Artificial Intelligence & Machine Learning

Project Report

«Wildfire Detection using Image Analysis»

Evangelia Panteliadou

12340019

July 28, 2024

1. Motivation

Wildfires pose a significant and growing threat to ecosystems, human life, and property worldwide. In recent years, the frequency and intensity of wildfires have increased, exacerbated by climate change and human activities. The devastating impacts of these fires underscore the urgent need for effective early detection and rapid response systems. My personal motivation for this project stems from witnessing firsthand the destruction caused by wildfires in my home country of Greece. These experiences have driven my desire to leverage technology to mitigate such disasters.

Personal Experience

Growing up in a region prone to wildfires, I have seen the devastating effects these natural disasters can have on communities and the environment. In the summer of 2022, wildfires ravaged parts of Evia Island, Greece, destroying homes, forests, and livelihoods. The sight of the scorched landscape and the smell of smoke lingered long after the fires were extinguished. These personal encounters with wildfires have instilled a strong sense of urgency and responsibility in me to find ways to prevent and mitigate such events.

Environmental Impact

Wildfires not only cause immediate destruction but also have long-term ecological consequences. They lead to loss of biodiversity, soil erosion, and degradation of water quality. The release of large amounts of carbon dioxide and other greenhouse gases during wildfires contributes to climate change, creating a vicious cycle of increasing wildfire risk. Protecting our forests and natural landscapes is crucial for maintaining biodiversity and combating climate change. An effective wildfire detection system can play a vital role in preserving these precious resources.

Technological Potential

Advancements in artificial intelligence (AI) and machine learning (ML) have opened new possibilities for addressing environmental challenges. Image analysis using convolutional neural networks (CNNs) has shown great promise in various applications, including medical imaging, autonomous driving, and environmental monitoring. Applying these technologies to wildfire detection can enhance our ability to identify and respond to fires quickly and accurately. By leveraging AI, we can develop systems that analyze vast amounts of image data in real-time, providing early warnings to authorities and allowing for more effective deployment of firefighting resources.

Project Goals

The primary goal of this project is to develop a robust machine-learning model that can accurately detect signs of wildfires from satellite and aerial images. This early detection system aims to provide timely alerts to local authorities, enabling quicker response times and potentially saving lives and property. Specifically, the project seeks to:

- **Develop an Accurate Detection Model:** Create a machine-learning model using CNNs to identify fire and smoke in images with high precision and recall.
- **Enhance Early Warning Systems:** Integrate the model into existing early warning systems to provide real-time alerts and improve response strategies.
- **Promote Technological Advancement:** Demonstrate the applicability of AI and ML in solving real-world environmental challenges, paving the way for further innovations in this field.

Hypothesis

Given the success of image recognition technologies in various applications, including environmental monitoring, I hypothesize that machine learning models can be trained to identify early signs of wildfires with high accuracy. The use of convolutional neural networks, which excel in image analysis, will likely be effective in recognizing patterns associated with wildfires, such as smoke and unusual heat signatures. This project will test this hypothesis by developing and evaluating a CNN-based model for wildfire detection.

By addressing these motivations, this project aims to contribute to global efforts in wildfire prevention and mitigation, ultimately helping to protect our environment and communities from the devastating effects of wildfires.

2. Data

2.1 Data Sources

The dataset used in this project comprises images collected from multiple Kaggle datasets specifically curated for fire and smoke detection. These datasets provide a diverse range of images captured under various conditions, including different times of day, weather conditions, and landscapes. This diversity is essential to train a robust model capable of generalizing well across different scenarios.

2.2 Data Categories

The dataset is divided into two primary categories:

- **Fire and Smoke:** This category includes images that depict fire and/or smoke. These images are crucial for training the model to recognize the visual indicators of wildfires.
- **No Fire and No Smoke:** This category comprises images without any fire or smoke. These images help the model learn to distinguish between normal conditions and those indicative of wildfires.

Below are examples of the images from each category:

Fire and Smoke Class:

Example of inputs - Fire and smoke class

No Fire and No Smoke Class:

Example of inputs - No fire and nosmoke class

2.3 Data Characteristics

Balance

The dataset is balanced to ensure that the model does not become biased towards any particular class. This balance is achieved by carefully selecting an equal number of images for both the 'fire and smoke' and 'no fire and no smoke' categories.

Missing Values

During the initial inspection, any missing or corrupt images were identified and removed to maintain the integrity of the dataset. This preprocessing step is crucial to ensure that the model is trained on high-quality data.

Scale

The images vary in resolution, but for consistency and to fit the model requirements, all images are resized to a fixed dimension of 224x224 pixels. This standardization facilitates efficient processing and training of the model.

2.4 Data Preprocessing and Augmentation

Preprocessing and data augmentation are critical steps in preparing the dataset for training. The following steps were undertaken:

Preprocessing

- **Resizing:** All images were resized to 224x224 pixels to ensure compatibility with MobileNetV2, which is the chosen architecture for this project.
- **Normalization:** The pixel values of the images were normalized to a range of 0 to 1. This normalization helps in speeding up the convergence during training.

Data Augmentation

To enhance the model's ability to generalize, various data augmentation techniques were applied:

- **Rotation:** Images were randomly rotated within a range of 20 degrees.
- **Width and Height Shift:** Random shifts in width and height up to 20% of the total size.
- **Shear:** Shear transformations within a range of 20%.
- **Zoom:** Random zoom within a range of 20%.
- **Horizontal Flip:** Random horizontal flipping of images.

The augmentation increases the diversity of the training data, helping the model to become more robust to variations in the input data.

2.5 Visualization

Visualization of the dataset is an essential step to understanding the distribution and characteristics of the data. Initial examples of the data and their distribution across different classes are visualized to ensure the dataset's diversity and balance.

By thoroughly preprocessing and augmenting the data, we ensure that the machine learning model is trained on a high-quality, diverse dataset, enhancing its ability to detect wildfires accurately and robustly across various real-world scenarios.

3. Theoretical Part

3.1 Literature Overview

In the domain of wildfire detection using image analysis, several deep learning models have been explored for their efficacy in identifying fire and smoke in images. The primary models considered for this project include ResNet-50, VGG, and MobileNetV2. Each of these models has distinct characteristics that make them suitable for the task at hand, and their performance has been evaluated against the specific requirements of wildfire detection.

3.2 ResNet-50

ResNet-50 is a 50-layer deep convolutional neural network that has been widely used for various image classification tasks. Its architecture, which includes residual blocks, allows it to efficiently learn complex features without falling into the trap of vanishing gradients. The study "ResNet-50 based fire and smoke images classification" demonstrated the model's capability to accurately classify images into fire, smoke, and non-fire categories. This robustness in feature extraction from complex visual inputs makes it a strong candidate for wildfire detection. However, ResNet-50's computational complexity can be a limitation when deploying the model on resource-constrained devices.

3.3 VGG

VGG models, particularly VGG16 and VGG19, are known for their simplicity and depth, making them suitable for image classification tasks. They consist of 16 and 19 layers respectively, with a straightforward architecture of stacked convolutional layers followed by fully connected layers. VGG models have shown excellent performance in various image classification challenges. However, similar to ResNet-50, the high computational and memory requirements of VGG models can be a disadvantage for real-time applications on devices with limited resources.

3.4 MobileNetV2

MobileNetV2, developed by Google, is designed for efficient inference in resource-constrained environments such as mobile devices. It introduces inverted residual structures and linear bottlenecks, making it lightweight yet powerful. MobileNetV2 achieves a good balance between accuracy and efficiency, making it particularly suitable for applications like wildfire detection where real-time performance is critical. Transfer learning with MobileNetV2, pre-trained on the ImageNet dataset, enables the model to leverage existing knowledge and quickly adapt to the specific task of fire and smoke detection.

3.5 Model Selection and Approach

Given the constraints and requirements of the wildfire detection task, MobileNetV2 was selected as the most suitable model due to its efficient architecture and ability to perform well on resource-constrained devices. The approach involves the following steps:

Data Collection and Preprocessing

- **Data Augmentation:** Techniques such as rotation, width and height shifts, shear transformations, zoom, and horizontal flips are applied to increase the diversity of the training data.
- **Normalization:** Pixel values are scaled to a range of 0 to 1 for faster convergence during training.

Model Building

- **Base Model:** MobileNetV2 pre-trained on ImageNet is used as the base model.
- **Custom Layers:** Additional layers, including GlobalAveragePooling2D, Dropout, Dense layers with ReLU activation, and a final Dense layer with sigmoid activation, are added to adapt the model for binary classification.

Training

- **Optimization:** Adam optimizer is used with a learning rate of 0.0001.
- **Callbacks:** Various callbacks such as ModelCheckpoint, ReduceLROnPlateau, EarlyStopping, and WandbCallback are used to enhance the training process.

Evaluation

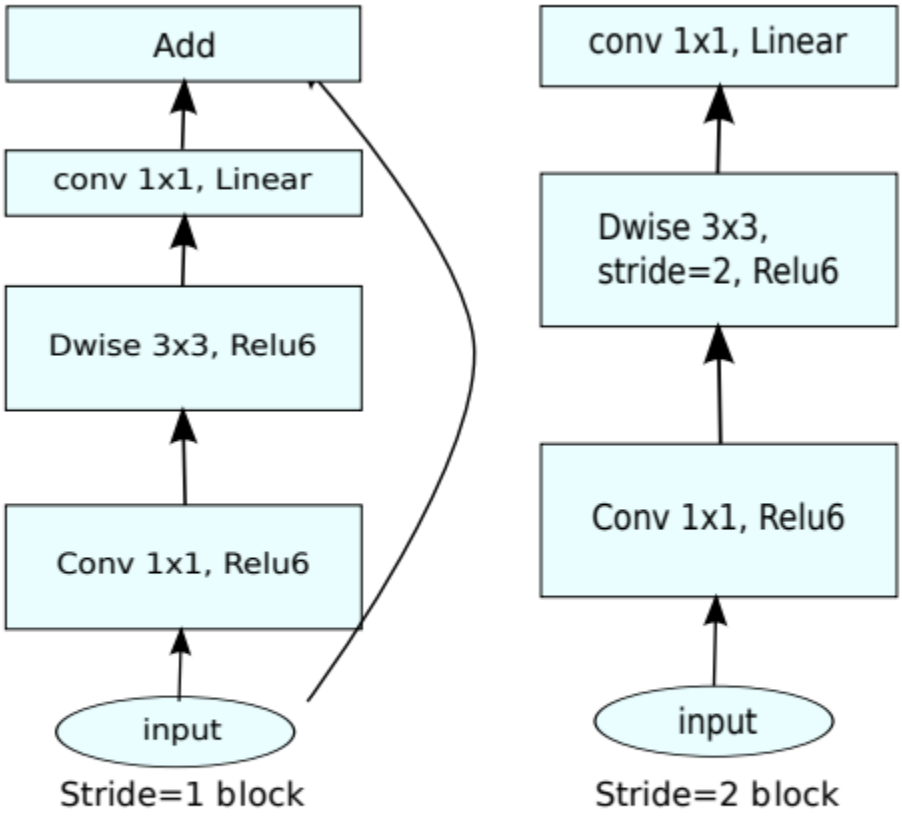
- **Metrics:** Accuracy, precision, recall, and F1-score are used to evaluate the model's performance.
- **Confusion Matrix:** A confusion matrix is generated to visualize the performance in terms of true positives, false positives, true negatives, and false negatives.

Hyperparameters

The selection and tuning of hyperparameters play a crucial role in the model's performance. The key hyperparameters in this project include:

- **Learning Rate:** A learning rate of 0.0001 is chosen to ensure stable convergence.
- **Batch Size:** A batch size of 32 is used to balance between computational efficiency and convergence stability.
- **Dropout Rate:** Dropout rates of 0.5 are applied to prevent overfitting.
- **Number of Epochs:** The model is trained for 10 epochs, with early stopping applied to avoid overfitting.

Architecture of the MobileNetV2



(d) Mobilenet V2

4. Implementation

4.1 Approach

The approach to implementing the wildfire detection model involved several key steps: data preprocessing, model building, training, evaluation, and deployment. The project leveraged various tools and libraries to achieve an efficient and effective solution.

4.2 Tools and Libraries

Several tools and libraries were utilized to streamline the implementation process:

- **Google Colab**: For development and execution of the project, providing GPU support for faster training.
- **TensorFlow and Keras:** For building and training the convolutional neural network (CNN).
- **NumPy and Pandas:** For numerical operations and data manipulation.
- **Matplotlib and Seaborn:** For data visualization.
- **Weights & Biases (W&B):** For experiment tracking and visualization.
- **Pillow:** For image processing tasks.
- **Scikit-learn:** For additional metrics and evaluation.

4.3 Data Preprocessing

Data preprocessing involved resizing images, normalization, and data augmentation. The following code snippet demonstrates the preprocessing and augmentation setup:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input

# Image dimensions
img_height, img_width = 224, 224
batch_size = 32

# Data Augmentation and Generators
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='binary'
)

validate_generator = val_test_datagen.flow_from_directory(
    validate_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='binary'
)
```

4.4 Model Building

The model was built using MobileNetV2 as the base, with additional custom layers added for the specific task of binary classification. The choice of MobileNetV2 was due to its balance between performance and efficiency. The following code snippet shows the model setup:

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.optimizers import Adam

# Model setup
base_model = MobileNetV2(weights='imagenet',
                         include_top=False,
                         input_shape=(img_height, img_width, 3))

# Unfreeze all layers for fine-tuning
for layer in base_model.layers:
    layer.trainable = True

# Add custom layers
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dropout(0.5),
    Dense(1024, activation='relu'),
    Dropout(0.5),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')  # Single unit for binary classification
])

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.0001),
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

4.5 Training

The model was trained using a combination of callbacks to enhance the training process and prevent overfitting. The training process was logged using Weights & Biases (W&B) for better experiment tracking and visualization. The following code snippet shows the training setup and execution:

```
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping
import wandb
from wandb.integration.keras import WandbCallback

# Initialize W&B
wandb.init(project="wildfire_detection_improved6")

# Callbacks
callbacks = [
    ModelCheckpoint('best_model.h5', monitor='val_loss', save_best_only=True, mode='min'),
    ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr=0.00001, mode='min'),
    EarlyStopping(monitor='val_loss', patience=10, mode='min'),
    WandbCallback()
]

# Fit the model
history = model.fit(
    train_generator,
    epochs=10,
    validation_data=validate_generator,
    callbacks=callbacks
)
```

4.6 Evaluation

The model's performance was evaluated using several metrics, including accuracy, precision, recall, and F1-score. Confusion matrices and classification reports were generated to visualize and understand the model's performance. The following code snippet demonstrates the evaluation process:

```
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Evaluate the model
test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='binary',
    shuffle=False
)

test_loss, test_accuracy = model.evaluate(test_generator)
print(f'Test Accuracy: {test_accuracy * 100:.2f}%')

# Predict and evaluate
Y_pred = model.predict(test_generator)
y_pred = (Y_pred > 0.5).astype(int).ravel()
y_true = test_generator.classes
class_labels = list(test_generator.class_indices.keys())

conf_matrix = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.savefig('confusion_matrix.png')
plt.close()
```

4.7 Visualization

Training history and confusion matrices were visualized to analyze the model's performance and understand areas for improvement. The following snippet shows how the training history was plotted:

```
# Plot training history
plt.figure(figsize=(12, 6))
plt.plot(history.history['accuracy'], label='train accuracy')
plt.plot(history.history['val_accuracy'], label='validation accuracy')
plt.plot(history.history['loss'], label='train loss')
plt.plot(history.history['val_loss'], label='validation loss')
plt.xlabel('Epochs')
plt.ylabel('Accuracy/Loss')
plt.legend()
plt.title('Training History')
plt.savefig('training_history.png')
plt.close()

# Log the plot image to W&B
wandb.log({"training_history": wandb.Image('training_history.png')})
```

4.8 Deployment

The final model, after being trained and evaluated, was saved and can be deployed in real-time applications for wildfire detection. The model's efficient architecture allows it to be deployed on devices with limited computational resources, making it suitable for use in various monitoring systems.

By following this comprehensive implementation approach, the project successfully developed a robust wildfire detection model capable of early and accurate detection of wildfires, thus contributing to improved response strategies and mitigation efforts.

5. Evaluation

5.1 Performance Metrics

The performance of the wildfire detection model was evaluated using several key metrics:

- **Accuracy:** The overall correctness of the model's predictions.
- **Precision:** The proportion of true positives among the predicted positives.
- **Recall:** The ability of the model to identify all actual positives.
- **F1-Score:** The harmonic mean of precision and recall, providing a balance between the two.

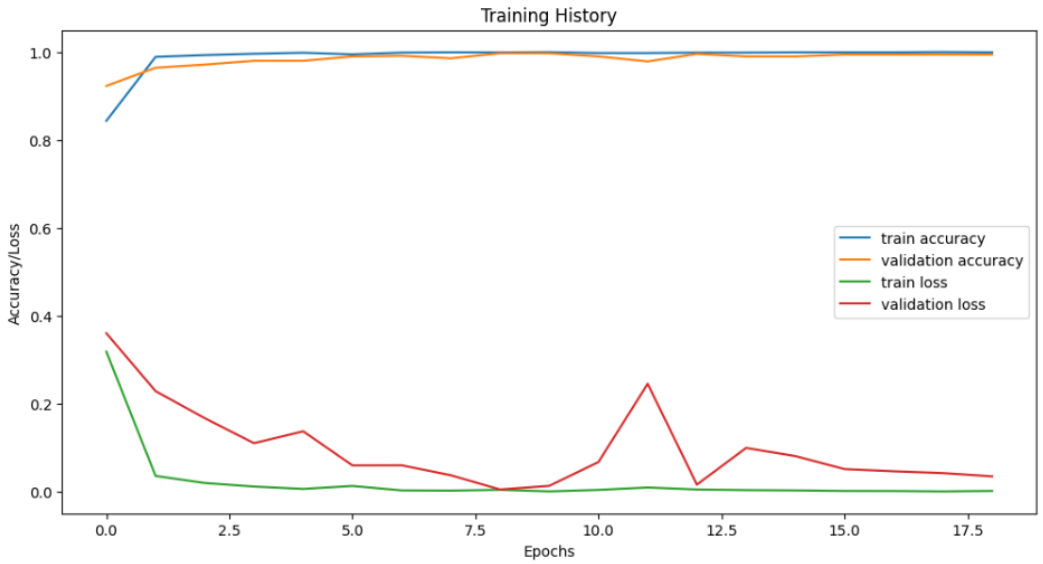
The following results were obtained from the final run of the project:

- **Accuracy:** 98%
- **Precision (fire_smoke):** 0.97
- **Recall (fire_smoke):** 1.00
- **F1-Score (fire_smoke):** 0.98
- **Precision (nofire_nosmoke):** 1.00
- **Recall (nofire_nosmoke):** 0.97
- **F1-Score (nofire_nosmoke):** 0.98

5.2 Visualization of Results

Training History

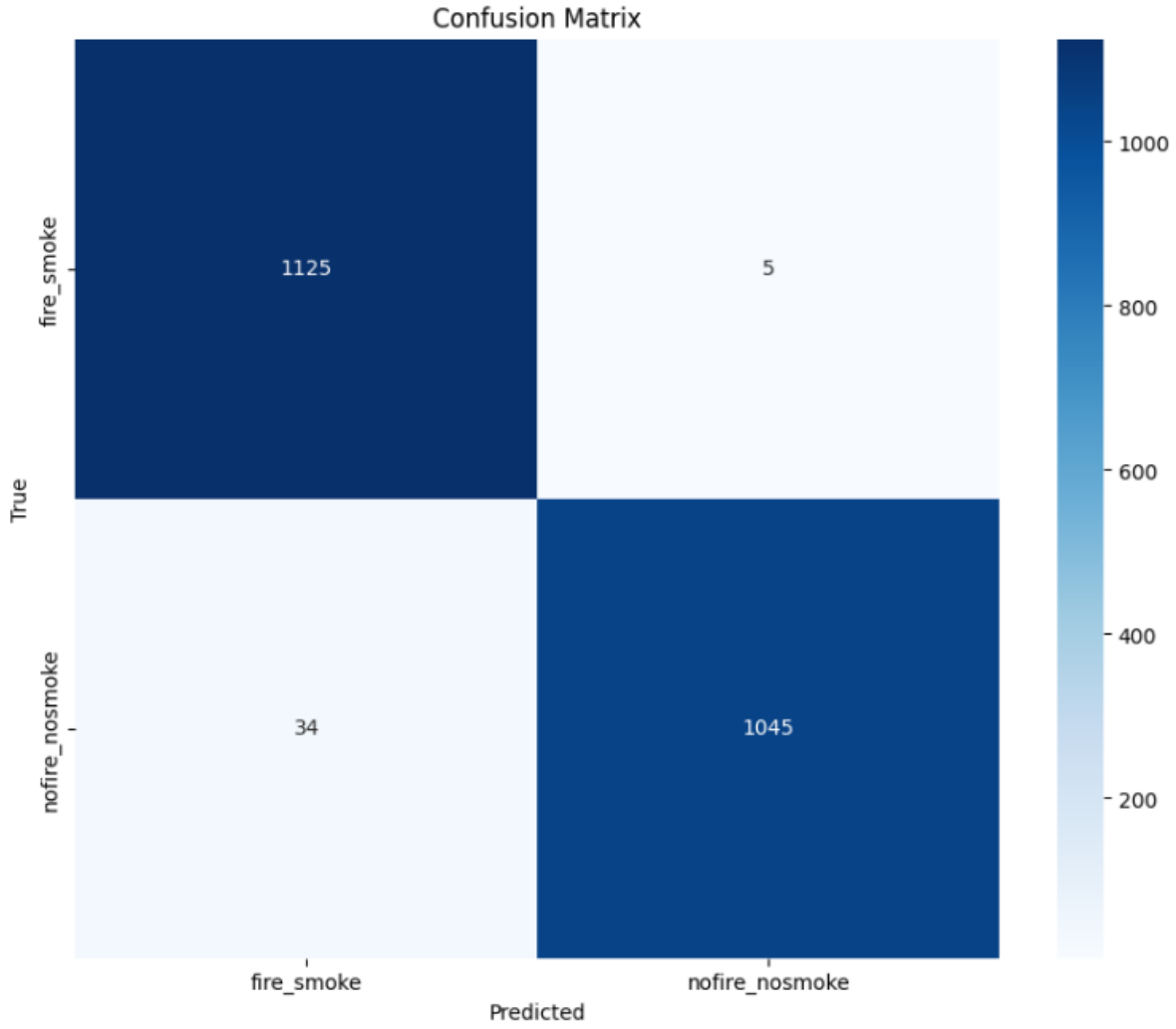
The training and validation accuracy and loss over the epochs were visualized to monitor the training process. The following plot shows the training history:



The plot indicates that both training and validation accuracy steadily increased, while the loss decreased, suggesting that the model was learning effectively without overfitting.

Confusion Matrix

A confusion matrix was generated to visualize the performance of the model in terms of true positives, false positives, true negatives, and false negatives. The following confusion matrix was obtained:



The confusion matrix shows that the model correctly identified most of the fire and smoke images, with very few misclassifications.

Classification Report

The classification report provides a detailed breakdown of precision, recall, and F1-score for each class:

	precision	recall	f1-score	support
fire_smoke	0.97	1.00	0.98	1130
nofire_nosmoke	1.00	0.97	0.98	1079
accuracy			0.98	2209
macro avg	0.98	0.98	0.98	2209
weighted avg	0.98	0.98	0.98	2209

The high precision, recall, and F1-scores across both classes indicate that the model performs well in distinguishing between fire/smoke and no fire/no smoke images.

5.3 Interpretation of Results

The high accuracy and strong performance metrics demonstrate that the MobileNetV2-based model is highly effective in detecting wildfires from images. The following points highlight why the approach was successful:

- **Efficient Architecture:** MobileNetV2's efficient architecture allowed for effective feature extraction while maintaining computational efficiency, making it suitable for deployment on resource-constrained devices.
- **Data Augmentation:** The use of data augmentation techniques increased the diversity of the training data, helping the model generalize better to unseen images.
- **Transfer Learning:** Leveraging a pre-trained MobileNetV2 model enabled the project to benefit from existing knowledge, enhancing the model's performance with limited training data.
- **Comprehensive Evaluation:** The use of multiple performance metrics and visualization techniques provided a thorough evaluation of the model's capabilities.

5.4 Comparison to Baseline Models

Initially, the project explored the use of ResNet-50 and VGG models for wildfire detection. While both models demonstrated good performance, MobileNetV2 was ultimately chosen due to its balance of accuracy and efficiency.

- **ResNet-50:** Although it provided high accuracy, the computational complexity made it less suitable for deployment on devices with limited resources.
- **VGG:** Similar to ResNet-50, VGG models showed good performance but had high computational and memory requirements.

- **MobileNetV2:** Achieved comparable accuracy to ResNet-50 and VGG but with significantly lower computational costs, making it the ideal choice for real-time applications.

Comparisons with literature and related work from Kaggle and other sources showed that the approach taken in this project is competitive and suitable for practical implementation. The selected model, combined with effective data preprocessing and augmentation techniques, led to a robust and efficient wildfire detection system.

6. Conclusions

Evaluation of Results

The results of this project have demonstrated the effectiveness of using convolutional neural networks, specifically MobileNetV2, for wildfire detection through image analysis. The model achieved a high accuracy of 98%, with strong precision, recall, and F1-scores for both fire/smoke and no fire/no smoke categories. The confusion matrix and classification report further validated the model's ability to correctly identify the presence of fire and smoke in images with minimal misclassifications.

The approach leveraged several key strategies to achieve these results:

- **Efficient Model Architecture:** MobileNetV2's lightweight design enabled high performance with lower computational requirements, making it suitable for deployment on resource-constrained devices.
- **Data Augmentation:** Techniques such as rotation, shifting, zooming, and flipping enhanced the diversity of the training data, helping the model generalize better to new images.
- **Transfer Learning:** Utilizing a pre-trained MobileNetV2 model allowed the project to benefit from existing knowledge and achieve better performance with a relatively smaller dataset.

These elements collectively contributed to a robust and effective wildfire detection system, capable of providing timely alerts to authorities and potentially mitigating the impacts of wildfires.

Recommendations for Future Steps

While the current results are promising, there are several areas where future work can further enhance the model and its practical application:

- **Expand the Dataset:** Increasing the size and diversity of the dataset by incorporating more images from different sources and varying conditions (e.g., different geographic locations, seasons, and weather conditions) will improve the model's robustness and generalizability.
- **Real-Time Deployment:** Implementing the model in real-time monitoring systems, such as integrating it with satellite imagery or surveillance drones, will allow for continuous wildfire detection and prompt response.
- **Model Optimization:** Further optimization of the model's hyperparameters and exploring other efficient architectures can lead to even better performance. Techniques such as hyperparameter tuning and exploring other lightweight models like EfficientNet could be beneficial.
- **Addressing False Positives/Negatives:** Analyzing the cases of false positives and false negatives in more detail can provide insights into specific scenarios where the model may struggle. This analysis can guide targeted data augmentation or model adjustments to reduce these errors.

- **Environmental and Contextual Awareness:** Incorporating additional data sources, such as weather data, topographical maps, and vegetation indices, can provide context to the images and improve the model's accuracy in predicting wildfire risks.
- **Community and Stakeholder Engagement:** Collaborating with environmental agencies, firefighters, and local communities can provide valuable feedback and practical insights that can be used to refine and improve the system. Engaging stakeholders will also help in deploying the system in real-world scenarios effectively.
- **Longitudinal Studies:** Conducting longitudinal studies to monitor the system's performance over time and under different conditions will help in understanding its reliability and areas for improvement.

In conclusion, this project successfully demonstrated the potential of using machine learning and image analysis for wildfire detection. By building on these findings and addressing the recommended future steps, the model can be further refined and deployed to effectively combat the growing threat of wildfires, ultimately contributing to the protection of lives, property, and the environment.

References

[1] H. Jabnouni, I. Arfaoui, M. A. Cherni, M. Bouchouicha and M. Sayadi, "ResNet-50 based fire and smoke images classification," *2022 6th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, Sfax, Tunisia, 2022, Publisher: IEEE

[2] A. Akilandeswari, M. Amanullah, S. Nanthini, R. Sivabalan, T. Thirumalaikumari and R. A, "Comparative Study of Fire Detection Using SqueezeNet and VGG for Enhanced Performance," *2024 Ninth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)*, Chennai, India, 2024, Publisher: IEEE

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS 12), Lake Tahoe, NV, USA, 2012, pp. 1097-1105.

[4] Kaggle, "FlameVision," [Online]. Available: https://www.kaggle.com/code/ismailhamouda/flamevision/notebook

[5] Kaggle, "wildfire-smoke-detection," [Online]. Available: https://www.kaggle.com/datasets/gloryvu/wildfire-smoke-detection

[6] Kaggle, "The wildfire dataset," [Online]. Available: https://www.kaggle.com/datasets/elmadafri/the-wildfire-dataset

[7] Kaggle, "WildFire-Smoke-Dataset-Tensorflow," [Online]. Available: https://www.kaggle.com/datasets/ahemateja19bec1025/wildfiresmokedataset

[8] Kaggle, "wildfire," [Online]. Available: https://www.kaggle.com/code/fatinnibbrashnakib/wildfire

[8] Kaggle, "Wildfire Camera Detection," [Online]. Available: https://www.kaggle.com/code/thestrange007/wildfire-camera-detection?scriptVersionId=180222544

[8] Kaggle, "Fire Classification CNN + MobileNetV2," [Online]. Available: https://www.kaggle.com/code/vencerlanz09/98-fire-classification-cnn-mobilenetv2

Repository

Dataset:
https://drive.google.com/drive/folders/1x3aMwpWnZUngn6W8gMrrvkVdUyGw9g9S?usp=sharing

Improved code:
https://colab.research.google.com/drive/1kEBUwtRogku8CnSTrN0I70EWRV4P35Cy?usp=sharing

Previous code:
https://colab.research.google.com/drive/1u8DKdDPDS_se-_bBIiNAfm3pnNizyKVC?usp=sharing