, the accuracy levels off, which shows that the network has achieved convergence. Similarly, the training loss curve starts with a sharp decreasing trend at the beginning of the training process, which levels off after a few epochs. This shows that the network has achieved a good reduction of the training loss.

In Fig. 3-11(b), the training process of the AlexNet network on the Forest Fire Images dataset shows a gradual increasing trend. It starts at a moderate accuracy level and continues to increase over the successive epochs. This shows that the network has been able to effectively learn the relevant features of the dataset. However, the rate at which the

accuracy increases is slower compared to the DeepFire dataset. Similarly, the training loss starts at a higher level and continues to decrease over the successive epochs. It shows a smooth decreasing trend throughout the training process.



(a)



(b)

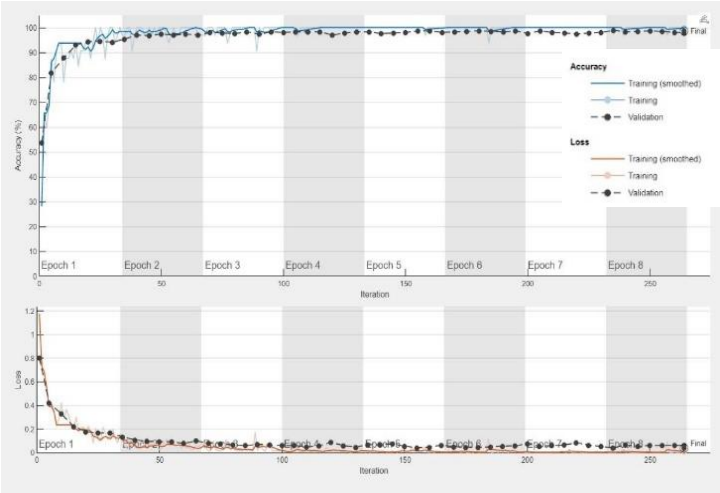
Fig. 3-11. Training Performance of AlexNet (Best Configuration) on Two Datasets: (a) DeepFire, (b) Forest Fire Images.

3.4.2.2. ResNet18

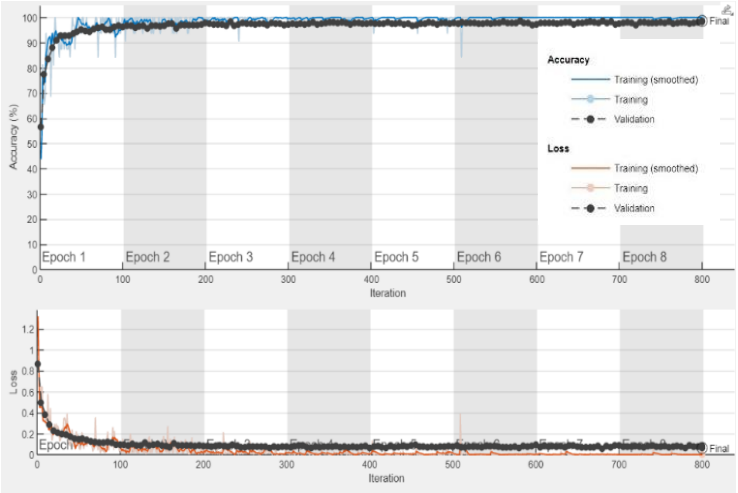
As can be seen in Fig. 3-12(a), the validation accuracy of ResNet-18 on the DeepFire dataset has a trend similar to the one seen in AlexNet, with high accuracy levels being reached consistently after the initial training stages. During the first few epochs, the model's validation loss drops significantly, after that, it stays rather constant with only small

variations. This shows that the model has successfully learned and generalized the data without any signs of overfitting.

As can be seen in Fig. 3-12(b), the ResNet-18 model trained on the Forest Fire Images dataset has efficiently and rapidly converged. The accuracy of the model increases rapidly during the initial epochs, which shows the model has quickly adapted to the task. The loss reduces rapidly and remains low during the training process. The curves are smooth and stable during the training process, which confirms the efficiency and robustness of the model configuration.



(a)



(b)

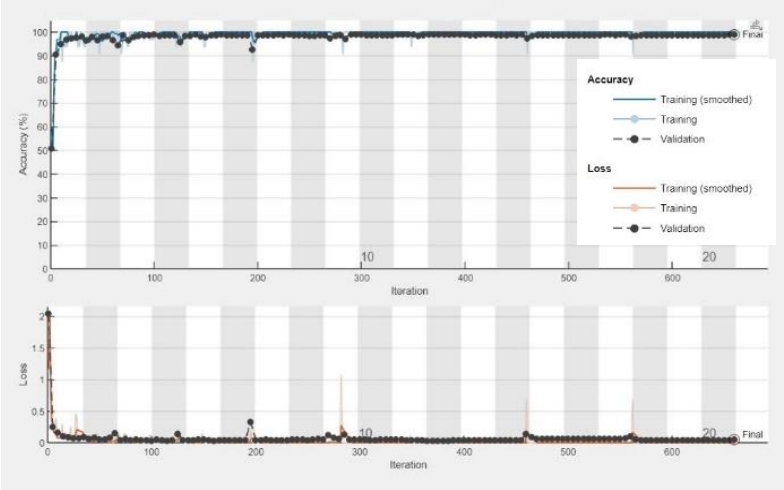
Fig. 3-12. Training Performance of ResNet18 (Best Configuration) on Two Datasets: (a) DeepFire, (b) Forest Fire Images.

3.4.2.3. SqueezeNet

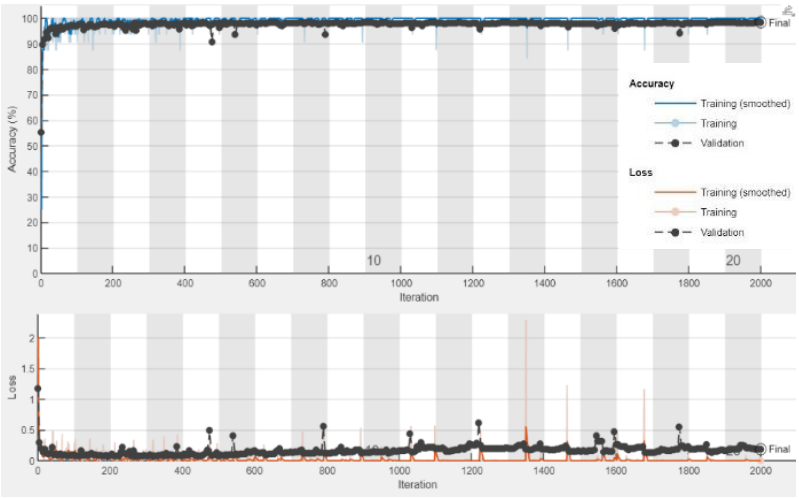
As shown in Fig. 3-13(a), the accuracy curve for the SqueezeNet model trained on the DeepFire dataset shows a sharp rising trend, with high accuracy achieved at the beginning of the training process. Although some fluctuations are noticed, the accuracy remains high throughout the training process. In contrast, the loss curve shows a sharp falling trend at the beginning, followed by a stable trend for the rest of the training process. Even though some fluctuations are noticed, the overall trend shows convergence as the model trains. In the accuracy curve, some fluctuations are more prominent at the beginning, while the overall trend shows convergence as the model trains.

From the accuracy and loss curves shown in Fig. 3-13(b) for the SqueezeNet model trained on the Forest Fire Images dataset, it can be concluded that the training process for this model shows a more volatile learning pattern compared to the other models. Although the model achieves a respectable accuracy, the learning pattern shows some instabilities,

which could be attributed to a high learning rate. In the accuracy graph, the training accuracy shows an increasing trend as the model trains, with some significant fluctuations. This shows that the model could be unstable, which could be improved by reducing the learning rate and employing some form of regularization. In the loss graph, the training loss shows a reducing trend as the model trains, with some fluctuations noticed throughout the training process.



(a)



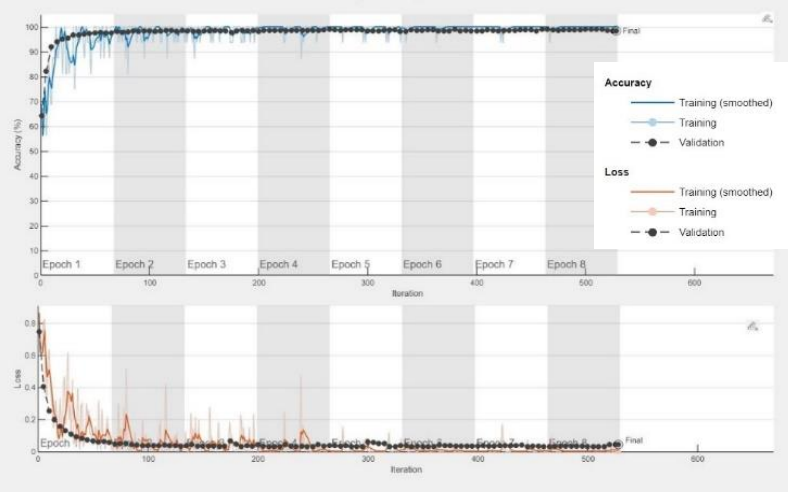
(b)

Fig. 3-13. Training Performance of SqueezeNet (Best Configuration) on Two Datasets: (a) DeepFire, (b) Forest Fire Images.

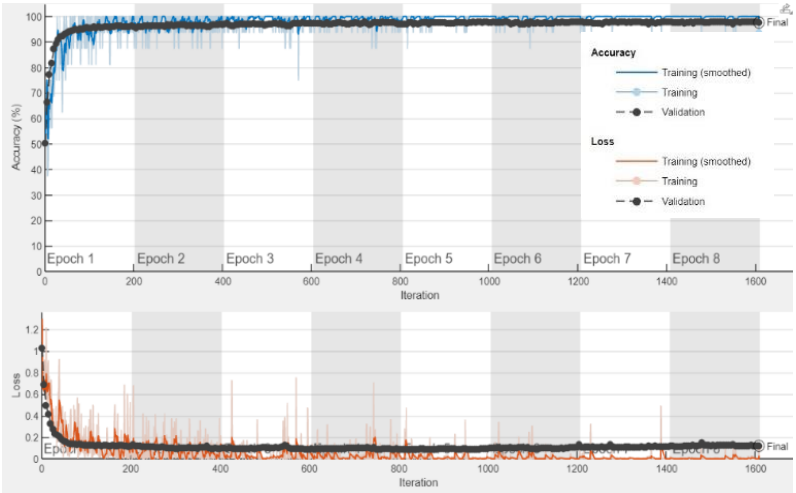
3.4.2.4. GoogleNet

Fig. 3-14 (a) shows the training process of the best configuration of GoogLeNet on the DeepFire dataset. As shown, the accuracy curve increases sharply. It almost reaches 100% within the first few epochs. Although some fluctuations exist, the accuracy stays at a high level, especially after the seventh epoch. The loss curve decreases sharply within the first few iterations. Some fluctuations exist, but they do not affect the stability of the training process.

Fig. 3-14 (b) shows the training process of the GoogLeNet network on the Forest Fire Images dataset. As shown, the accuracy curve stays at a high level throughout the training process. It shows a stable training process. During the training process, the accuracy continues to improve. It shows that the feature learning process of the GoogLeNet network performs well. As shown, the loss curve decreases smoothly. It shows that the GoogLeNet network performs well in minimizing the loss function. During the training process, no abrupt spikes exist. It shows that the training configuration of the GoogLeNet network was well tuned.



(a)



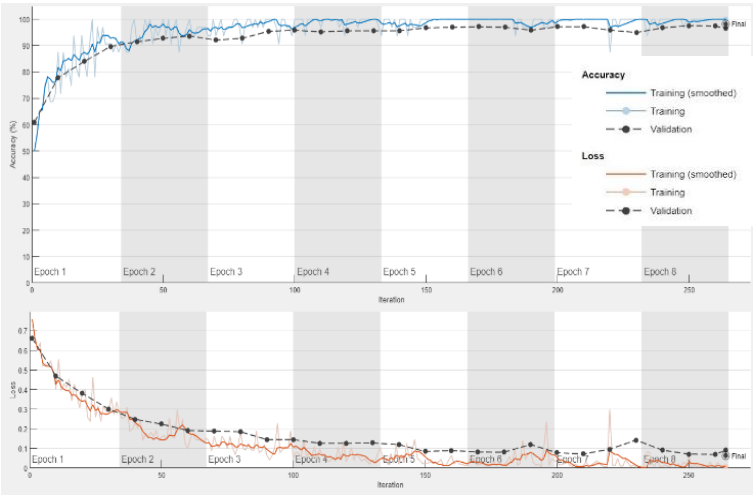
(b)

Fig. 3-14 Training Performance of GoogleNet (Best Configuration) on Two Datasets: (a) DeepFire, (b) Forest Fire Images.

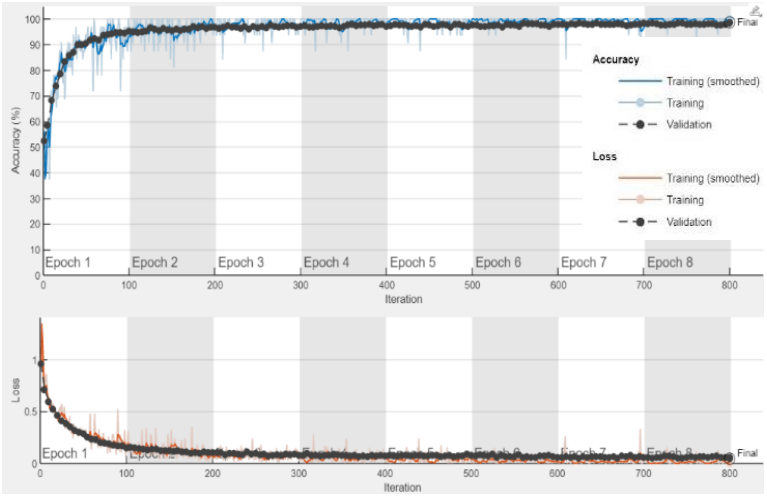
3.4.2.5. MobileNetV2

As shown in Fig. 3-15(a), the MobileNetV2 model trained on the DeepFire dataset shows a stable learning curve. The accuracy starts at a low value and increases incrementally as the training progresses through the epochs. This shows the strong learning capabilities of the MobileNetV2 model, which were successful. The loss starts at a relatively high value and continues to decrease incrementally as the training progresses through the epochs. The smooth increment of accuracy and loss shows that the optimization was successful and controlled, without any abrupt changes or spikes in the curve, or any signs of overfitting.

Fig. 3-15(b) shows the learning curve of the MobileNetV2 model trained on the Forest Fire Images dataset. The accuracy increases incrementally as the training progresses through the epochs, while the loss continues to decrease incrementally. Although small fluctuations are visible in the curve, the increment of accuracy and loss is smooth and continuous. This shows that the optimization was successful and controlled, without any signs of overfitting or divergence.



(a)

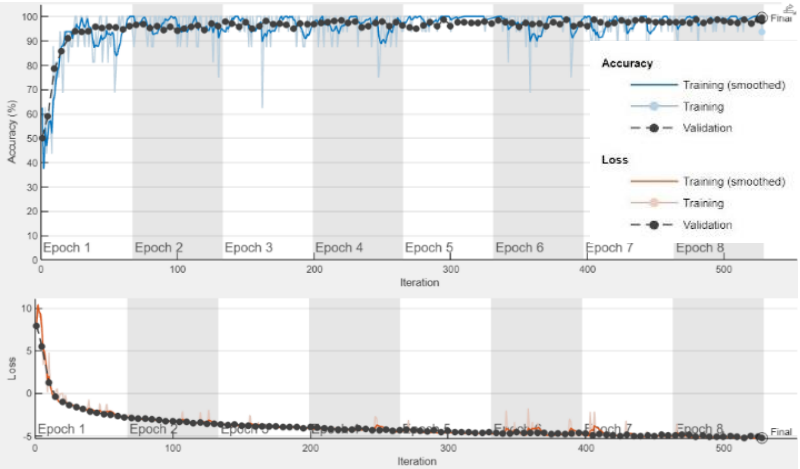


(b)

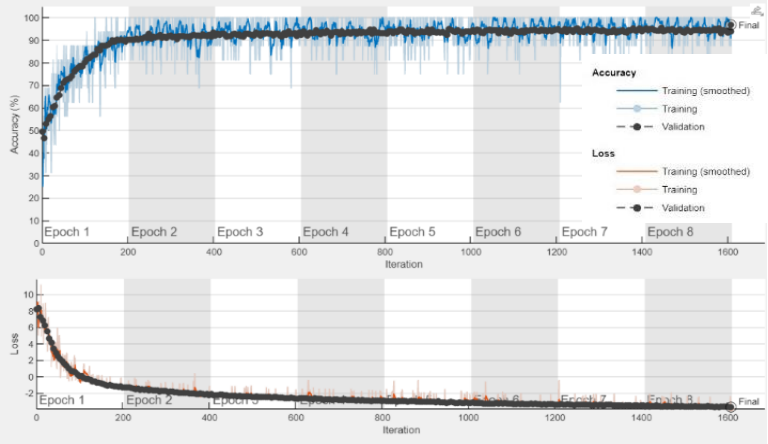
Fig. 3-15 Training Performance of MobileNetV2 (Best Configuration) on Two Datasets: (a) DeepFire, (b) Forest Fire Images.

3.4.2.6. EfficientNetb0

As can be seen in Fig. 3-16(a), the EfficientNetB0 model, while being trained on the data provided by the DeepFire dataset, has an effective training process. The accuracy of the model starts low and continues to rise steadily. This shows the high learning potential of the model. The loss level starts high and continues to dip steadily, reaching the lowest level towards the end of the training process. This shows the high level of success in the training process. The graph shows the high level of optimization in the model, which has not seen any problems or issues during the training process. In Fig. 3-16(b), the training process of the EfficientNetB0 model has been shown on the Forest Fire Images dataset. This is done to show the high level of stability and efficiency in the learning process. The accuracy level is consistently high during the training process. This shows the high learning potential of the model. The loss level is consistently low during the training process. This shows the high level of success in the training process.



(a)



(b)

Fig. 3-16. Training Performance of EfficientNetB0 (Best Configuration) on Two Datasets: (a) DeepFire, (b) Forest Fire Images.

3.4.3. Performance of the Six Top Architectures

In this subsection, the confusion matrices generated by the six pre-trained CNN models on both datasets are used to assess their performance.From this analysis, the model that is the most effective for forest fire detection can be determined based on the performance of the models. Fig. 3-17 below shows the confusion matrices on the DeepFire dataset.

- AlexNet (Fig. 3-17 a) performed best by classifying all 190 images of "fire" and 189 images of "no fire" correctly, with just one false positive.
- GoogLeNet (Fig. 3-17 b) classified all images of "fire" correctly but misclassified two images of "no fire" as images of "fire," resulting in slightly more false positives than AlexNet.
- ResNet18 (Fig. 3-17 c) performed almost as well as GoogLeNet by classifying 190 images of "fire" and 188 images of "no fire" correctly, resulting in two false positives.
- SqueezeNet (Fig. 3-17 d) classified 189 images of "fire" and 188 images of "no fire" correctly. Out of the three false positives, one image of "fire" was classified as an image of "no fire," while two images of "no fire" were classified as images of "fire."
- MobileNetV2 (Fig. 3-17 e) performed very well by classifying 189 images of "fire" and 188 images of "no fire" correctly, resulting in one false positive for each class.

- EfficientNetB0 (Fig. 3-17 f) classified all 190 images of "fire" and 188 images of "no fire" correctly, resulting in two false positives.

It is evident from Fig. 3-17 that AlexNet performed the best on the DeepFire dataset by classifying all images of "fire" and all images of "no fire" except for one false positive. AlexNet is the most accurate model for classifying images of forest fires and images of "no forest fire."

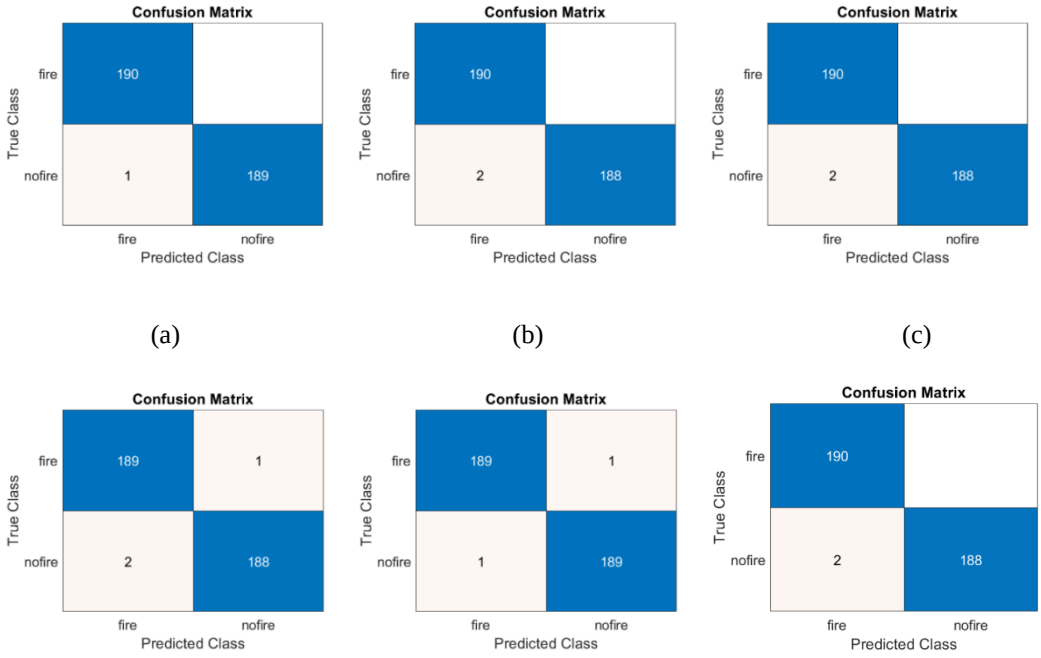


Fig. 3-17. Confusion matrices of the top six networks on the DeepFire dataset.

Fig. 3-18 shows the confusion matrices for the top-performing models, which were then tested and validated for the Forest Fire Images dataset.

- ResNet18 (Fig. 3-18 a): The ResNet18 model showed outstanding results, with all 25 instances of the "fire" class and 24 instances of the 25 instances of the "no fire" class being correctly classified. The model had a single false positive, which corresponded to a "no fire" image that it misclassified as a "fire" image. There were no instances of false negatives.
- AlexNet (Fig. 3-18 b), MobileNetV2 (Figure 3.18c): The AlexNet and MobileNetV2 models correctly classified all instances of the "fire" class and 23 instances of the 25 instances of the

"no fire" class, with the model misclassifying two instances of the "no fire" class as a "fire" image.

- GoogLeNet (Fig. 3-18 d): The GoogLeNet model had a perfect recall for the "fire" class, with all instances being correctly classified. However, it had a higher number of false positives, misclassifying three instances of the "no fire" class.
- SqueezeNet (Fig. 3-18 e): The SqueezeNet model correctly classified all instances of the "fire" class, while it had a higher number of false positives, misclassifying four instances of the "no fire" class.
- EfficientNetB0 (Fig. 3-18 f): The EfficientNetB0 model had errors for both classes, with 23 instances of the "fire" and 23 instances of the "no fire" class being correctly classified, while it had a total of four errors, with two instances of the "fire" class being misclassified as instances of the "no fire" class and two instances of the "no fire" class being misclassified as instances of the "fire" class.

The ResNet18 model had the best performance for the Forest Fire Images dataset, with a single misclassification and a shorter training time compared with the other models. The ResNet18 model had a high precision for.

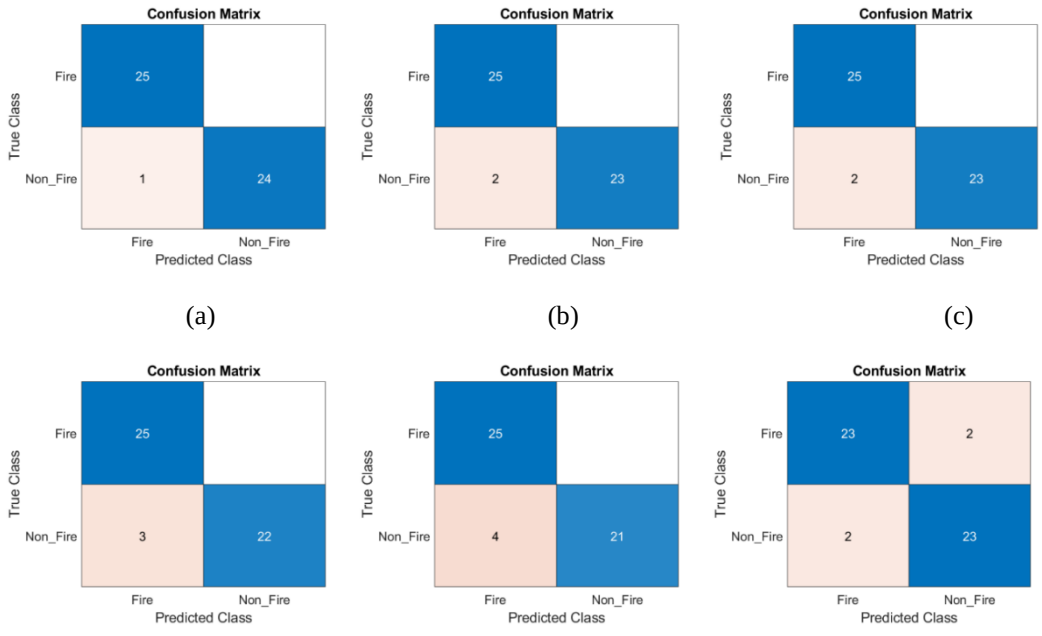


Fig. 3-18. Confusion matrices of the top six networks on the Forest Fire Images dataset.

To evaluate the overall classification performance, the ROC curve of the top six architectures is provided in Fig. 3-19.

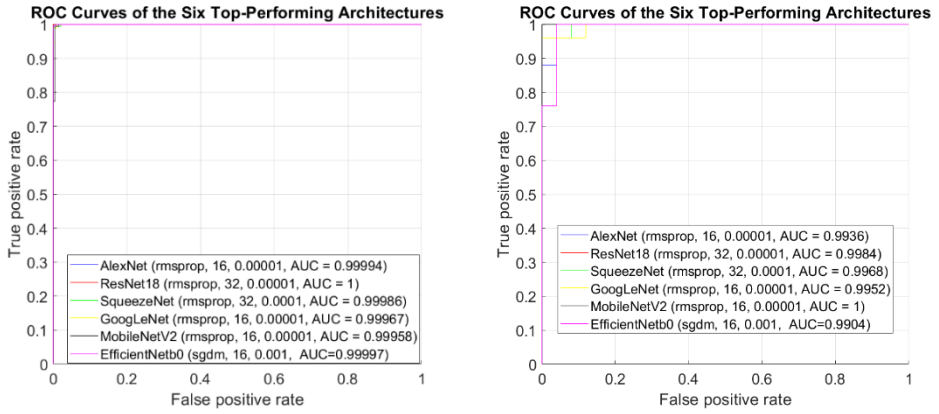


Fig. 3-19.ROC curves of the Top-performing architecture for each network on both datasets.

Fig. 3-19(a) presents the ROC curve of the top-performing networks on the Deepfire dataset. The ROC curve shows that all networks have high discriminative capabilities, as indicated by the AUC values approaching 1, meaning high classification accuracy. The ROC curve is close to the top-left corner, meaning high true-positive and low false-positive values. Although the ResNet18 network has an AUC of 1.0, meaning high classification accuracy, the confusion matrix indicates that there are two false positives. The AUC is threshold-independent, and the confusion matrix is based on a threshold of 0.5. The overall ROC curve suggests that the selected networks have high classification accuracy for forest fire detection. Fig. 3-19(b) presents the ROC curve of the top-performing networks on the Forest Fire Images dataset. The ROC curve of the top-performing networks indicates that all networks have high discriminative capabilities, as indicated by the AUC values approaching 1, meaning high classification accuracy. The ROC curve is close to the top-left corner, meaning high true-positive and low false-positive values. The overall ROC curve suggests that the selected networks have high classification accuracy for forest fire detection.

Four main performance indicators were used to provide a more complete picture of how well the chosen DL models worked: Accuracy, Precision, Recall, and F1-Score. Figures 3-20 and 3-21 show the findings as bar charts. They show how well each model did in its best setup on both datasets.

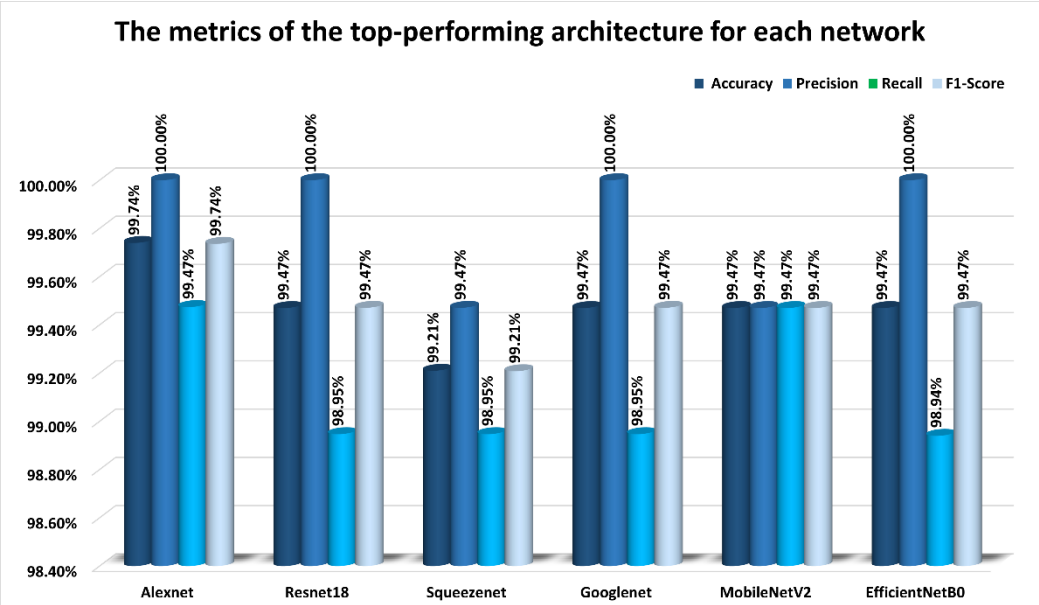


Fig. 3-20. Performance Metrics of the Best Configuration of Each Network on Deep-Fire Dataset.

From Fig.3-20, it is clear that all six models, perform very well in classifying the images in the dataset. AlexNet performs the best, achieving 99.74% accuracy, 100% precision, 99.47% recall, and 99.74% F1 score, indicating a very accurate classification of images with very few false negatives. ResNet18 follows closely behind, achieving 99.47% accuracy and 100% precision, with a 98.95% and 99.47% F1 score, respectively, indicating a very impressive performance, although slightly lower than AlexNet. SqueezeNet achieves 99.21% accuracy, 99.47% precision, 98.95% recall, and 99.21% F1 score, indicating a moderate performance, although slightly lower than the other models. MobileNetV2 showed consistent and balanced results, with 99.47% for accuracy, precision, recall, and F1 score. EfficientNetB0 showed comparable results to GoogLeNet, with an accuracy of 99.47%, precision of 100%, recall of 98.94%, and an F1 score of 99.47%, thus proving the robustness of the classification model.

Thus, it can be concluded that AlexNet performed the best out of all the models, with the highest values for classification and proving to be the most effective model for the DeepFire dataset. The high accuracy of AlexNet is further supplemented by the fact that it performed with the least training time of 56 minutes and 33 seconds, compared to ResNet18, which took 66 minutes and 29 seconds; SqueezeNet, which took 69 minutes and 2 seconds; GoogLeNet, which took 101 minutes and 23 seconds; MobileNetV2, which took 101 minutes and 26 seconds; and EfficientNetB0, which took 255 minutes and 4 seconds.

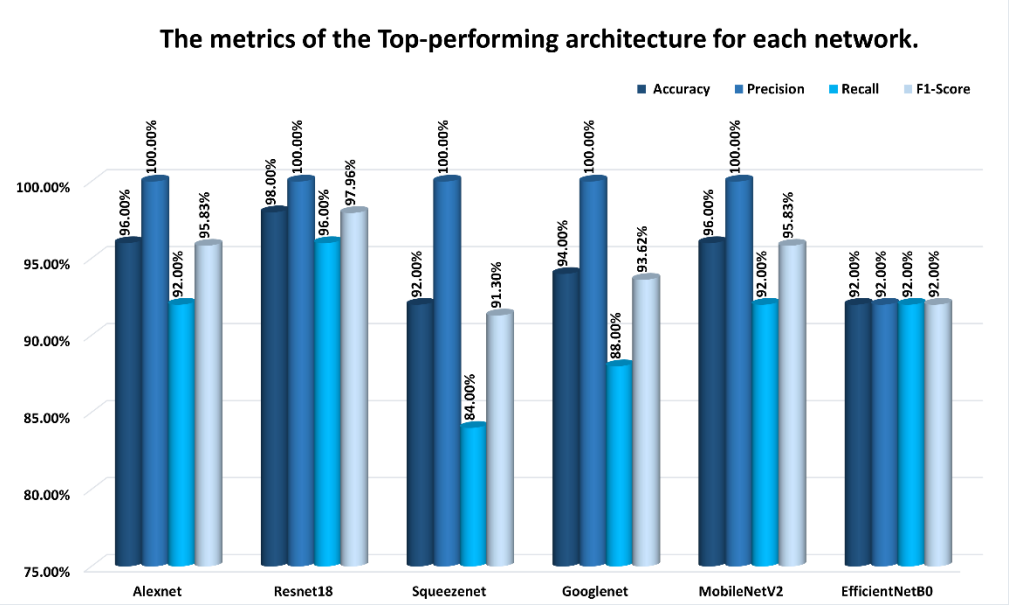


Fig. 3-21. Performance Metrics of the Best Configuration of Each Network on Forest Fire Images dataset.

Fig. 3-21 provides a comparative overview of the performance of the six CNN architectures, when implemented and tested on the Forest Fire Images dataset with the optimal hyperparameters. The dataset comprises 50 photos for testing, with each image contributing 2% to the overall accuracy and metric score.

The ResNet18 architecture showed the best performance, with an accuracy score of 98.00%, precision score of 100.00%, recall score of 96.00%, and F1 score of 97.96%. The AlexNet and MobileNetV2 architectures showed almost similar performance, with an accuracy score of 96.00%, precision score of 100.00%, recall score of 92.00%, and F1 score of 95.83%, indicating that these architectures have high precision and slightly low recall.

The GoogLeNet architecture showed a slightly low performance, with an accuracy score of 94.00%, precision score of 100.00%, recall score of 88.00%, and F1 score of 93.62%. The EfficientNetB0 architecture showed a consistent performance, with an accuracy score, precision score, recall score, and F1 score all at 92.00%. The SqueezeNet architecture showed a consistent performance, with an accuracy score, precision score, recall score, and F1 score all at 92.00%, while showing 100.00% precision, 84.00% recall, and 91.30% F1 score, indicating that it has a tendency towards high precision and low recall.

Therefore, ResNet18 has shown the best performance for the Forest Fire Images dataset, exhibiting the most superior assessment metrics compared to other architectures. The ResNet18 architecture has shown a better trade-off between classification and computational efficiency, with a training time of 7 hours and 36 minutes, while the AlexNet architecture has shown a faster training time of 5 hours and 7 minutes. The ResNet18 architecture has shown better performance than other architectures, namely SqueezeNet (7 hours 41 minutes), GoogLeNet (11 hours 49 minutes), MobileNetV2 (9 hours 33 minutes), and EfficientNetB0 (25 hours 3 minutes).

In addition, the computational complexities of all models were evaluated in terms of two critical architectural characteristics: the total number of floating-point operations (FLOPs) required to perform a single inference and the total number of parameters in each model. These characteristics are presented in Table 3-2. It is important to note that, for all pre-trained architectures, no layers were frozen during the transfer learning process. Instead, only the original final classification layer was replaced with a new output layer consisting of two neurons corresponding to the binary classes (fire and no fire). Consequently, all network parameters remained trainable, resulting in zero non-trainable parameters across all models and enabling full end-to-end fine-tuning of each architecture.

As shown in Table 3-2, AlexNet has the largest number of parameters, approximately 56.9 million, while it has a relatively low computational complexity of 0.43 GFLOPs because of its shallow architecture. On the other hand, ResNet-18 and GoogLeNet have a much higher computational complexity, approximately 1.79 and 2.53 GFLOPs, respectively, in comparison to their parameters because of their deeper and more intricate convolutional architecture. SqueezeNet has the fewest parameters, approximately 0.72 million, while it

has a relatively high computational complexity of 1.70 GFLOPs because of the presence of many 1x1 and 3x3 convolutional layers in the architecture of the model. MobileNetV2 has a low computational complexity of 0.31, in addition to a low parameter count of 2.2 million, making it a highly efficient model for deployment in resource-constrained environments, while EfficientNetB0 has a moderate parameter count of 5.3 million, in addition to a moderate computational complexity of 0.73 GFLOPs, because of the use of a compound scaling technique to improve efficiency in the model architecture.

Table 3-2. Computational complexity (in Gflops) and number of parameters of the best-performing CNN models.

Models	Flops (Gflops)	Number Parameters
AlexNet	0.431675	56,876,418
ResNet18	1.792617	11,172,738
GoogLeNet	2.529371	5,975,602
SqueezeNet	1.703077	723,522
MobileNetV2	0.309630	2,209,378
EfficientNetB0	0.731000	5,269,542

3.4.4. Cross-dataset validation

To assess the generalization potential of the selected DL models, a cross-dataset validation was carried out. For this evaluation, each of the models was trained on a given dataset (DeepFire or Forest Fire Images), after which it was tested on the other dataset. This provides a deeper understanding of the potential of the models to maintain their performance on datasets that are defined by different attributes, such as resolution, type of fire, scene complexity, and environmental conditions.

Table 3-3.Cross-Dataset evaluation of model accuracy.

Models	Trained on	Tested on	Accuracy %
AlexNet	DeepFire	Forest Fire Images (test)	94.00
	DeepFire	Forest Fire Images (train)	95.81
	Forest Fire Images	DeepFire (test)	98.68
	Forest Fire Images	DeepFire (train)	98.16
ResNet18	DeepFire	Forest Fire Images (test)	96.00
	DeepFire	Forest Fire Images (train)	96.27
	Forest Fire Images	DeepFire (test)	98.42
	Forest Fire Images	DeepFire (train)	98.09
SqueezeNet	DeepFire	Forest Fire Images (test)	94.00
	DeepFire	Forest Fire Images (train)	96.29
	Forest Fire Images	DeepFire (test)	96.85
	Forest Fire Images	DeepFire (train)	97.30
GoogLeNet	DeepFire	Forest Fire Images (test)	96.00
	DeepFire	Forest Fire Images (train)	96.35
	Forest Fire Images	DeepFire (test)	97.89
	Forest Fire Images	DeepFire (train)	97.57
MobileNetV2	DeepFire	Forest Fire Images (test)	90.00
	DeepFire	Forest Fire Images (train)	94.47
	Forest Fire Images	DeepFire (test)	98.42
	Forest Fire Images	DeepFire (train)	97.57
EfficientNetB0	DeepFire	Forest Fire Images (test)	94.00
	DeepFire	Forest Fire Images (train)	95.81
	Forest Fire Images	DeepFire (test)	97.89
	Forest Fire Images	DeepFire (train)	97.89

The results obtained, as presented in Table 3-3, show that there are various architectures, including ResNet18 and AlexNet, which show robust cross-dataset generalization ability. For example, AlexNet, which obtained 99.74% accuracy in the evaluation of the DeepFire dataset, showed a high level of robustness in terms of performance, even after being trained on the Forest Fire Images dataset and evaluated on the DeepFire dataset, obtaining 98.68% accuracy. When the cross-dataset evaluation was carried out on the DeepFire dataset, AlexNet showed a high level of robustness in terms of accuracy, obtaining 98.16%. Similarly, ResNet18 obtained 98.42% accuracy in the evaluation of the Forest Fire Images dataset, and 96.00% accuracy in the evaluation of the DeepFire dataset after being trained on Forest Fire Images and evaluated on DeepFire, respectively. However, other architectures, including GoogLeNet and SqueezeNet, showed robust performance in terms of accuracy in the evaluation of various datasets. However, in the evaluation of the data from the unseen dataset, MobileNetV2 showed a significant reduction in accuracy, to 90.00%. From the results obtained, it is clear that there are various architectures, including ResNet18 and AlexNet, which are robust in terms of performance, making them very useful in real-world forest fire detection scenarios, considering the fact that there are many discrepancies in terms of data distribution in real-world scenarios.

3.4.5. Performance comparison against state-of-the-art techniques

In this subsection, the proposed fire detection framework is compared with several recent DL and TL approaches reported in the literature. The comparison utilizes four assessment metrics: accuracy, precision, recall, and F1 score. The benchmark methodologies including those suggested by Ali Khan et al. [78], Islam et al. [89], Wang et al. [90], Khan and Khan [91] and Prakash et al. [92]. Tables 3-4 and 3-5 summarize the comparative results obtained on the DeepFire and Forest Fire Images datasets, respectively. All methods listed in Table 3-4 were evaluated on the DeepFire dataset, whereas those presented in Table 3-5 were assessed using the Forest Fire Images dataset.For a fair comparison, the performance of the proposed approach corresponds to the best configurations identified in the preceding experimental analysis. Specifically, on the DeepFire dataset, the reported results are based on the AlexNet architecture optimized with the RMSprop algorithm, a batch size of 16, and a

learning rate of 0.00001. On the Forest Fire Images dataset, the reported results correspond to the ResNet18 architecture trained using the same optimizer and learning rate, with a batch size of 32. These configurations were selected because they yielded the highest overall performance in the within-dataset evaluations.

Table 3-4.Comparison of the proposed approach against State-of-the-Art techniques on the DeepFire dataset.

Technique	Accuracy %	Precision %	Recall %	F1-score %
AlexNet	99.74	100	99.47	99.74
FFireNet [91]	98.42	97.42	99.47	98.43
RBFN-RAISR [92]	97.55	94.19	96.44	93.33
Efficientnetb7+ACNet [89]	95.97	95.19	96.01	95.54
VGG19 [78]	95.00	95.72	94.21	94.96
Reduce-VGGNet [90]	91.20	97.22	97.22	97.22

As shown in Table 3-4, the proposed approach using AlexNet has the best overall performance compared to the other approaches, with an accuracy of 99.74%, precision of 100.00%, recall of 99.47%, and an F1 score of 99.74%. The results of the proposed approach, therefore, not only indicate perfect classification but also an excellent balance between sensitivity and specificity. Among the approaches, the FFireNet [91] model is observed to have good overall performance, especially in terms of recall and F1 scores of 99.47% and 98.43%, respectively. The RBFN-RAISR [92] and Efficientnetb7+ACNet [89] approaches have good accuracy of 97.55% and 95.97%, respectively. The Reduce-VGGNet [90] model has high precision and recall of 97.22%, but its overall accuracy is lower, with an accuracy of 91.20%.The VGG19 [78] model, however, has an accuracy of 95.00%, precision of 95.72%, recall of 94.21%, and an F1 score of 94.96%, which is an indication of its reliability. Although it is clear that the VGG19 model is reliable and can

be considered effective in forest fire detection, it is still lower compared to the proposed approach, especially in recall and overall accuracy. It is, therefore, clear that the overall assessment of the proposed approach is reliable and effective, especially in forest fire detection scenarios, where high accuracy, perfect precision, and high recall are essential.

Table 3-5.Comparison of the proposed approach against State-of-the-Art techniques on the Forest Fire Images dataset.

Technique	Accuracy %	Precision %	Recall %	F1-score %
ResNet18	98.00	100	96.00	97.95
MobileNet	97.87	98.02	98.13	98.08
DenseNet	97.50	97.84	97.62	97.73
Xception	94.09	94.58	94.84	94.71
NASNetMobile	85.09	84.13	89.98	86.95
ResNet50V2	84.89	85.20	87.68	86.42
InceptionV3	83.23	83.47	87.21	85.30
Inception ResNetV2	55.19	55.19	100	71.09

As shown in Table 3-5, the proposed approach based on ResNet18 achieves the best classification accuracy of 98.00%. It is also observed that the proposed approach achieves a high level of precision equal to 100%, which confirms that there are no false positives during the performance evaluation process. Though the recall level is slightly lower at 96.00%, the overall performance is satisfactory with an F1 score equal to 97.95%.All the comparative results provided in Table 3-5 are referenced from Yandouzi et al. [88]. Considering all the architectures discussed by Yandouzi et al., it is observed that the performance of the MobileNet and DenseNet architectures is comparable to the proposed approach with an F1 score equal to 98.08% and 97.73%, respectively. Xception architecture ranks third with an F1 score equal to 94.71%, while its overall accuracy

reduces to 94.09%. On the other hand, NASNetMobile, ResNet50V2, and InceptionV3 achieve moderate performance with an F1 score ranging from 85.30% to 86.95%, which corresponds to lower values of precision and recall. It is observed that the performance of the Inception-ResNetV2 architecture is the worst, with an overall accuracy of 55.19% and an F1 score equal to 71.09%. Though this architecture achieves 100% recall, its low precision reduces its overall performance significantly. It can be concluded that the proposed approach based on ResNet18 outperforms all other architectures based on the overall performance criteria and confirms its high level of robustness for forest fire detection applications.

3.5. Conclusion

This chapter presented the methodology for forest fire detection using TL with several state-of-the-art CNN architectures, including AlexNet, GoogLeNet, ResNet18, SqueezeNet, EfficientNetB0, and MobileNetV2. Each model was carefully fine-tuned for binary fire/no-fire classification by modifying the final layers and testing different hyperparameter configurations to determine the optimal setup.The chapter also outlined the experimental framework, including the datasets used, performance evaluation metrics, and hardware and software specifications, ensuring a fair and reproducible comparison among all models.

Overall, this methodology provides a solid foundation for assessing detection performance and computational efficiency, forming a comprehensive framework for studying and analyzing the results of the different models.

4. Low-complex CNN for forest fire detection

4.1. Introduction

This chapter presents the proposed low-complexity CNN architecture developed for forest fire detection. It begins by describing the design of the CNN model and the experimental setup used to evaluate its performance. The chapter then discusses the impact of various training parameters, the model's performance on two different datasets, and the training progress throughout different configurations. Additionally, A comparison with existing methods in the literature is presented to highlight the effectiveness of the proposed approach.

4.2. The proposed CNN architecture

In this subsection, a lightweight CNN architecture is proposed for binary forest fire detection. Training and evaluation of the proposed model are carried out using the DeepFire and Forest Fire Images datasets, which are also applied in the TL experiments. As illustrated in Fig.4-1, the proposed system follows a structured processing pipeline consisting of image preprocessing, classification, and embedded deployment. Each stage of the proposed approach is described in detail below.

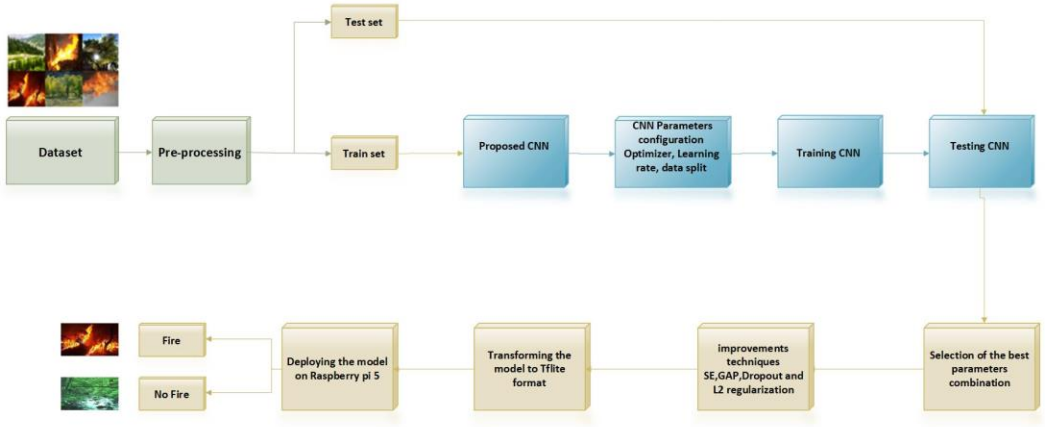


Fig. 4-1.The proposed CNN for forest fire detection.

4.2.1. Dataset and image preprocessing

The DeepFire and Forest Fire Images datasets were employed in the same manner as described in the previous chapter, maintaining their original class definitions and data splits. For further details regarding these datasets and their preparation, please refer to Subsections 3.3.2.1 and 3.3.2.2.

4.2.2. Lightweight CNN architecture design

Following preprocessing, the images are fed into a lightweight CNN specifically designed to minimize computational complexity while maintaining effective feature extraction capability. The proposed base architecture consists of three convolutional layers for hierarchical feature extraction, followed by a single fully connected layer of only two neurons for binary classification. The convolutional layers progressively learn discriminative features associated with forest fire patterns, including color intensity variations, texture irregularities, and spatial structures. Nonlinear activations follow every convolutional layer to enable the model to capture complex patterns, while pooling layers help decrease spatial resolution and reduce computational complexity. The fully connected layer aggregates the extracted features and produces the final classification output.

4.2.3. Integration of enhancement techniques

To enhance generalization and robustness, several architectural improvement techniques are individually incorporated into the base model. Each technique is evaluated independently to isolate its impact on performance. Dropout is applied during training to reduce overfitting by randomly deactivating neurons. L2 regularization is introduced to penalize large weight values and improve generalization. Squeeze-and-Excitation (SE) blocks are incorporated to adaptively recalibrate channel-wise feature responses by highlighting important features and reducing the influence of less relevant ones. Global Average Pooling (GAP) is employed as an alternative to traditional fully connected layers to reduce the number of trainable parameters while preserving discriminative information. Each enhanced variant is trained and evaluated separately under the same experimental conditions. (Detailed descriptions of these techniques are provided in Section 2.2.5).

4.2.4. Training and evaluation protocol

The lightweight CNN models are trained and evaluated using the same training and testing partitions defined for the datasets, but without any parameter sharing, initialization, or dependency on the TL experiments. All models are trained from scratch. During training, performance is monitored using validation data to assess convergence and classification effectiveness. In the evaluation phase, the trained models classify input images into fire or no fire categories. Performance is measured using standard binary classification metrics derived from the confusion matrix (Section 3.2.3).

4.2.5. Embedded deployment and efficiency evaluation

To assess real world applicability, all trained lightweight CNN variants are converted to TFLite format and deployed on a Raspberry Pi 5 platform. The deployed models are evaluated in terms of inference latency**,** memory consumption, and real time detection capability, providing insight into their suitability for practical forest fire monitoring systems operating under limited computational resources. (see Section 4.3.3)

4.2.6. Output classification

The final layer of the proposed system employs a Softmax activation function to produce probability scores for the two classes (fire and no fire). Based on the highest probability value, each input image is assigned a binary decision label. This output can be directly integrated into real-time forest fire detection and early warning systems.

4.3. Experimental setup

In this section, we present the experimental setup of the proposed architecture, detailing the dataset, CNN configuration, evaluation metrics for model performance, and the hardware and software specifications.

4.3.1. The CNN setup

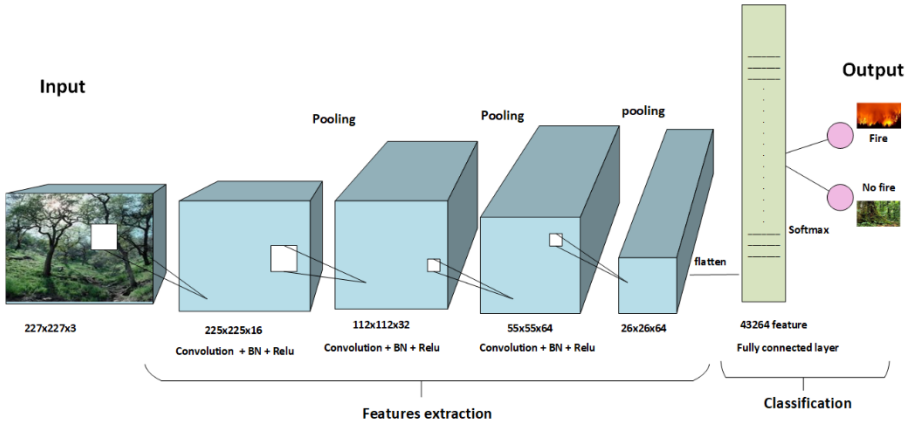


Fig. 4-2.The proposed CNN architecture.

A customized CNN architecture was developed to specifically tackle the task of forest fire detection. This architecture combines high accuracy with efficient computation, making it ideal for real-time applications, such as surveillance and early fire detection systems. To assess the model's generalization capabilities across various fire scenarios and image complexities, training and evaluation were performed on both the DeepFire and Forest Fire Images datasets. The architecture, shown in Fig. 4-2, includes the following layers:

- **Convolutional Layer 1**: 16 filters of size 3×3 to extract initial visual characteristics including edges and textures from the input images.
- **Batch Normalization**: Introduced after every convolutional layer to stabilize the learning process and accelerate convergence**.**
- **Activation Function**: ReLU is employed to incorporate non-linear behavior, which helps the model learn more sophisticated representations.
- **Max Pooling Layer**: Using a 2×2 max-pooling layer with stride 2 reduces the spatial resolution and computational requirements, while keeping the most significant features.
- **Convolutional Layers 2 and 3**: These layers use 32 and 64 filters respectively to capture higher-level, more abstract features related to fire detection.
- **Fully Connected Layer**: The feature maps are reshaped into a vector and forwarded to a fully connected layer with two output neurons representing the fire and non-fire classes.

- **SoftMax Activation**: The Softmax function is used in the output layer to transform the model's outputs into class probabilities.

This architecture was designed to progressively extract both low- and high-level features, ensuring it can detect fire effectively across various environments. Its compact design makes it computationally efficient, ensuring adaptability in real-world applications with limited hardware resources.

4.3.2. Datasets description

This study employs the same datasets used in the TL experiments described in Section 3.3.2 of the previous chapter in order to maintain consistency and enable fair comparisons across all experimental settings. In particular, the experiments are conducted using the DeepFire dataset and the Forest Fire Images dataset, which are widely recognized benchmarks for image-based forest fire detection. These datasets consist of labeled images belonging to fire and no fire classes and present substantial variability in terms of scene complexity, illumination conditions, background characteristics, and fire appearance. The use of identical datasets throughout the different experimental phases enables a reliable assessment of the effects of architectural design choices, training strategies, and deployment configurations on overall model performance.

Additional details concerning the data sources, class distributions, preprocessing procedures, and the division of the datasets into training, validation, and testing subsets are provided in Section 3.3.2 and are therefore omitted here to avoid redundancy.

4.3.3. Evaluation metrics

This study adopts the same evaluation metrics employed in the TL experimental phase described in Section 3.3.3, thereby ensuring consistency. The classification effectiveness of both the proposed and comparative models is analyzed using conventional metrics typically used in image classification tasks, including accuracy and other measures derived from the confusion matrix.

In addition to classification effectiveness, computational efficiency is systematically evaluated by analyzing the number of trainable parameters, Floating Point Operations (FLOPs), model size, and inference accuracy obtained using TF models. These metrics provide valuable insights into the computational complexity and resource requirements of each model.

Moreover, for deployment in real-time scenarios with limited computational resources, additional metrics are introduced and evaluated using TFLite. These include:

4.3.3.1. Inference time

Inference time is defined as the time a trained model takes to process an input sample and generate a prediction. It is typically measured from the time the input enters the model until the model produces its output. Inference time is a critical performance metric for real-time and edge computing applications, as it directly affects system responsiveness and latency.

4.3.3.2. CPU time

CPU time denotes the amount of time during which the central processing unit actively executes instructions for a given task or process. In the context of DL inference, CPU time measures the computational effort consumed by the processor while performing model inference, excluding idle time and input/output delays. This metric is useful for evaluating computational efficiency and processor utilization, especially on resource-constrained platforms.

4.3.3.3. Memory usage

Memory usage represents the amount of system memory (RAM) required by a model during execution. It includes memory allocated for model parameters, intermediate feature maps, and temporary buffers during inference. Efficient memory usage is essential for deploying DL models on embedded and edge devices, where memory resources are limited.

4.3.3.4. Model size

The size of a trained model refers to the storage space it occupies, typically measured in megabytes (MB). This depends on both the number of parameters and the numerical

precision used to store them (e.g., 32-bit or 16-bit floating point, or 8-bit integer). Model size directly impacts storage requirements, loading time, and deployability on devices with limited storage capacity.

4.3.4. Hardware and software specifications

Experiments were carried out using python 3.11 on a ThinkPad laptop to train and evaluate the lightweight CNN using the DeepFire and Forest Fire Images datasets. The device is equipped with a 2.3 GHz processor, 8 GB of RAM, and operates on Windows 10 Pro. This configuration represents a typical edge-computing environment. Furthermore, the proposed low-complexity CNN model has been deployed on a Raspberry Pi 5 to assess the real-world applicability.

The Raspberry Pi was selected as the main hardware platform because of its versatility and widespread adoption across different domains [104]. This choice is motivated by several factors, including its compact size, efficiency, low power requirements, and cost-effectiveness.The Raspberry Pi has become a popular option for a wide range of IoT applications.

The Raspberry Pi Foundation first introduced the platform in 2012 with the goal of promoting computer science education and providing affordable computing resources for experimentation [105] . Since then, multiple generations have been released, including Raspberry Pi 1 (2012), Pi 2 (2015), Pi 3 (2016), Pi 4 (2019), and the latest Pi 5 (2023). Each generation brought significant enhancements in CPU performance, memory capacity, and connectivity options. The Raspberry Pi 5 represents the most advanced version of the series, providing substantial performance improvements over its predecessors. It is powered by a Broadcom BCM2712 system-on-chip (SoC) featuring a 64-bit quad-core Arm Cortex-A76 CPU running at 2.4 GHz, coupled with a Video Core VII GPU that supports dual 4Kp60 HDMI output. The system is available in 4 GB and 8 GB LPDDR4X RAM variants, enabling it to handle more computationally demanding applications such as computer vision and deep-learning inference [104].

4.4. Results and discussion

This section presents and analyzes the experimental results. First, the proposed low-complexity CNN architecture was evaluated on the DeepFire and Forest Fire Images datasets, and its performance was then compared with leading deep learning methods.

4.4.1. Results

For the DeepFire dataset, the best performance was achieved using the Rmsprop optimizer with a learning rate of 0.001 and a 20%-80% train-validation split. This configuration yielded a test accuracy of 97.37%, a validation accuracy of 98.03%, and a training accuracy of 99.26%. Similarly, for the Forest Fire Images dataset, the highest performance was recorded using the Adam optimizer with a learning rate of 0.0001 and a 90%-10% train-validation split, achieving a test accuracy of 92.00%, a validation accuracy of 88.26%, and a training accuracy of 99.83%. These findings highlight the effectiveness of the proposed approach across both datasets. The achieved results are visualized graphically in a bar chart in Fig.4-3, and the confusion matrices for both datasets are shown in Fig.4-4, along with the ROC curves in Fig.4-5.

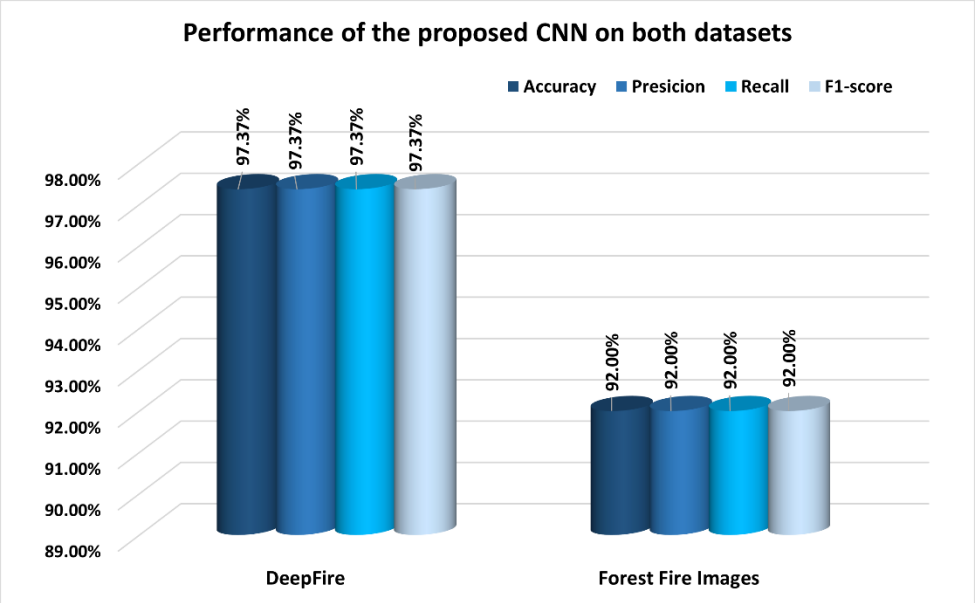
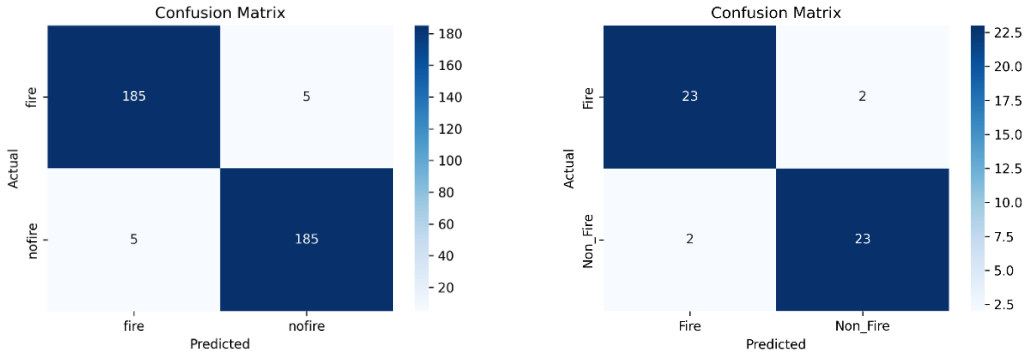


Fig. 4-3.Achieved results for both datasets.

As illustrated in Fig4-3, the proposed architecture demonstrates excellent performance on the DeepFire dataset, achieving an accuracy of 97.37%, with precision, recall, and F1-score all reaching 97.37%. On the second dataset, the model also performs strongly,

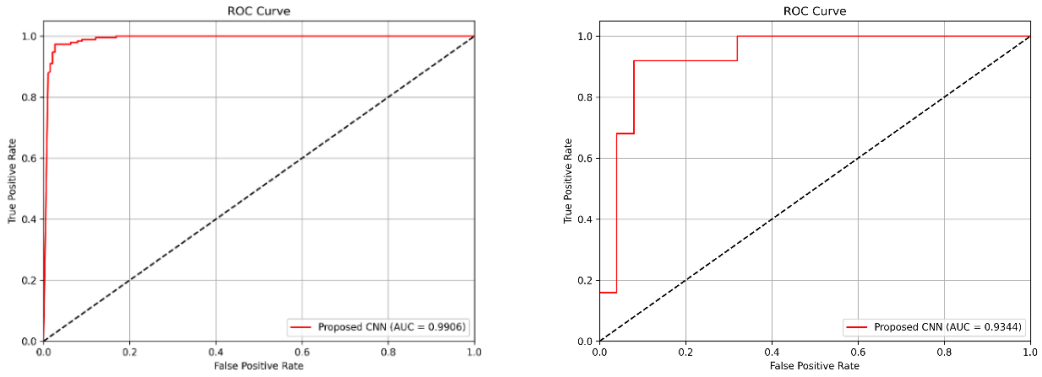
achieving 92.00% accuracy, 92.00% precision, 92.00% recall, and an F1-score of 92.00%. These results emphasize the effectiveness of the proposed model in detecting forest fires with high precision and recall across different datasets.



(a) (b)

Fig. 4-4.The Confusion matrices for both datasets.

As illustrated in Fig.4-4, the confusion matrices demonstrate the proposed model's performance on two datasets. In Fig.4-4 (a), the DeepFire dataset results show 185 correct 'fire' classifications and 185 correct 'non-fire' classifications, with only five 'fire' instances misclassified as 'no-fire' and five 'no-fire' instances misclassified as 'fire'. This indicates high accuracy and a strong ability to distinguish between classes. Similarly, Fig.4-4 (b) for the Forest Fire Images dataset reveals 23 correct 'fire' predictions and 23 correct 'non-fire' predictions, with just two 'fire' misclassification and two 'no-fire' misclassifications, further confirming the model's reliability. Overall, the results highlight the model's promising performance for forest fire detection, particularly given its simple yet efficient architecture.



(a) (b)

Fig. 4-5.The Roc curves for both datasets.

Fig.4-5 (a) illustrates the ROC curve of the proposed low-complexity CNN architecture evaluated on the DeepFire dataset. The model achieved a high Area Under the Curve (AUC) of 0.9906, reflecting its excellent discriminative capability between fire and non-fire classes. The ROC curve ascends rapidly to the top-left corner, demonstrating a strong true positive rate coupled with a minimal false positive rate across multiple threshold settings. This performance confirms that the proposed model is highly effective when applied to the DeepFire dataset, which consists of 1900 images and represents moderate complexity.

Fig.4-5 (b) presents the ROC curve of the same model tested on the Forest Fire dataset, which contains a larger and more diverse set of 4661 images. The model achieved an impressive AUC of 0.9344, demonstrating strong classification performance. The

curve shows a consistently high true positive rate with minimal false alarms, indicating that the model maintains its effectiveness even when applied to a more extensive dataset. This result reinforces the model's suitability for forest fire detection across different data scenarios.

4.4.2. Performance comparison against state-of-the-art techniques

This subsection presents a comparative analysis of our proposed CNN-based fire detection approach against three state-of-the-art DL methods: Khan et al.[78], Islam et al.[89] and wang et al.[90].The comparative results are presented in Table 4-1 for the DeepFire dataset and in Table 4-2 for the Forest Fire Images dataset.

Table 4-1.Performance comparison on DeepFire dataset.

Dataset	Models	Layers			Number of parameters	Acc %	F1-score %
		ConvL	FCL	Depth			
DeepFire	Proposed approach	3	1	4	110562	97.37	97.37
	Reduce-VGGNet [90]	13	2	15	40,400,000	91.20	97.22
	VGG19 [78]	16	3	19	143,667,240	95.00	94.96
	Efficientnetb7+ ACNet [90]	55	1	56	+66,000,000	95.97	95.54

Table 4-1 presents a performance comparison of various DL models on the DeepFire dataset, evaluating architectural complexity including the number of layers, parameters, and depth (convolutional and fully connected layers) in relation to their accuracy and F1-score. The proposed approach, which consists of only three convolutional layers and one fully connected layer, resulting in a total depth of four, achieved outstanding performance, reaching 97.37% in both accuracy and F1-score, while maintaining a lightweight architecture with only 110,562 parameters. In contrast, more complex architectures such as VGG19 and EfficientNetB7 enhanced with ACNet, which comprise millions of parameters and depths of 19 and 56 layers respectively, demonstrated lower performance. Specifically, the proposed model's depth is approximately 75% smaller than that of VGG19, and 92.86% smaller than EfficientNetB7+ACNet. Despite their considerable complexity, these deeper networks achieved lower accuracies: VGG19 reached 95.00%, and EfficientNetB7+ACNet achieved 95.97%, yet both required significantly longer training times and still did not match the proposed model's performance.These results reinforce the efficiency and effectiveness of the lightweight CNN architecture, confirming that well-designed low-complexity models can deliver superior outcomes, even when compared to deeper and more resource-intensive alternatives.

Table 4-2.Performance comparison on Forest Fire Images dataset.

Dataset	Models	Layers			Number of parameters	Acc %	F1-score %
		ConvL	FCL	Depth			
Forest Fire Images	Proposed approach	3	1	4	110562	92.00	92.00
	InceptionV3	48	1	49	23,851,784	83.23	85.29
	Resnet50V2	49	1	50	25,613,800	84.89	86.41
	NASNet mobile	88	1	89	5,289,978	85.09	86.95
	InceptionResnetV2	80	1	81	55,873,736	55.19	71.12

A consistent pattern was observed on the Forest Fire Images dataset (Table 4-2), where the proposed architecture maintained strong performance, achieving both an accuracy and an F1-score of 92.00%. The model continued to outperform more complex and deeper networks, such as InceptionV3, ResNet50V2, NASNet Mobile, and InceptionResNetV2, whose depths range from 49 to 89 layers. It is important to note that all comparative results reported in Table 4-2 are sourced from Yandouzi et al. [88]. Remarkably, the proposed model is approximately 91.84% to 95.51% shallower than these architectures. Despite their increased complexity, these deeper models delivered inferior results: InceptionV3 and ResNet50V2 achieved accuracies of 83.23% and 84.89%, respectively, while NASNet Mobile and InceptionResNetV2 recorded 85.09% and 55.19%. These findings highlight the effectiveness of the proposed low-complexity CNN, which not only surpasses heavier pretrained networks in performance but also offers significant advantages in computational efficiency, making it particularly suitable for deployment in resource-constrained environments.

4.5. Deployment of the proposed models on Raspberry Pi 5

To test the real-world performance of the proposed CNN model, the model was implemented on a Raspberry Pi 5 as illustrated in Fig.4-6. This deployment aimed to evaluate the model's efficiency in terms of inference speed, memory consumption, and overall performance under practical operating conditions, particularly in environments with limited

computational resources. Deploying the model on such a platform is essential for verifying its suitability for real-time forest fire detection in remote or inaccessible areas.

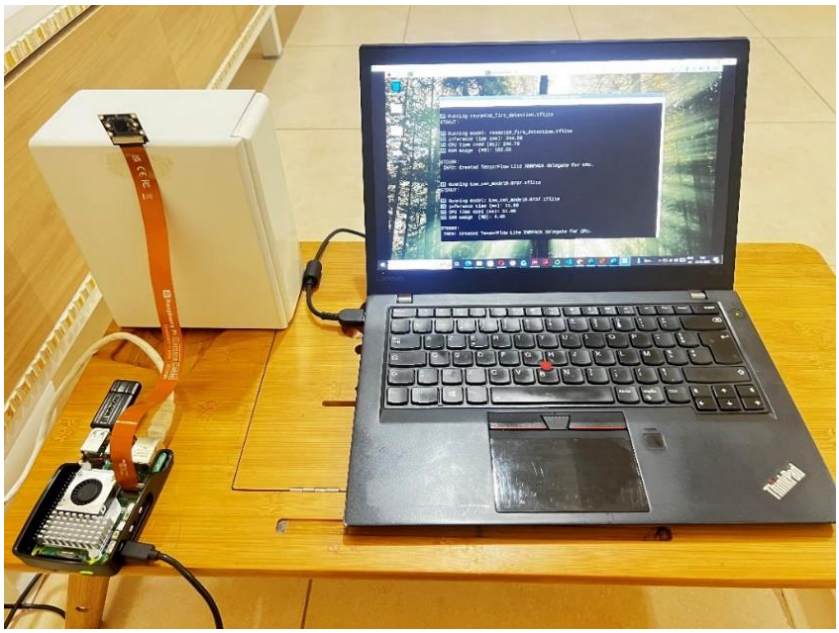


Fig. 4-6.Experimental setup of the forest fire detection system using Raspberry Pi 5.

The implementation process involved converting the trained model into TFLite format to reduce its size and optimize it for edge deployment. This conversion enables faster inference and lower memory consumption, both of which are critical for achieving real-time detection on resource-constrained platforms such as the Raspberry Pi. In addition to the baseline implementation, several DL enhancement techniques were investigated to further reduce the computational complexity of the proposed CNN-based solution while maintaining or improving its performance. These techniques, commonly employed to improve generalization and mitigate overfitting, include Dropout, L2 Regularization, Squeeze-and-Excitation (SE) blocks, and Global Average Pooling (GAP). Each technique was integrated individually into the baseline model, and its impact was evaluated under identical experimental conditions. The objective was to determine whether these enhancements could improve model performance and reduce computational complexity relative to the original, minimalistic model. Additionally, two pre-trained networks were included: a heavy one (ResNet50) and a lightweight one (MobileNetV1).

Table 4-3 presents a comprehensive comparison of several CNN architectures, including pretrained models (MobileNetV1 and ResNet50) and proposed lightweight CNN variants, evaluated on the DeepFire and Forest Fire datasets. The comparison considers multiple performance and efficiency metrics, including FLOPS, number of parameters, accuracy, inference and CPU time, memory usage, and model size in TF and TFLite formats. As shown in Table 4-3, pre-trained models such as ResNet50 achieved the highest accuracy, reaching 99.21% and 100% on the DeepFire and Forest Fire Images datasets, respectively. Despite the excellent performance, it comes with the cost of very high computational complexity (7,755 MFLOPS), a large number of parameters (23.6 million, corresponding to a model size of approximately 92,575 KB and 91,777 KB for TF and TFLite files, respectively), memory usage of 182.56 MB, and a long inference time of 244.56 ms.MobileNetV1, on the other hand, provides a good balance between accuracy and efficiency, achieving 98.68% accuracy on the DeepFire dataset and 98% on the Forest Fire Images dataset, with relatively low computational complexity (1147.69MFLOPS), a shorter inference time of 36.85 ms, and a moderate model size of 12,829 KB and 12,512KB for TF and TFLite files, respectively.In contrast, the proposed lightweight CNN variants demonstrate outstanding computational efficiency, with FLOPS ranging from 83.54 to 280.86 MFLOPS, model sizes below 1,400 KB and 500 KB for TF and TFLite files, respectively, and a reduction in the number of parameters from millions to 110,562. Despite this drastic reduction in complexity, thesemodels maintain competitive accuracy levels, ranging between 95.53% and 97.37% on the DeepFire dataset and between 90% and 96% on the Forest Fire Images dataset. Among these models, the baseline low-complexity CNN achieved the best overall trade-off, with an accuracy of 97.37% on the DeepFire dataset, an inference time of 11.06 ms, memory usage of 4.45 MB, and a compact model size of only 436.6 KB. Meanwhile, the CNN+GAP variant achieved the highest accuracy on the Forest Fire Images dataset (96%), while reducing the number of parameters from 110,562 in the baseline model (itself a reduction from millions in the pre-trained models) to just 24,162. It also achieved an inference time of 10.96 ms, memory usage of 2.80 MB, and a compact model size of 100 KB for the TFLite file. This highlights the effectiveness of global average pooling in improving efficiency without sacrificing performance.

Table 4-3. Performance evaluation and comparison of CNN models on Raspberry Pi 5 using TF and TFLite.

Model	TF					TFLite					
	FLOPS (MFLOPS)	number of Parameters	Model size (KB)	Accuracy %		Inference Time (ms)	CPU Time (ms)	Memory usage (MB)	Model size (KB)	Accuracy%	
				Deep fire	Forest fire					Deep fire	Forest fire
Resnet50	7755.3	23591810	92575	99.21	100	244.32	244.30	182.58	91777	99.21	100
Mobilenet_v1	1147.69	3230914	12829	98.68	98	36.95	37.10	27.12	12512	98.68	98
Basic CNN model	264.47	110562	1363	97.37	92	11.06	11.00	4.45	437	97.37	92
CNN + dropout	280.86	123938	1499	95.79	92	10.20	10.20	4.64	488	95.79	92
CNN + GAP	264.34	24162	353	96.84	96	10.96	11.10	2.80	100	96.84	96
CNN + L2	280.86	123944	1504	95.53	92	10.18	10.30	4.64	489	95	92
CNN+ SE	83.54	56609	757	96.31	90	9.05	25.90	2.31	232	96.32	90

4.6. Conclusion

This chapter presented the proposed low-complexity CNN for forest fire detection. With three convolutional layers and a single fully connected layer, the model achieved high accuracy while maintaining minimal computational requirements.

The evaluation across two wildfire datasets showed strong performance, low inference time, small memory footprint, and compact model size, making it suitable for deployment on resource-constrained platforms like the Raspberry Pi.

Overall, this study demonstrates that lightweight, task-specific CNNs can provide reliable and efficient solutions for real-time forest fire detection in embedded and edge environments.

5. Conclusion and future work

5.1. Conclusion

This research presents a comprehensive investigation into the application of convolutional neural networks for forest fire detection. The study was structured around three main phases that focused on the evaluation of pre-trained DL models, the design of a low-complexity convolutional neural network, and the practical deployment of the developed systems on embedded hardware. The overall goal was to enhance the efficiency, accuracy, and applicability of DL methods for real-time fire detection in environments with limited computational resources.

In the first phase, six advanced pre-trained convolutional neural network architectures, namely AlexNet, ResNet18, SqueezeNet, GoogLeNet, MobileNetV2, and EfficientNetB0, were systematically evaluated using two benchmark datasets: the DeepFire dataset and the Forest Fire Images dataset. Through TL, these models were fine-tuned and analyzed under various hyperparameter configurations, including learning rate, batch size, and optimizer type. The results provided valuable insights into the performance and behavior of each network, highlighting the trade-off between accuracy and computational cost.

In the second phase, a custom lightweight convolutional neural network was developed specifically for forest fire detection. The proposed model was designed to achieve simplicity, efficiency, and high accuracy while reducing computational requirements. Despite its compact structure, the model achieved strong classification results that surpassed several of the larger pre-trained architectures, while also requiring fewer parameters, less memory, and shorter inference time.

In the third phase, several well-known enhancement techniques were integrated into the proposed model. These included Global Average Pooling, Dropout, Squeeze and Excitation blocks, and L2 regularization. Each technique was examined to assess its effect on model generalization and robustness. Furthermore, the proposed network in its original and enhanced versions, along with selected pre-trained networks, was implemented on a Raspberry Pi 5 device. This deployment provided a practical validation of the models,

confirming that the proposed lightweight architecture maintained strong accuracy and rapid processing even under limited hardware capabilities.

In summary, this study demonstrates the feasibility of designing CNNs that are efficient, accurate, and suitable for real-world implementation. The results establish a solid foundation for the creation of intelligent, scalable, and real-time fire detection systems that can be integrated into drones, sensor networks, or Internet of Things platforms to support early detection and fast response to wildfire events.

5.2. Future directions

Based on the outcomes of this research, several promising directions can be pursued in future work.

Dataset expansion and multimodal learning: Future research should include larger and more diverse datasets that represent a wider variety of forest conditions, weather scenarios, and fire types. The inclusion of additional data modalities, such as infrared, thermal, and multispectral imagery, would improve model robustness and allow detection under challenging conditions such as nighttime or heavy smoke.

Hybrid and ensemble learning models: Combining CNNs with other modern DL architectures, such as Vision Transformers, or using ensemble approaches that merge predictions from multiple models, can improve detection accuracy and reduce false alarms. These approaches can enhance the model's ability to generalize across complex and dynamic visual environments.

Real-Time and field implementation: Expanding the current framework to process live video feeds instead of static images is an important next step. Future efforts should also focus on deploying the system in real-world conditions using drones, unmanned aerial vehicles, or distributed sensor networks to test reliability, latency, and energy efficiency under dynamic environmental conditions.

References

[1] G. Zanchi *et al.*, “Simulation of water and chemical transport of chloride from the forest ecosystem to the stream,” *Environ. Model. Softw.*, vol. 138, p. 104984, 2021, doi: 10.1016/j.envsoft.2021.104984.

[2] M. Bo, L. Mercalli, F. Pognant, D. C. Berro, and M. Clerico., “Urban air pollution, climate change and wildfires: The case study of an extended forest fire episode in northern Italy favoured by drought and warm weather conditions,” *Energy reports*, vol. 6, pp. 781–786, 2020, doi: https://doi.org/10.1016/j.egyr.2019.11.002.

[3] A. A. A. Alkhatib, “A review on forest fire detection techniques,” *Int. J. Distrib. Sens. Networks*, vol. 2014, 2014, doi: 10.1155/2014/597368.

[4] P. Pereira, I. Bogunovic, W. Zhao, and D. Barcelo, “Short-term effect of wildfires and prescribed fires on ecosystem services,” *Curr. Opin. Environ. Sci. \& Heal.*, vol. 22, p. 100266, 2021, doi: 10.1016/j.coesh.2021.100266.

[5] M. D. Carmona-Yáñez *et al.*, “Short-term impacts of wildfire and post-fire mulching on ecosystem multifunctionality in a semi-arid pine forest,” *For. Ecol. Manage.*, vol. 541, no. April, 2023, doi: 10.1016/j.foreco.2023.121000.

[6] E. Grant and J. D. Runkle, “Long-term health effects of wildfire exposure: A scoping review,” *J. Clim. Chang. Heal.*, vol. 6, p. 100110, 2022, doi: 10.1016/j.joclim.2021.100110.

[7] C. X. Cunningham, G. J. Williamson, and D. M. Bowman, “Increasing frequency and intensity of the most extreme wildfires on Earth,” *Nat. Ecol. \& Evol.*, vol. 8, no. 8, pp. 1420--1425, 2024, doi: 10.1038/s41559-024-02452-2.

[8] S. Vardoulakis, G. Marks, and M. J. Abramson, “Lessons learned from the Australian bushfires: climate change, air pollution, and public health,” *JAMA Intern. Med.*, vol. 180, no. 5, pp. 635--636, 2020, doi: 10.1001/jamainternmed.2020.0703.

[9] Statista, “Area burned by forest and wildland fires in the Mediterranean basin as of November 2024, with average for 2006 to 2023, by country.” [Online]. Available:

https://www.statista.com/statistics/1412691/area-burned-by-wildfire-in-mediterranean-countries/utm.com

[10] France24., “At least 42 people killed in Algeria wildfires, president says,” france24. [Online]. Available: https://www.france24.com/en/africa/20210810-algeria-wildfires-kill-dozens-in-latest-country-struck-by-blazes?utm.com

[11] C. Eberle and O. H. Roa, “Technical Report: Mediterranean wildfires - Interconnected Disaster Risks 2021/2022,” 2021.

[12] BBC, “Algeria forest fires: At least 38 dead, emergency officials say,” BBC. Accessed: Apr. 28, 2025. [Online]. Available: https://www.bbc.com/news/world-africa-62581108?utm.com

[13] Aljazeera, “Wildfires in Algeria kill dozens, force hundreds to flee homes,” aljazeera. Accessed: Apr. 28, 2025. [Online]. Available: https://www.aljazeera.com/news/2023/7/24/deadly-algeria-wildfires-amid-extreme-heat-high-winds

[14] Copernicus, “EFFIS Annual Statistics for Algeria,” copernicus. Accessed: Apr. 29, 2025. [Online]. Available: https://forest-fire.emergency.copernicus.eu/apps/effis.statistics/estimates/DZA

[15] V. Vipin, “Image Processing Based Forest Fire Detection,” *Int. J. Emerg. Technol. Adv. Eng.*, vol. 2, no. 2, pp. 87–95, 2012.

[16] G. Liu, H. Yuan, and L. Huang, “A fire alarm judgment method using multiple smoke alarms based on Bayesian estimation,” *Fire Saf. J.*, vol. 136, p. 103733, 2023, doi: 10.1016/j.firesaf.2023.103733.

[17] A. Nettis, V. Massimi, R. Nutricato, D. O. Nitti, S. Samarelli, and G. Uva, “Satellite-based interferometry for monitoring structural deformations of bridge portfolios,” *Autom. Constr.*, vol. 147, p. 104707, 2023, doi: 10.1016/j.autcon.2022.104707.

[18] D. Kaliyev, O. Shvets, and G. Györök, “Computer Vision-based Fire Detection using Enhanced Chromatic Segmentation and Optical Flow Model,” *Acta Polytech. Hungarica*, vol. 20, no. 6, pp. 27–45, 2023, doi:

10.12700/APH.20.6.2023.6.2.

[19] Y. Zhang *et al.*, “Visual search at alibaba,” *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 993–1001, 2018, doi: 10.1145/3219819.3219820.

[20] J. Liu, Y. Liu, D. Li, H. Wang, X. Huang, and L. Song, “DSDCLA: Driving style detection via hybrid CNN-LSTM with multi-level attention fusion,” *Appl. Intell.*, vol. 53, no. 16, pp. 19237--19254, 2023, doi: 10.1007/s10489-023-04451-5.

[21] N. N. Prakash, V. Rajesh, D. L. Namakhwa, S. D. Pande, and S. H. Ahammad, “A DenseNet CNN-based liver lesion prediction and classification for future medical diagnosis,” *Sci. African*, vol. 20, p. e01629, 2023, doi: https://doi.org/10.1016/j.sciaf.2023.e01629.

[22] L. Zhang, M. Wang, Y. Fu, and Y. Ding, “A Forest Fire Recognition Method Using UAV Images Based on Transfer Learning,” *Forests*, vol. 13, no. 7, 2022, doi: 10.3390/f13070975.

[23] H. P. Gupta and R. Mishra, “Utilizing Transfer Learning and pre-trained Models for Effective Forest Fire Detection: A Case Study of Uttarakhand,” 2024, [Online]. Available: http://arxiv.org/abs/2410.06743

[24] M. Boukabous and M. Azizi, “Crime prediction using a hybrid sentiment analysis approach based on the bidirectional encoder representations from transformers,” *Indones. J. Electr. Eng. Comput. Sci.*, vol. 25, no. 2, pp. 1131–1139, 2022, doi: 10.11591/ijeecs.v25.i2.pp1131-1139.

[25] I. Idrissi, M. Azizi, and O. Moussaoui, “IoT security with Deep Learning-based Intrusion Detection Systems: A systematic literature review,” *4th Int. Conf. Intell. Comput. Data Sci. ICDS 2020*, 2020, doi: 10.1109/ICDS50568.2020.9268713.

[26] I. Idrissi, M. Boukabous, M. Azizi, O. Moussaoui, and H. El Fadili, “Toward a deep learning-based intrusion detection system for iot against botnet attacks,” *IAES Int. J. Artif. Intell.*, vol. 10, no. 1, pp. 110–120, 2021, doi: 10.11591/ijai.v10.i1.pp110-120.

[27] D. Cornelisse, “An intuitive guide to Convolutional Neural Networks.” Accessed: May 12, 2025. [Online]. Available: https://www.freecodecamp.org/news/an-

intuitive-guide-to-convolutional-neural-networks-260c2de0a050

[28] M. D. Zeiler, “Hierarchical Convolutional Deep Learning in Computer Vision,” New York University, 2014.

[29] C. Szegedy *et al.*, “Going Deeper with Convolutions Christian,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[30] M. Oquab and Bottou, “Is object localization for free? – Weakly-supervised learning with convolutional neural networks Maxime,” *Openaccess.Thecvf.Com*, no. iii, 2015, [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2015/html/Oquab_Is_Object_Localization_2015_CVPR_paper.html

[31] M. Lin, Q. Chen, and S. Yan, “Network in network,” *2nd Int. Conf. Learn. Represent. ICLR 2014 - Conf. Track Proc.*, pp. 1–10, 2014.

[32] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, “Deep learning for visual understanding: A review,” *Neurocomputing*, vol. 187, pp. 27–48, 2016, doi: 10.1016/j.neucom.2015.09.116.

[33] C. Szegedy and S. G. Com, “Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift,” vol. 37, 2015.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Adv. Neural Inf. Process. Syst. (NIPS 2012)*, vol. 25, 2012, doi: https://doi.org/10.1145/3065386.

[35] Y. Boureau, J. Ponce, J. P. Fr, and Y. Lecun, “A Theoretical Analysis of Feature Pooling in Visual Recognition Y-Lan,” *Icml*, pp. 111--118, 2010, [Online]. Available: https://www.di.ens.fr/sierra/pdfs/icml2010b.pdf

[36] V. Suárez-Paniagua and I. Segura-Bedmar, “Evaluation of pooling operations in convolutional architectures for drug-drug interaction extraction,” *BMC Bioinformatics*, vol. 19, pp. 92–101, 2018, doi: 10.1186/s12859-018-2195-1.

[37] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, “High-Performance Neural Networks for Visual Object Classification,” 2011, [Online].

Available: http://arxiv.org/abs/1102.0183

[38] M. D. Zeiler and R. Fergus, “Stochastic pooling for regularization of deep convolutional neural networks,” *1st Int. Conf. Learn. Represent. ICLR 2013 - Conf. Track Proc.*, pp. 1–9, 2013, doi: doi.org/10.48550/arXiv.1301.3557.

[39] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition.,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, 2015.

[40] W. Ouyang *et al.*, “DeepID-Net: Deformable deep convolutional neural networks for object detection,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, pp. 2403–2412, 2015, doi: 10.1109/CVPR.2015.7298854.

[41] J. Jin, A. Dundar, E. Culurciello, and W. Lafayette, “FLATTENED CONVOLUTIONAL NEURAL NETWORKS FOR FEEDFORWARD ACCELERATION,” no. 2014, pp. 1–11, 2015.

[42] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 580–587, 2014, doi: 10.1109/CVPR.2014.81.

[43] A. A. de A. Madeira, “INTELLIGENT SYSTEM FOR FIRE DETECTION,” universidade d coimbra, 2020.

[44] B. SAVDA, “OPTIMIZATION OF DEEP NEURAL NETWORK ARCHITECTURES FOR THE FOREST FIRE DETECTION,” 2023.

[45] G. Du, C. Tian, Z. Li, D. Zhang, Y. Yin, and Y. Ouyang, “Efficient Softmax Hardware Architecture for Deep Neural Networks,” pp. 75–80, 2019.

[46] A. H. Ali and M. G. Yaseen, “Transfer Learning : A New Promising Techniques,” 2023.

[47] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks ?,” pp. 1–9.

[48] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," pp. 1–13, 2016, [Online]. Available: http://arxiv.org/abs/1602.07360

[49] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[50] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017, [Online]. Available: http://arxiv.org/abs/1704.04861

[51] Q. V. Le Mingxing Tan, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks Mingxing," *Can. J. Emerg. Med.*, vol. 15, no. 3, p. 190, 2019.

[52] T. Tieleman, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA Neural networks Mach. Learn.*, vol. 4, no. 2, p. 26, 2012.

[53] S. Ruder, "An overview of gradient descent optimization algorithms," pp. 1–14, 2016, [Online]. Available: http://arxiv.org/abs/1609.04747

[54] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, 2015.

[55] Chinwe I., Anyama O., Uzoma A. U., and Abasiama S., "Effect of Learning Rate on Artificial Neural Network in Machine Learning," *Int. J. Eng. Res.* , vol. 4, no. 2, pp. 359–363, 2021.

[56] P. M. Radiuk, "Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets," *Inf. Technol. Manag. Sci.*, vol. 20, no. 1, pp. 20–24, 2018, doi: 10.1515/itms-2017-0003.

[57] O. Elharrouss *et al.*, "Loss Functions in Deep Learning: A Comprehensive Review," 2025, [Online]. Available: http://arxiv.org/abs/2504.04242

[58] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[59] A. Y. Ng, "Feature selection, L 1 vs. L 2 regularization, and rotational invariance," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 78.

[60] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 2, pp. 39–43, 2018, doi: 10.1109/QCE60285.2024.10249.

[61] M. Chetoui and M. A. Akhloufi., "Fire and Smoke Detection Using Fine-Tuned YOLOv8 and YOLOv7 Deep Models," *Fire*, vol. 7, no. 4, p. 135, 2024, doi: https://doi.org/10.3390/fire7040135.

[62] R. N. Vasconcelos *et al.*, "Fire Detection with Deep Learning: A Comprehensive Review," *Land*, vol. 13, no. 10, 2024, doi: 10.3390/land13101696.

[63] B. Özel, S. M. Alam, and M. U. Khan, "Review of Modern Forest Fire Detection Techniques: Innovations in Image Processing and Deep Learning," *Information*, vol. 15, no. 9, p. 538, 2024, doi: https://doi.org/10.3390/info15090538.

[64] G. Guangmeng and Z. Mei, "Using MODIS land surface temperature to evaluate forest fire risk of Northeast China," *IEEE Geosci. Remote Sens. Lett.*, vol. 1, no. 2, pp. 98–100, 2004, doi: 10.1109/LGRS.2004.826550.

[65] Z. Li, J. Cihlar, and S. Nadon, "Satellite-based detection of Canadian Boreal forest fires: Development and application of the algorithm," *Int. J. Remote Sens.*, vol. 21, no. 16, pp. 3057–3069, 2000, doi: 10.1080/01431160050144956.

[66] K. Nakau, M. Fukuda, and K. Kushida, "Forest fire detection based on MODIS satellite imager y , and Comparison of NOA A satellite imagery with fire fighters ' information. IARC/JAXA Terrestrial Team Workshop.," no. February, pp. 18–23, 2006.

[67] B. Ko, "Wildfire smoke detection using temporospatial features and random forest classifiers," *Opt. Eng.*, vol. 51, no. 1, p. 017208, 2012, doi:

10.1117/1.oe.51.1.017208.

[68] L. Ma, K. Wu, and L. Zhu, “Fire smoke detection in video images using Kalman filter and Gaussian mixture color model,” *Proc. - Int. Conf. Artif. Intell. Comput. Intell. AICI 2010*, vol. 1, no. 1, pp. 484–487, 2010, doi: 10.1109/AICI.2010.107.

[69] M. Kandil and M. Salama, “A NEW HYBRID ALGORITHM FOR FIRE VISION RECOGNITION,” *IEEE EUROCON 2009*, pp. 1460--1466, 2009, doi: 10.1109/EURCON.2009.5167833.

[70] T. H. Chen, P. H. Wu, and Y. C. Chiou, “An early fire-detection method based on image processing,” *Proc. - Int. Conf. Image Process. ICIP*, vol. 3, pp. 1707–1710, 2004, doi: 10.1109/ICIP.2004.1421401.

[71] T. Çelik and H. Demirel, “Fire detection in video sequences using a generic color model,” *Fire Saf. J.*, vol. 44, no. 2, pp. 147–158, 2009, doi: 10.1016/j.firesaf.2008.05.005.

[72] W. Homg, J. Peng, and C. Chen, “A New Image-Based Real-Time Flame Detection,” *IEEE Int. Conf. Netw.*, pp. 1–6, 2005.

[73] G. Marbach, M. Loepfe, and T. Brupbacher, “An image processing technique for fire detection in video images,” *Fire Saf. J.*, vol. 41, no. 4, pp. 285–289, 2006, doi: 10.1016/j.firesaf.2006.02.001.

[74] B. Uğur Töreyin, Y. Dedeoğlu, and A. Enis Çetin, “Wavelet based real-time smoke detection in video,” *13th Eur. Signal Process. Conf. EUSIPCO 2005*, pp. 293–296, 2005.

[75] S. Ye, Z. Bai, H. Chen, R. Bohush, and S. Ablameyko, “An effective algorithm to detect both smoke and flame using color and wavelet analysis,” *Pattern Recognit. Image Anal.*, vol. 27, no. 1, pp. 131–138, 2017, doi: 10.1134/S1054661817010138.

[76] B. Lee and D. Han, “Real-time fire detection using camera sequence image in tunnel environment,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4681 LNCS, pp. 1209–1220, 2007, doi: 10.1007/978-3-540-74171-8_123.

[77] P. Foggia, A. Saggese, and M. Vento, “Real-time fire detection for video-surveillance applications using a combination of experts based on color, shape, and motion,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 9, pp. 1545–1556, 2015, doi: 10.1109/TCSVT.2015.2392531.

[78] A. Khan, B. Hassan, S. Khan, R. Ahmed, and A. Abuassba, “DeepFire: A Novel Dataset and Deep Transfer Learning Benchmark for Forest Fire Detection,” *Mob. Inf. Syst.*, vol. 2022, no. 1, p. 14, 2022, doi: 10.1155/2022/5358359.

[79] Spoorthy M R and Hemanth Kumar, “Detection of Forest Fire Areas using Machine Learning,” *Int. J. Adv. Res. Sci. Commun. Technol.*, no. June 2022, pp. 699–704, 2022, doi: 10.48175/ijarsct-5623.

[80] K. Muhammad, J. Ahmad, and S. W. Baik, “Early fire detection using convolutional neural networks during surveillance for effective disaster management,” *Neurocomputing*, vol. 288, pp. 30–42, 2018, doi: 10.1016/j.neucom.2017.04.083.

[81] K. Govil, M. L. Welch, J. T. Ball, and C. R. Pennypacker, “Preliminary results from a wildfire detection system using deep learning on remote camera images,” *Remote Sens.*, vol. 12, no. 1, 2020, doi: 10.3390/RS12010166.

[82] Y. Tang, H. Feng, J. Chen, and Y. Chen, “ForestResNet: A deep learning algorithm for forest image classification,” *J. Phys. Conf. Ser.*, vol. 2024, no. 1, 2021, doi: 10.1088/1742-6596/2024/1/012053.

[83] X. Sun, L. Sun, and Y. Huang, “Forest fire smoke recognition based on convolutional neural network,” *J. For. Res.*, vol. 32, no. 5, pp. 1921–1927, 2021, doi: 10.1007/s11676-020-01230-7.

[84] A. Shamsoshoara, F. Afghah, A. Razi, L. Zheng, P. Z. Fulé, and E. Blasch, “Aerial imagery pile burn detection using deep learning: The FLAME dataset,” *Comput. Networks*, vol. 193, p. 108001, 2021, doi: 10.1016/j.comnet.2021.108001.

[85] A. Namburu, P. Selvaraj, S. Mohan, S. Ragavanantham, and E. T. Eldin, “Forest Fire Identification in UAV Imagery Using X-MobileNet,” *Electron.*, vol. 12, no. 3, pp. 1–19, 2023, doi: 10.3390/electronics12030733.

[86] S. Treneska and B. R. Stojkoska, “Wildfire detection from UAV collected images using transfer learning,” *Conf. 18th Int. Conf. Informatics Inf. Technol.*, no. August, 2021, [Online]. Available: https://www.researchgate.net/publication/353732371

[87] M. J. Sousa, A. Moutinho, and M. Almeida, “Wildfire detection using transfer learning on augmented datasets,” *Expert Syst. Appl.*, vol. 142, 2020, doi: 10.1016/j.eswa.2019.112975.

[88] M. Yandouzi *et al.*, “Forest Fires Detection using Deep Transfer Learning,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 8, pp. 268–275, 2022, doi: 10.14569/IJACSA.2022.0130832.

[89] A. M. Islam *et al.*, “An Attention-Guided Deep-Learning-Based Network with Bayesian Optimization for Forest Fire Classification and Localization,” *Forests*, vol. 14, no. 10, pp. 1–24, 2023, doi: 10.3390/f14102080.

[90] L. Wang, H. Zhang, Y. Zhang, K. Hu, and K. An, “A Deep Learning-Based Experiment on Forest Wildfire Detection in Machine Vision Course,” *IEEE Access*, vol. 11, no. March, pp. 32671–32681, 2023, doi: 10.1109/ACCESS.2023.3262701.

[91] S. Khan and A. Khan, “FFireNet: Deep learning based forest fire classification and detection in smart cities,” *Symmetry (Basel).*, vol. 14, no. 10, p. 2155, 2022, doi: https://doi.org/10.3390/sym14102155.

[92] M. Prakash, S. Neelakandan, M. Tamilselvi, S. Velmurugan, S. . Baghavathi Priya, and E. Ofori Martinson, “Deep Learning-Based Wildfire Image Detection and Classification Systems for Controlling Biomass,” *Int. J. Intell. Syst.*, vol. 2023, p. 18, 2023, doi: 10.1155/2023/7939516.

[93] R. Ghali, M. A. Akhloufi, and W. S. Mseddi, “Deep Learning and Transformer Approaches for UAV-Based Wildfire Detection and Segmentation,” *Sensors*, vol. 22, no. 5, pp. 1–18, 2022, doi: 10.3390/s22051977.

[94] “Forest Fire Images,” kaggle. Accessed: Aug. 05, 2024. [Online]. Available: https://www.kaggle.com/datasets/mohnishsaiprasad/forest-fire-images

[95] M. Hossin and M. N. Sulaiman, “A REVIEW ON EVALUATION METRICS FOR DATA CLASSIFICATION EVALUATIONS,” *Int. J. Data Min. Knowl. Manag. Process*, vol. 5, no. 2, p. 102478, 2015.

[96] I. Idrissi, M. Azizi, and O. Moussaoui, “Accelerating the update of a DL-based IDS for IoT using deep transfer learning,” *Indones. J. Electr. Eng. Comput. Sci.*, vol. 23, no. 2, pp. 1059–1067, 2021, doi: 10.11591/ijeecs.v23.i2.pp1059-1067.

[97] S. B. KUKUK and Z. H. KİLİMCİ, “Comprehensive Analysis of Forest Fire Detection using Deep Learning Models and Conventional Machine Learning Algorithms,” *Int. J. Comput. Exp. Sci. Eng.*, vol. 7, no. 2, pp. 84–94, 2021, doi: 10.22399/ijcesen.950045.

[98] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, 2006, doi: 10.1016/j.patrec.2005.10.010.

[99] J. A. Hanley and B. J. McNeil, “The meaning and use of the area under a receiver operating characteristic (ROC) curve,” *Radiology*, vol. 143, no. 1, pp. 29–36, 1982, doi: 10.1148/radiology.143.1.7063747.

[100] T. Saito and M. Rehmsmeier, “The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets,” *PLoS One*, vol. 10, no. 3, pp. 1–21, 2015, doi: 10.1371/journal.pone.0118432.

[101] M. Geiger *et al.*, “Scaling description of generalization with number of parameters in deep learning,” *J. Stat. Mech. Theory Exp.*, vol. 2020, no. 2, p. 023401, 2020, doi: 10.1088/1742-5468/2011/07/L07002.

[102] J. E. Moody, “An Analysis of Generalization and Regularization in Nonlinear Learning Systems,” *Neural Inf. Process. Syst.*, no. 1, pp. 847–854, 1992.

[103] J. Johnson, “Rethinking floating point for deep learning,” 2018, [Online]. Available: http://arxiv.org/abs/1811.01721

[104] A. L. Zekovic, “Hardware and Software of Computer Vision IoT Solutions Leveraging Raspberry Pi Boards,” vol. 17, no. 1, pp. 2–7, 2025.

[105] N. Kandasamy,Gajendiran Sabariswaran, Kandasamy Mathiyazhagan, “Influences

of wildfire on the forest ecosystem and climate change: A comprehensive study," *Environ. Res.*, vol. 240, p. 117537, 2024, doi: https://doi.org/10.1016/j.envres.2023.117537.