**Capstone Project Report (For-rest-from-Fires-Fire-Detection-Model-using-Deep-Learning)**

**Model Design, Performance Evaluation, and Future Work**

**Author:** Abhishek A M  
**Date:** 12/08/24

### INTRODUCTION

### Project Overview

The goal of this capstone project was to develop a Convolutional Neural Network (CNN) model for image classification. The dataset used includes images categorized into three classes: Smoke, Fire, and Non-Fire. The primary objective was to create a model that accurately classifies images into these categories, which can be crucial for applications such as fire detection and safety monitoring.

### Objective

The objective of this project was to design, train, and evaluate a CNN model that can effectively classify images into three distinct categories: Smoke, Fire, and Non-Fire. By achieving high accuracy and robustness, the model aims to contribute to improved fire detection systems.

### Approach

To tackle the classification problem, we employed a Convolutional Neural Network (CNN) due to its effectiveness in image recognition tasks. The approach involved pre-processing the images, building the CNN architecture, training the model with data augmentation techniques, and evaluating its performance using various metrics.

## **DATA PREPARATION**

### Dataset Description

The dataset for this project consists of images categorized into three classes: Smoke, Fire, and Non-Fire. The dataset includes a total of 10,500 images, with 3,500 images per class. The images were sourced from a collection that includes various scenarios of smoke, fire, and non-fire conditions, making it well-suited for training a classification model.

### Data Preprocessing

To prepare the dataset for training, the following preprocessing steps were applied:

* **Resizing:** All images were resized to a consistent size of 128x128 pixels to ensure uniform input dimensions for the model.
* **Normalization:** Pixel values were normalized to a range of 0 to 1 by dividing by 255. This helps in speeding up the training and improving model performance.

### Data Augmentation

To enhance the diversity of the training data and improve the model's generalization capability, the following augmentation techniques were used:

* **Rotation:** Images were randomly rotated within a range of 40 degrees.
* **Width and Height Shifts:** Images were shifted horizontally and vertically by up to 20% of their width and height.
* **Shear and Zoom:** Shear and zoom transformations were applied with a range of 20%.
* **Horizontal Flip:** Images were randomly flipped horizontally.
* **Fill Mode:** The fill mode for augmented images was set to 'nearest' to handle newly created pixels during transformations.

## **MODEL ARCHITECTURE AND TRAINING**

### Model Architecture

The model used for this project is a Convolutional Neural Network (CNN) designed to classify images into three categories: Smoke, Fire, and Non-Fire. The architecture of the model is as follows:

1. **Convolutional Layer 1:**
   * **Type:** Conv2D
   * **Filters:** 32
   * **Kernel Size:** (3, 3)
   * **Activation:** ReLU
   * **Input Shape:** (128, 128, 3)
2. **Max-Pooling Layer 1:**
   * **Type:** MaxPooling2D
   * **Pool Size:** (2, 2)
3. **Convolutional Layer 2:**
   * **Type:** Conv2D
   * **Filters:** 64
   * **Kernel Size:** (3, 3)
   * **Activation:** ReLU
4. **Max-Pooling Layer 2:**
   * **Type:** MaxPooling2D
   * **Pool Size:** (2, 2)
5. **Convolutional Layer 3:**
   * **Type:** Conv2D
   * **Filters:** 128
   * **Kernel Size:** (3, 3)
   * **Activation:** ReLU
6. **Max-Pooling Layer 3:**
   * **Type:** MaxPooling2D
   * **Pool Size:** (2, 2)
7. **Flatten Layer:**
   * **Type:** Flatten
8. **Fully Connected Layer 1:**
   * **Type:** Dense
   * **Units:** 512
   * **Activation:** ReLU
   * **Dropout Rate:** 0.5
9. **Output Layer:**
   * **Type:** Dense
   * **Units:** 3
   * **Activation:** Softmax

### Training Process

The model was trained using the following parameters:

* **Epochs:** 10
* **Batch Size:** 32
* **Optimizer:** Adam
* **Learning Rate:** 0.001
* **Loss Function:** Categorical Crossentropy
* **Metrics:** Accuracy

Training was performed with data augmentation to enhance the diversity of the training set.

### Hyperparameters

* **Learning Rate:** 0.001
* **Optimizer:** Adam
* **Dropout Rate:** 0.5 (applied to the fully connected layer)
* **Data Augmentation:** Rotation, width/height shift, shear, zoom, and horizontal flip

## **MODEL EVALUATION AND PERFORMANCE ANALYSIS**

### Model Evaluation

The model was evaluated using a separate test dataset that was not seen by the model during training. The evaluation metrics used include accuracy, loss, and the confusion matrix.

**Test Results:**

* **Test Loss:** 0.1816
* **Test Accuracy:** 0.9435

These results indicate that the model achieved a high accuracy of approximately 94.35% on the test set, showing good generalization to unseen data.

### Confusion Matrix

The confusion matrix provides insight into how well the model performs across different classes. The matrix for the test data is as follows:

|  | **Smoke** | **Fire** | **Non-Fire** |
| --- | --- | --- | --- |
| **Smoke** | 3190 | 64 | 246 |
| **Fire** | 19 | 3412 | 69 |
| **Non-Fire** | 148 | 47 | 3305 |

#### Analysis:

* **Precision, Recall, F1-Score:** The precision, recall, and F1-score for each class are as follows:

| **Class** | **Precision** | **Recall** | **F1-Score** |
| --- | --- | --- | --- |
| **Smoke** | 0.95 | 0.91 | 0.93 |
| **Fire** | 0.97 | 0.97 | 0.97 |
| **Non-Fire** | 0.91 | 0.94 | 0.93 |

* **Accuracy:** The overall accuracy of the model is 94.0%. The metrics indicate strong performance, with particularly high precision and recall for the Fire class.

### Performance Metrics

* **Precision:** Measures the proportion of positive identifications that were actually correct.
* **Recall:** Measures the proportion of actual positives that were correctly identified.
* **F1-Score:** The harmonic mean of precision and recall, providing a single metric that balances both.

### Visualization

The confusion matrix and classification report are visualized to provide a clearer understanding of the model's performance. This helps in identifying which classes are well predicted and which may require further tuning or additional data.

## **DISCUSSION OF FUTURE WORK**

### Improvements and Next Steps

While the current model demonstrates strong performance, there are several areas where improvements can be made:

* **Hyperparameter Tuning:** Further tuning of hyperparameters, such as learning rate, batch size, and the number of epochs, could potentially enhance the model's performance.
* **Architectural Changes:** Exploring different neural network architectures or adding more layers might yield better results. For example, experimenting with architectures like ResNet or Inception could provide improvements.
* **Data Augmentation:** Implementing more advanced data augmentation techniques or increasing the variability in training data could help improve the model's robustness and generalization.

### Model Enhancements

To further enhance the model, consider the following strategies:

* **Transfer Learning:** Utilize pre-trained models and fine-tune them on the current dataset. This approach leverages the learned features from large datasets and can accelerate training while improving performance.
* **Regularization Techniques:** Incorporate techniques such as dropout or L2 regularization to prevent overfitting and improve generalization.
* **Ensemble Methods:** Combine predictions from multiple models to potentially achieve better accuracy and reliability.

### Real-world Applications

The model’s application extends to real-world scenarios such as:

* **Fire Detection Systems:** Implementing the model in fire detection systems for real-time monitoring and alerts.
* **Safety and Surveillance:** Integrating the model into safety and surveillance systems to enhance security measures.

### Challenges and Solutions

During the project, several challenges were encountered:

* **Data Imbalance:** Addressed by using balanced datasets and augmenting data.
* **Model Overfitting:** Mitigated by using regularization techniques and monitoring validation performance.

Future work should focus on addressing these challenges more comprehensively and exploring innovative solutions.

### Additional Research

Potential areas for additional research include:

* **Multi-Modal Data:** Combining image data with other sensor data (e.g., temperature) for more robust predictions.
* **Real-time Processing:** Developing models and systems capable of real-time data processing and decision-making.

## **THE SOURCE CODE USED TO CREATE THE PIPELINE**

Below is the source code used to create and train the model for the project. This code includes data preprocessing, model definition, training, and evaluation steps.

7.1 **Data Preprocessing**

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Directories for training and testing datasets

train\_dir = 'path/to/train'

test\_dir = 'path/to/test'

# Create ImageDataGenerators with data augmentation for training and rescaling for testing

train\_datagen = ImageDataGenerator(

rescale=1./255,

rotation\_range=40,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

fill\_mode='nearest'

)

test\_datagen = ImageDataGenerator(rescale=1./255)

# Create data generators

train\_generator = train\_datagen.flow\_from\_directory(

train\_dir,

target\_size=(128, 128),

batch\_size=32,

class\_mode='categorical',

shuffle=True

)

test\_generator = test\_datagen.flow\_from\_directory(

test\_dir,

target\_size=(128, 128),

batch\_size=32,

class\_mode='categorical'

)

7.2 **Model Definition**

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.optimizers import Adam

# Define the model

model = Sequential([

Conv2D(32, (3, 3), activation='relu', input\_shape=(128, 128, 3)),

MaxPooling2D((2, 2)),

Conv2D(64, (3, 3), activation='relu'),

MaxPooling2D((2, 2)),

Conv2D(128, (3, 3), activation='relu'),

MaxPooling2D((2, 2)),

Flatten(),

Dense(512, activation='relu'),

Dropout(0.5),

Dense(3, activation='softmax')

])

# Compile the model

model.compile(optimizer=Adam(), loss='categorical\_crossentropy', metrics=['accuracy'])

7.3 **Model Training**

# Define steps per epoch and validation steps

train\_steps = len(train\_generator)

test\_steps = len(test\_generator)

# Train the model

history = model.fit(

train\_generator,

steps\_per\_epoch=train\_steps,

epochs=10,

validation\_data=test\_generator,

validation\_steps=test\_steps

)

7.4 **Model Evaluation**

from sklearn.metrics import confusion\_matrix, classification\_report, ConfusionMatrixDisplay

import numpy as np

import matplotlib.pyplot as plt

# Evaluate the model on the test set

test\_loss, test\_accuracy = model.evaluate(test\_generator, steps=test\_steps)

print(f"Test Loss: {test\_loss}")

print(f"Test Accuracy: {test\_accuracy}")

# Predict classes

test\_generator.reset()

predictions = model.predict(test\_generator, steps=test\_steps, verbose=1)

predicted\_classes = np.argmax(predictions, axis=1)

true\_classes = test\_generator.classes

class\_labels = list(test\_generator.class\_indices.keys())

# Compute and display the confusion matrix

cm = confusion\_matrix(true\_classes, predicted\_classes)

disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=class\_labels)

# Plot confusion matrix

fig, ax = plt.subplots(figsize=(10, 8))

disp.plot(ax=ax)

plt.show()

# Print classification report

report = classification\_report(true\_classes, predicted\_classes, target\_names=class\_labels)

print(report)