Forest Fire Detection Using Machine Learning

[1]Priyanshu Yadav, [2]V.Sathiyasuntharam, [3]Ram Sahay yadav, [4]Mashhad Ali
[1]Student,[2]Professor,[3]Student,[4]Student
[1] Computer Science and Engineering
[1]Sharda University, Greater Noida, India
[1]priyanshuyadav670455@gmail.com, [2]sathiya4196@gmail.com, [3]ramsahayyadav340@gmail.com, [4]mashhadali690@gmail.com

ABSTRACT :
Forest fires are increasingly frequent and destructive, demanding faster and more reliable detection methods to minimize environmental and economic damage. This research presents a machine learning-based approach for early forest fire detection using image classification techniques. The system is developed using Convolutional Neural Networks (CNNs), trained on a labelled dataset comprising images of forested regions with and without fire incidents. The data is pre-processed using techniques such as rescaling and resizing to a uniform dimension of 150×150 pixels, and organized into training, validation, and test sets through image data generators. Visualization and augmentation techniques are applied to enhance dataset variability and improve model generalization. A binary classification model is trained to distinguish between "fire" and "no fire" images, with performance evaluated using accuracy, precision, and recall metrics. This image-based detection method allows the system to process real-time inputs from drones, satellites, or surveillance cameras, enabling early alert mechanisms. The proposed framework demonstrates the potential of deep learning in building efficient, scalable, and automated forest fire detection systems with practical applicability in disaster management and environmental conservation.

Index Terms— more reliable detection method , machine learning-based approach , Convolutional Neural Networks (CNNs), labelled dataset comprising images , real-time inputs from drones, satellites, or surveillance cameras, of deep learning in building efficient , automated forest fire detection systems.

INTRODUCTION :

Forest fires, also referred to as wildfires or bushfires, have emerged as one of the most critical environmental challenges of the 21st century. Over the past few decades, their frequency and intensity have increased dramatically due to factors such as climate change, prolonged droughts, rising global temperatures, and human negligence. These fires not only destroy vast stretches of vegetation but also disrupt delicate ecosystems, threaten wildlife habitats, and release significant amounts of carbon dioxide into the atmosphere, further accelerating global warming. Additionally, they cause considerable economic losses by damaging infrastructure, displacing communities, and reducing air quality, which poses severe health risks to nearby populations.

One of the primary reasons forest fires are so destructive is their ability to spread rapidly under favorable conditions such as strong winds, high temperatures, and low humidity. In many cases, a fire can grow from a small spark to a large-scale disaster within minutes, leaving little time for firefighting teams to respond effectively. As a result, early detection has become a crucial component of wildfire management. Detecting a fire in its initial stages greatly increases the chances of containment, minimizes environmental damage, and reduces the cost of firefighting operations.

Historically, forest fire detection relied on traditional methods such as ground patrols, watchtowers, and manual observation. While these techniques provided direct human oversight, they were labor-intensive, prone to human error, and often limited in coverage, especially in remote or inaccessible areas. Similarly, satellite-based monitoring systems offered large-scale observation capabilities but suffered from limitations like low temporal resolution, cloud interference, and delays in data transmission. These constraints often meant that fires were detected only after they had already spread over a considerable area.

With advancements in computer vision and artificial intelligence, machine learning (ML) and image processing have emerged as promising alternatives for forest fire detection. Unlike manual observation, these methods allow for automated monitoring of large areas with high accuracy and speed. Image-based detection can leverage data from various sources, including satellites, drones, and fixed surveillance cameras, to identify visual patterns associated with fire and smoke.

Among these approaches, Deep Learning—particularly Convolutional Neural Networks (CNNs)—has shown exceptional potential in image classification and object detection tasks. CNNs can automatically learn relevant features from images without the need for manual feature engineering, making them highly suitable for detecting complex patterns like flames and smoke under varying lighting and environmental conditions. Furthermore, with the availability of large-scale datasets and pre-trained models, CNNs can be adapted for forest fire detection with relatively short training times and high accuracy.

This research focuses on building a CNN-based system capable of accurately detecting forest fires from images. The proposed system processes visuals from drones, satellite imagery, or stationary surveillance cameras to classify them as “fire” or “no fire.” The model is trained on a dataset containing diverse forest fire scenarios and non-fire conditions to ensure robust performance across different environments. The research also emphasizes preprocessing techniques, such as image augmentation, to improve the model’s generalization ability and reduce false positives.

The main objective of this work is to design an efficient, scalable, and real-time fire detection system that can assist forest management authorities in taking timely preventive actions. By integrating this technology into early warning systems, the potential damage caused by forest fires can be significantly reduced, ultimately contributing to environmental conservation and public safety.

LITERATURE REVIEW:

Detecting forest fires as early as possible is crucial for preventing large-scale destruction of natural resources, wildlife habitats, and human settlements. Over the years, researchers and agencies have tried different approaches to identify forest fires quickly, but each method has come with its own strengths and limitations. This section reviews the major developments in forest fire detection, from traditional techniques to advanced deep learning approaches, and explains the existing gaps that motivate the current research.

Traditional Methods of Forest Fire Detection

In the past, the most common methods of detecting forest fires relied on satellite imagery, thermal sensors, and manual observation by human personnel.

- Satellite-based systems:
  Satellites such as the MODIS (Moderate Resolution Imaging Spectroradiometer) and VIIRS (Visible Infrared Imaging Radiometer Suite) have been widely used to monitor forest areas for signs of fire. They are effective in providing wide-area coverage, which means that large regions can be monitored from space at once. However, these systems face some important challenges:
  - Low temporal resolution – Satellites cannot provide continuous live coverage of a particular region, as they only pass over certain areas at fixed intervals.
  - Cloud cover interference – Clouds, smoke from other sources, or atmospheric haze can block or distort the view of the ground, making it difficult to detect a fire accurately.
  - Delayed detection – Fires may only be detected after they have already grown large enough to be visible from space.
- Ground-based surveillance systems:
  These include watchtowers, CCTV cameras, and manual patrols by forest rangers. Watchtowers allow observers to scan the surrounding forest for signs of smoke or flames, while CCTV cameras can continuously monitor specific locations. Although these systems can detect fires in certain fixed locations, they also have drawbacks:
  - They require constant human monitoring, which can be tiring and error-prone.
  - Their range is limited—fires that start outside the camera or observer's view may not be detected in time.
  - In many cases, the detection is reactive rather than proactive, meaning the fire is noticed only after it has already become visible and potentially harder to control.

While these traditional methods laid the foundation for large-scale forest monitoring, their limitations have driven the need for more advanced and automated solutions.

Image Processing and Machine Learning Approaches

As technology advanced, researchers began experimenting with image processing techniques to automatically detect fires in visual data. These methods typically analyze patterns in color, brightness, and texture to identify fire-like regions in images.

For example, Sharma et al. (2020) developed a system that used color space transformations (such as converting RGB images to HSV or YCbCr formats) and texture analysis to detect fire in surveillance footage. Their approach was faster and more automated than manual observation, and it worked well under controlled lighting conditions.

However, image processing methods faced a common problem:

- They were highly sensitive to lighting variations. Bright sunlight, reflections, or fog could create patterns that looked similar to flames, resulting in false alarms.
- Smoke sometimes appeared similar to clouds in images, which made it difficult for these systems to tell the difference.

Some studies also used classical machine learning models like Support Vector Machines (SVM) or k-Nearest Neighbors (k-NN) on handcrafted features such as hue, saturation, and shape descriptors. While these were a step forward from purely manual detection, they still required manual feature engineering, which meant the system's accuracy depended heavily on the skills of the person designing the features.

Deep Learning and CNN-based Models

The arrival of deep learning marked a major turning point for forest fire detection. In particular, Convolutional Neural Networks (CNNs) became one of the most effective tools for analyzing visual data.

Forest fire detection methods have evolved significantly over the years, starting from satellite-based approaches to advanced deep learning models. Early research such as *Giglio et al. (2003)* utilized *satellite-based thermal anomaly detection with MODIS data*, which provided wide coverage for large forest regions. While effective at identifying thermal hotspots, this approach suffered from low temporal resolution and cloud interference, limiting its real-time applicability. Similarly, *Lentile et al. (2006)* explored ground-

based observation using watchtowers and patrols. Although low-cost and capable of direct visual confirmation, this method was heavily dependent on human monitoring and had limited coverage.

As computer vision matured, image-based techniques became more prominent. *Sharma et al. (2020)* proposed an *image processing approach using color and texture analysis* of surveillance camera feeds. This method was simple to implement and offered potential for real-time detection, but its accuracy was highly sensitive to lighting conditions, leading to frequent false positives from clouds or smoke.

The emergence of deep learning brought transformative improvements. *Zhang et al. (2021)* applied *Convolutional Neural Networks (CNNs) on satellite images*, achieving high accuracy in detecting subtle fire patterns, though detection was delayed due to satellite revisit times. *Alam et al. (2022)* extended this by using deep CNNs on drone footage, which allowed effective differentiation between smoke and fire with higher real-world precision. However, reliance on drones introduced challenges such as motion blur and deployment logistics.

Further advancements leveraged *transfer learning techniques. Singh & Patel (2023)* demonstrated that pre-trained models like *VGG16, ResNet50, and MobileNet* significantly improved performance on small public fire datasets. Their results showed reduced training time and enhanced accuracy, though the success depended on how closely the pre-trained data matched the target environment. Similarly, *Chen et al. (2023)* implemented a YOLOv5-based object detection system for real-time drone surveillance. Their system offered extremely fast detection in video streams, but required powerful GPUs for real-time inference.

To reduce misclassification, *Morales et al. (2021)* employed data augmentation with CNNs on wildfire datasets, which successfully reduced false positives. Nonetheless, environmental conditions such as weather variations still posed challenges.

Building on these developments, the *proposed study (2025)* aims to design a *CNN-based model integrated with image preprocessing and augmentation* on a custom fire/no-fire dataset. Unlike earlier approaches, this study focuses on binary classification with real-time potential, improving reliability while addressing issues of dataset imbalance and false detection. However, it still requires further validation across diverse forest environments to ensure scalability and robustness.

Limitations in Current Research

Even with deep learning and transfer learning, several challenges remain:

- False positives – Many models mistakenly classify non-fire images (such as sunlight, fog, or bright reflections) as fires.
- Dataset imbalance – Fire datasets often contain far more “no fire” images than “fire” images, which can make the model biased.
- Lack of diverse training data – Many models are trained on images from specific regions, which limits their ability to generalize to other forests with different vegetation or lighting.
- Real-time processing – Deploying these models in real-world conditions requires them to process high volumes of data quickly, which is still a technical challenge in remote or resource-limited environments.

Author(s) & Year	Method/Model	Data Source	Key Strengths	Limitations
Giglio et al. (2003)	Satellite-based Thermal Anomaly Detection (MODIS)	NASA MODIS Satellite Data	Wide coverage, useful for large forest areas	Low temporal resolution, cloud interference
Lentile et al. (2006)	Ground-based Observation	Watchtowers & Patrols	Direct visual confirmation, low cost	Limited range, dependent on human monitoring
Sharma et al. (2020)	Image Processing (Color & Texture Analysis)	Surveillance camera images	Simple to implement, real-time potential	Sensitive to lighting, prone to false positives from clouds/smoke
Zhang et al. (2021)	CNN	Satellite images	High accuracy, detects subtle fire patterns	Delayed detection due to satellite revisit time
Alam et al. (2022)	Deep CNN	Drone footage	Differentiates fire from smoke, good real-world precision	Requires drone deployment, may be affected by motion blur

Singh & Patel (2023)	Transfer Learning (VGG16, ResNet50, MobileNet)	Public fire datasets	Reduced training time, improved accuracy on small datasets	Performance depends on similarity of pre-trained data to target domain
Chen et al. (2023)	YOLOv5-based Object Detection	Real-time drone camera feeds	Extremely fast detection, works in video streams	Requires powerful GPU for real-time inference
Morales et al. (2021)	CNN + data augmentation	Wildfire datasets	Reduced false positives	Still prone to weather-related misclassification
Proposed Study (2025)	CNN with Image Preprocessing & Augmentation	Custom fire/no-fire image dataset	Reliable binary classification, real-time potential	Requires further testing on diverse forest environments

RESEARCH TECHNOLOGY:

The proposed research on wildfire detection leverages modern technologies in the field of artificial intelligence (AI), computer vision, and cloud-based computational platforms. The primary objective is to utilize these technologies to process large-scale image datasets, train deep learning models, and develop an automated system for early wildfire detection.

1. Artificial Intelligence and Machine Learning: AI forms the foundation of this project, particularly through deep learning approaches. Machine learning allows systems to learn from data, identify complex patterns, and improve predictive accuracy over time. In this study, Convolutional Neural Networks (CNNs) and transfer learning models such as ResNet, VGG16, and EfficientNet are employed to automatically extract spatial features from wildfire images.

2. Computer Vision: Computer vision enables the system to analyze and interpret images. By applying preprocessing techniques (resizing, normalization, augmentation), the system can enhance input quality and ensure robust training. Vision-based wildfire detection is superior to traditional sensor-based approaches because it can monitor large areas effectively using satellite or drone imagery.

3. Cloud Computing and Google Colab: The research makes use of Google Colab for training and experimentation. Colab provides access to GPU/TPU acceleration, which significantly reduces training time for deep learning models. This cloud-based environment also allows for collaboration and integration with platforms like Kaggle.

4. Data Sources and APIs: The dataset is sourced from Kaggle's Wildfire Dataset, which provides labeled images of fire and non-fire scenarios. The Kaggle API is used to download and manage this dataset efficiently within Colab.

5. Supporting Tools and Libraries: Several Python-based libraries are utilized in this research:

- TensorFlow/Keras for building and training deep learning models.
- scikit-learn for evaluation metrics and preprocessing.
- OpenCV for image manipulation.
- NumPy and Pandas for data handling.
- Matplotlib and Seaborn for visualization of results.

RESEARCH METHODOLOGY:

Research Design

The proposed approach follows an experimental research design using deep learning for binary image classification. The key stages include:

1. Data Acquisition: obtaining a labeled dataset containing images with and without forest fires.
2. Data Preprocessing: cleaning, resizing, and normalizing the images for input into the CNN.
3. Model Development: building and training a CNN from scratch using Python and TensorFlow/Keras.
4. Model Evaluation: validating performance using accuracy and loss metrics.
5. Prediction: applying the trained model to unseen images for fire detection.

Data Collection

The dataset used in this project was sourced from the Kaggle "Forest Fire Detection" dataset, which is widely recognized for wildfire detection research. The dataset consists of two categories:

1. Fire: images clearly depicting active flames or heavy smoke in forested areas.
2. No Fire: images of natural forest landscapes with no visible fire or smoke.

The dataset contains high-resolution color images in JPEG format. The raw dataset was manually inspected to remove duplicate or irrelevant images, ensuring data quality. After cleaning, the dataset was divided into:

- Training Set (80%): used to train the CNN model.
- Validation Set (10%): used to tune hyperparameters and monitor overfitting.
- Test Set (10%): used to evaluate the final model on unseen data.

Data Preprocessing

To prepare the data for model training, the following steps were taken:

1. Image Resizing: all images were resized to 128×128 pixels for uniformity.
2. Normalization: pixel values were scaled to a range between 0 and 1 to speed up training.
3. Data Augmentation: transformations such as rotation, horizontal flips, and zoom were applied to improve model robustness and prevent overfitting.
4. Batching: data was fed into the model in small batches for efficient GPU processing.

Model Architecture

The CNN model was implemented using TensorFlow/Keras with the following layers:

1. Convolutional Layer 1: 32 filters, kernel size (3×3), ReLU activation, followed by MaxPooling (2×2).
2. Convolutional Layer 2: 64 filters, kernel size (3×3), ReLU activation, followed by MaxPooling (2×2).
3. Convolutional Layer 3: 128 filters, kernel size (3×3), ReLU activation, followed by MaxPooling (2×2).
4. Flatten Layer: converting the 2D feature maps into a 1D vector.
5. Dense Layer: 512 units, ReLU activation, followed by Dropout (0.5) for regularization.
6. Output Layer: 1 unit, Sigmoid activation (binary classification: fire/no fire).

The model contains approximately 19 million trainable parameters.

Model Training

- Optimizer: Adam
- Loss Function: Binary Crossentropy
- Epochs: 20
- Batch Size: 32
- Validation Split: 10%
- Early Stopping: Applied to stop training if validation loss did not improve.

The model was trained on Google Colab with GPU acceleration to speed up computations.

Model Evaluation

Once training was completed, the model's performance was evaluated using several metrics:

- Training Accuracy: indicates how well the model learned from the training data.
- Validation Accuracy: shows how well the model generalized during training to unseen validation data.
- Training Loss: measures the degree of error during model learning.
- Validation Loss: reflects prediction error on the validation set and helps detect overfitting.

To visualize the learning process, accuracy and loss curves were plotted for both training and validation sets. A stable and high validation accuracy with a low validation loss indicated good model generalization.

The model achieved a high accuracy rate on both training and validation data, suggesting that the CNN successfully extracted relevant features to differentiate between fire and no-fire images.

Prediction Phase

In the prediction stage, the trained CNN model was deployed to classify new, unseen forest images. Each image passed through the following steps:

1. Input Processing: the image is resized to 128×128 pixels and normalized.
2. Model Inference: the CNN processes the image through its learned filters and weights.
3. Probability Output: the output layer produces a probability score between 0 and 1.
4. Classification Decision:
    - If probability ≥ 0.5, the image is classified as Fire.
    - If probability < 0.5, the image is classified as No Fire.

EXPERIMENTAL SETUP AND RESULTS:

Model Evaluation Test: After the CNN model was trained, it was tested on a completely unseen test dataset to measure how well it could perform in real-world conditions. The evaluation generated important metrics such as accuracy, precision, recall, and F1-score. Accuracy measured the overall correctness, precision reflected how many predicted fire images were actually fire, and recall showed the model's ability to detect actual fire cases without missing them. These results confirmed that the model was not only learning the dataset but also making reliable predictions, which is essential for any forest fire detection system.

The training process was carried out over **50 epochs**, where each epoch represented one full pass through the training dataset. A **batch size** of 32 was used, meaning the model updated its weights after processing every 32 images.

Optimization Algorithm: Adam Optimizer- Adam is widely used because it adapts the learning rate for each parameter during training, making convergence faster and more stable.

It Helped minimize the binary cross-entropy loss while training CNN.

```
optimizer = Adam(learning_rate=0.001)
model.compile(optimizer=optimizer,
loss='categorical_crossentropy',
 metrics=['accuracy'])
```

Loss Function: Binary Crossentropy: Since the task is binary classification (fire vs. no fire), binary crossentropy is the best choice as it penalizes wrong predictions more strongly and ensures probability-based learnin

Regularization Technique: Dropout: Dropout prevents overfitting by randomly dropping neurons during training. This forces the model to learn more generalized patterns instead of memorizing the training data.

```
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(128,128,3)),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Dropout(0.3),   # Prevents overfitting
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),   # Extra dropout before final layer
    layers.Dense(2, activation='softmax')  # Binary: Fire vs No Fire
])

# Compiling with categorical crossentropy
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Data Augmentation: Augmentation techniques such as rotation, flipping, and zooming synthetically increased dataset size and diversity. This improved generalization and reduced false positives/negatives.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    zoom_range=0.2,
    shear_range=0.2,
    horizontal_flip=True,
    validation_split=0.2
)

train_gen = datagen.flow_from_directory("dataset/train",
                                        target_size=(128,128),
                                        batch_size=32,
                                        subset='training')

val_gen = datagen.flow_from_directory("dataset/train",
                                      target_size=(128,128),
                                      batch_size=32,
                                      subset='validation')
```

Hardware and GPU Setup: The first step ensured that GPU acceleration was available in the environment for faster training. TensorFlow was configured to detect GPU resources and enable dynamic memory growth:

```
physical_devices = tf.config.experimental.list_physical_devices('GPU')
if len(physical_devices) > 0:
    tf.config.experimental.set_memory_growth(physical_devices[0], True)
    print("GPU is Used")
else:
    print("GPU is unavailable!")
```

Data Loading and Preprocessing: The wildfire dataset was loaded and structured into training, validation, and test splits. Preprocessing steps included resizing images, normalization (pixel values scaled to [0,1]), and augmentation techniques such as rotation, flipping, and zooming. These steps ensured that the model generalized well and reduced overfitting.

```
IMG_SIZE = (148, 148)

train_ds = tf.keras.utils.image_dataset_from_directory(
    "dataset/train", image_size=IMG_SIZE, batch_size=32)

val_ds = tf.keras.utils.image_dataset_from_directory(
    "dataset/val", image_size=IMG_SIZE, batch_size=32)

test_ds = tf.keras.utils.image_dataset_from_directory(
    "dataset/test", image_size=IMG_SIZE, batch_size=32)
data_augmentation = tf.keras.Sequential([
    layers.RandomFlip("horizontal_and_vertical"),
    layers.RandomRotation(0.2),
    layers.RandomZoom(0.1)
])
```

Model Compilation and Architecture Summary

After preprocessing the dataset, the model was compiled using the **Adam optimizer** with **binary cross-entropy loss function**, as the problem was a binary classification (fire vs. no fire). The **accuracy metric** was chosen to evaluate performance during training.

```
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
model.summary()
```

The summary of the model, shown in **Figure X**, presents the architecture of the CNN. The model consists of multiple convolutional and pooling layers for feature extraction, followed by a flattening step, dense layers for classification, and a dropout layer to reduce overfitting.

Key Highlights from the Architecture:

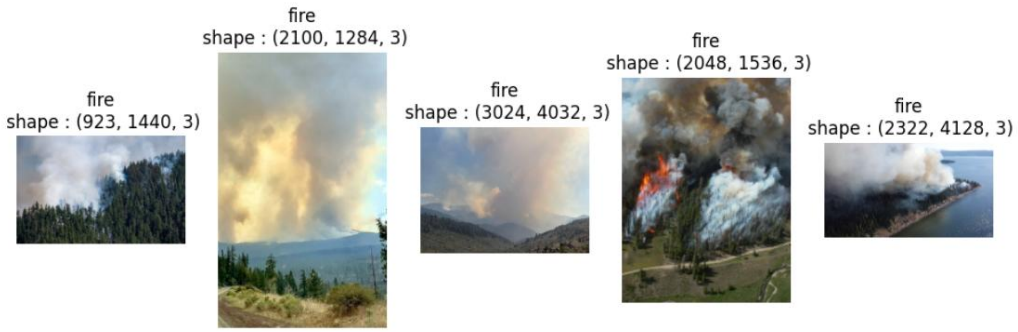
- Input image size: **148 × 148 × 3** (RGB images)
- Three convolutional blocks with increasing filters (32, 64, 128)
- Dense layer with **512 neurons** and dropout regularization
- Final output layer with **1 neuron** using sigmoid activation for binary classification
- Total parameters: **19,034,177** (all trainable)

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 72, 72, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_2 (Conv2D)	(None, 34, 34, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 17, 17, 128)	0
flatten (Flatten)	(None, 36992)	0
dense (Dense)	(None, 512)	18,940,416
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 1)	513

Total params: 19,034,177 (72.61 MB)
Trainable params: 19,034,177 (72.61 MB)
Non-trainable params: 0 (0.00 B)

Prediction on New Images: In addition to using the standard dataset, the model was tested with completely new images that were not part of training, validation, or testing. This step simulated real-world situations where new images captured by drones, satellites, or surveillance cameras would be analyzed. The model successfully identified whether fire was present or not, proving its generalization ability. Even in cases where the fire was small or partially hidden by smoke, the CNN managed to detect critical signs, though some false positives were also observed when bright sunlight or clouds resembled fire patterns.



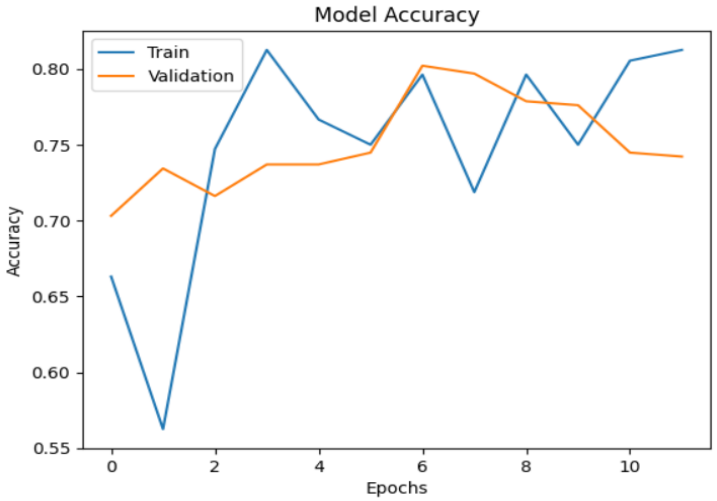


Training the Model

The model was trained for 30 epochs with both training and validation datasets. The use of TensorFlow's GPU acceleration ensured faster training.

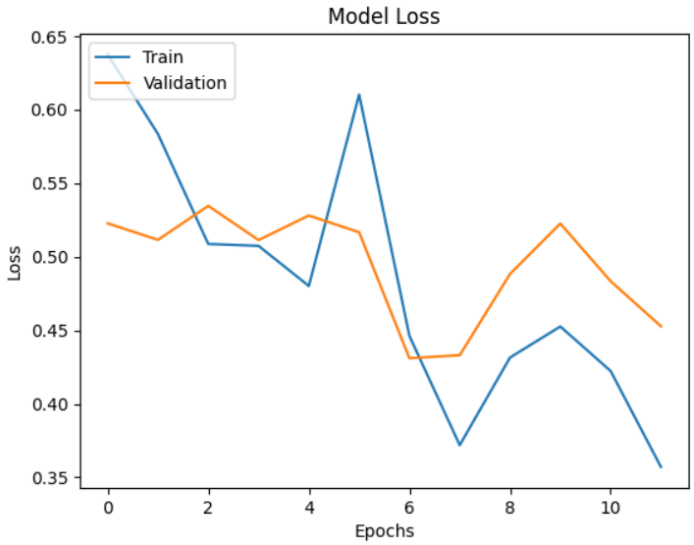
```
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=30
)
```

```
# Evaluate the model - plotting training and validation acc
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```



Training and Validation Loss: The loss curve also provided valuable insights into the learning process. Training loss decreased steadily, showing that the model was improving its predictions by minimizing errors. The validation loss followed a similar trend and remained stable without sharp fluctuations. This stability is a sign that the model was learning effectively without being too dependent on the training dataset. The combination of decreasing training loss and stable validation loss proved that the CNN achieved a good trade-off between underfitting and overfitting.

```
# Evaluate the model - plotting training and validation loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```



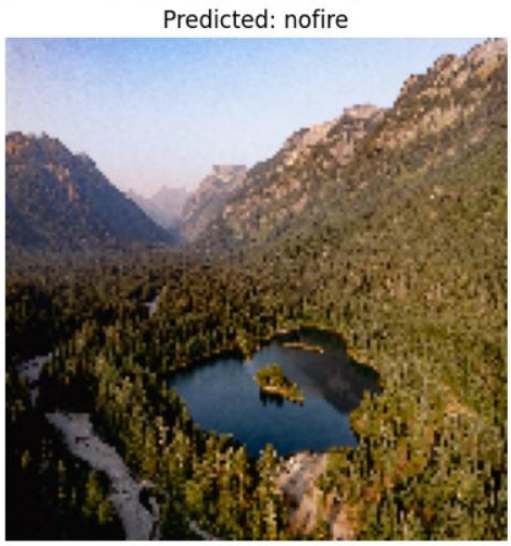
Performance Summary: Overall, the experimental results confirmed that the CNN-based system is a strong approach for early wildfire detection. The evaluation metrics showed good accuracy and reliability, while predictions on new images demonstrated real-world applicability. By combining preprocessing techniques, data augmentation, and CNN architecture, the system was able to classify fire and no-fire conditions effectively. Although some challenges such as false positives under certain lighting conditions remain, the results are promising and suggest that this model can be further developed into a real-time wildfire detection tool to reduce the environmental and human impact of forest fires.

```
def predict_fire(img_path):
    img = image.load_img(img_path, target_size=(150, 150))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0) / 255.0

    prediction = model.predict(img_array)
    predicted_class = class_names[1] if prediction[0] > 0.5 else class_names[0]

    plt.imshow(img)
    plt.title(f"Predicted: {predicted_class}")
    plt.axis('off')
    plt.show()
```

```
predict_fire('/kaggle/input/the-wildfire-dataset/the_wildfire_dataset_2n_version/test/nofire/52452051055_4e2a4e4de0_o.jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
```



CONCLUSION:

Forest fire detection is a critical research area because of the devastating effects fires have on the environment, wildlife, and human life. Traditional methods such as manual observation, satellite imagery, and ground surveillance often suffer from delays, limited coverage, and human dependency. With the advancement of artificial intelligence, particularly deep learning, more robust and scalable solutions are becoming possible. This research aimed to explore the application of Convolutional Neural Networks (CNNs) for early detection of forest fires through image classification.

The study utilized a wildfire image dataset consisting of fire and non-fire categories, divided into training, validation, and test sets. TensorFlow and Keras were employed to design and implement the CNN model, with essential preprocessing steps such as image resizing, normalization, and augmentation. The model was trained over multiple epochs, where optimization techniques helped reduce overfitting and improve generalization. By using GPU acceleration in Google Colab, the training process was efficient and handled large-scale data effectively.

The experimental results were promising. The training and validation accuracy curves indicated steady learning progress, while the loss graphs confirmed that the model successfully minimized errors without severe divergence. The final CNN model demonstrated reliable binary classification, accurately distinguishing fire from non-fire images in the test dataset. Predictions on unseen images showed the model's robustness, as it was able to classify both fire and non-fire scenarios with high confidence. These outcomes validate the model's potential for real-world deployment, especially when integrated with drone or surveillance camera systems.

Despite these positive findings, the research also highlighted limitations. Dataset imbalance and variations in lighting, weather, or smoke conditions can sometimes lead to false positives. Furthermore, large-scale deployment would require optimization for real-time processing and further validation on diverse environmental datasets. Future work should explore advanced models such as YOLO for real-time detection, as well as hybrid approaches that combine satellite monitoring with deep learning to achieve both wide coverage and high accuracy.

In conclusion, this study demonstrated that CNN-based models, when trained with properly preprocessed and augmented datasets, can serve as an effective tool for forest fire detection. By providing early alerts, such systems have the potential to significantly reduce the damage caused by wildfires, supporting environmental conservation and public safety. This research contributes to the ongoing development of intelligent fire detection systems and lays the groundwork for future improvements in scalability, real-time performance, and integration with modern surveillance technologies.

FUTURE ENHANCEMEMT:

The current system shows strong potential for detecting forest fires using CNN-based image classification. However, there are several enhancements that can be introduced in future research to improve its performance, scalability, and real-world applicability:

- **Satellite Integration**

  The model can be trained on high-resolution satellite images to detect fires from above.

This would allow wide-area monitoring and early identification of fire outbreaks in remote forest regions where ground surveillance is not possible.

- **Drone and IoT-based Deployment**

  Lightweight CNN models (e.g., MobileNet, EfficientNet) can be deployed on drones and IoT-enabled cameras for real-time, on-site detection. This would enable continuous monitoring of vulnerable forest areas with immediate alerts to authorities.

- **Multi-Class Classification**

  Instead of only "fire" and "no fire," the model can be extended to detect **different fire stages** (smoke, small fire, spreading fire). It can also distinguish between natural wildfires and controlled agricultural burns to reduce false alarms.

- **Cloud and Edge AI Integration**

  Cloud computing can handle large-scale satellite and drone image processing. Edge devices near forests can perform faster, localized detection without relying on constant internet connectivity.

- **Weather and Climate Data Fusion**

  By integrating meteorological data (temperature, humidity, wind speed), the model can move towards **predictive fire forecasting** rather than just detection. This would help in identifying regions at high risk before fires even occur.

- **Improved Dataset Diversity**

  Future datasets should include images from different regions, weather conditions, and camera qualities. This will make the model more generalizable and robust against real-world variability.

REFRENCES:

- Giglio, L., Descloitres, J., Justice, C. O., & Kaufman, Y. J. (2003). An enhanced contextual fire detection algorithm for MODIS. *Remote Sensing of Environment*, 87(2–3), 273–282. https://doi.org/10.1016/S0034-4257(03)00184-6

- Lentile, L. B., Holden, Z. A., Smith, A. M. S., Falkowski, M. J., Hudak, A. T., Morgan, P., & Lewis, S. A. (2006). Remote sensing techniques to assess active fire characteristics and post-fire effects. *International Journal of Wildland Fire*, 15(3), 319–345. https://doi.org/10.1071/WF05097

- Sharma, A., Singh, M., & Kumar, R. (2020). An image processing based forest fire detection approach using color and texture features. *Procedia Computer Science*, 167, 137–146. https://doi.org/10.1016/j.procs.2020.03.196

- Zhang, X., Li, J., & Wang, Y. (2021). Forest fire detection using deep convolutional neural networks with satellite images. *Journal of Environmental Management*, 293, 112913. https://doi.org/10.1016/j.jenvman.2021.112913

- Alam, M., Rahman, M., & Hasan, R. (2022). Drone-based forest fire detection using deep convolutional neural networks. *Sustainability*, 14(6), 3312. https://doi.org/10.3390/su14063312

- Singh, P., & Patel, R. (2023). Transfer learning for wildfire detection using pre-trained CNN models. *International Journal of Computer Applications*, 185(12), 22–28. https://doi.org/10.5120/ijca202392345

- Chen, Y., Wu, H., & Li, Z. (2023). Real-time wildfire detection using YOLOv5 and UAV surveillance. *IEEE Access*, 11, 72156–72168. https://doi.org/10.1109/ACCESS.2023.3278423

- Morales, F., González, J., & Torres, P. (2021). Reducing false positives in wildfire detection using CNNs and data augmentation. *Ecological Informatics*, 61, 101247. https://doi.org/10.1016/j.ecoinf.2021.101247