

FOREST FIRE DETECTION

Applied Data Science

CS22B2009 Srihitha Pulapa
CS22B2017 P.Dhanush
CS22B2019 H.SaiSandeep
CS22B2048 Aditi Gupta

November 11, 2023

Abstract

Forest fires present a substantial hazard to both the environment and communities, underscoring the importance of early detection as an instrumental measure in mitigating their potentially devastating consequences. Within this presentation, we unveil our data science initiative centered around forest fire detection, leveraging sophisticated image processing techniques such as TensorFlow and Keras.

1 Work Distribution :

The collaborative efforts of the team are integral to the various stages of the project. Topic selection, model selection, and the creation and training of the model. is conducted through thorough group discussions, ensuring a comprehensive approach.

1.1 Work Completed by CS22B2009 :

CS22B2009 contributed significantly to the project by focusing on the following key tasks:

- Drafting the Abstract
- Data Collection
- Data Balancing
- Model Evaluation
- Overleaf Presentation

1.2 Work Completed by CS22B2017

The responsibilities undertaken by CS22B2017 encompassed several crucial aspects of the project, including:

- Defining the Project Goal
- Image Resizing
- Conducting Size Analysis of Images
- Data Splitting
- Data Processing

1.3 Work Completed by CS22B2019

The role of CS22B2019 involved critical contributions to the project, including:

- Determining the Project Scope
- Establishing the Background
- Reviewing and Referencing Previous Research Papers on Forest Fire Detection
- Analyzing the Data
- Data Splitting

1.4 Work Completed by CS22B2048

CS22B2048 played a pivotal role in shaping the project's objectives by engaging in the following tasks:

- Defining the Objectives
- Data Collection
- Data Balancing
- Labeling the Data
- Loading the Data Sets

2 Problem Definition and Scope

2.1 Project Goal:

The primary objective is to enhance the timely prediction and detection of forest fire initiation in California, thereby augmenting early detection and response capabilities. This initiative aims to mitigate the risk of forest fire propagation, thereby minimizing potential ecological and environmental damage.

2.2 Scope:

2.2.1 Background:

Forest fires are a significant concern in California due to its vast forested areas, dry climate, and frequent high winds, which create conditions conducive to rapid fire spread. Timely detection is critical for preventing large-scale ecological and environmental damage.

2.2.2 Definition of "Forest Fire" in California

2.2.3 Size

A forest fire is considered significant if it covers an area larger than 10 acres (approximately 4 hectares). This size threshold accounts for California's vast forests and the potential for fires to quickly expand.

2.2.4 Intensity

Flame Height: Flames can range from 1 to 3 meters (approximately 3 to 10 feet) in height.

Rate of Spread: The fire advances at a moderate pace, typically between 100 to 300 meters (approximately 328 to 984 feet) per hour.

Temperature: The fire's temperature can range from 500 to 800 degrees Celsius (932 to 1,472 degrees Fahrenheit).

Smoke Production: The fire generates dense smoke that can reduce visibility and pose health risks.

2.2.5 Duration

For a fire to be classified as a forest fire in California, it should persist for a minimum duration of one hour.

2.3 Objectives

2.3.1 Early Detection

Develop an early warning system to detect forest fires in their incipient stages, improving the response time for firefighting and containment efforts.

2.3.2 Precision and Reliability

Create a system that minimizes false alarms while consistently identifying genuine forest fires, even in challenging environmental conditions.

2.3.3 Environmental Impact Mitigation

Contribute to the mitigation of environmental and ecological damage by providing early and accurate information for rapid response and containment of forest fires.

3 Data Collection

A total of 636 images were gathered from the following online sources:

- Pixabay: <https://pixabay.com/images/search/forest%20fire/>
- Getty Images: <https://www.gettyimages.in/photos/forest-fire>
- Pexels: <https://www.pexels.com/search/forest/>
- iStockphoto: <https://www.istockphoto.com/photos/forest>

It's worth noting that the collected images varied in size. Subsequently, these images have been stored in an unstructured data format within Google Drive.

4 Exploratory Data Analysis (EDA)

The quantity of images categorized under distinct labels is computed. The code snippet below, implemented in Python, assesses the uniformity in the dimensions of images: :

Listing 1: Python Code For checking sizes of images

```
import cv2
import os
image_dir = r"C:\Users\srihitha-pulapa\Desktop\ADS_PROJECT\data_collection"
for filename in os.listdir(image_dir):
    image_path = os.path.join(image_dir, filename)
    img = cv2.imread(image_path)
    height, width, _ = img.shape
    print(f"Image: {filename}, Height: {height}, Width: {width}")
```

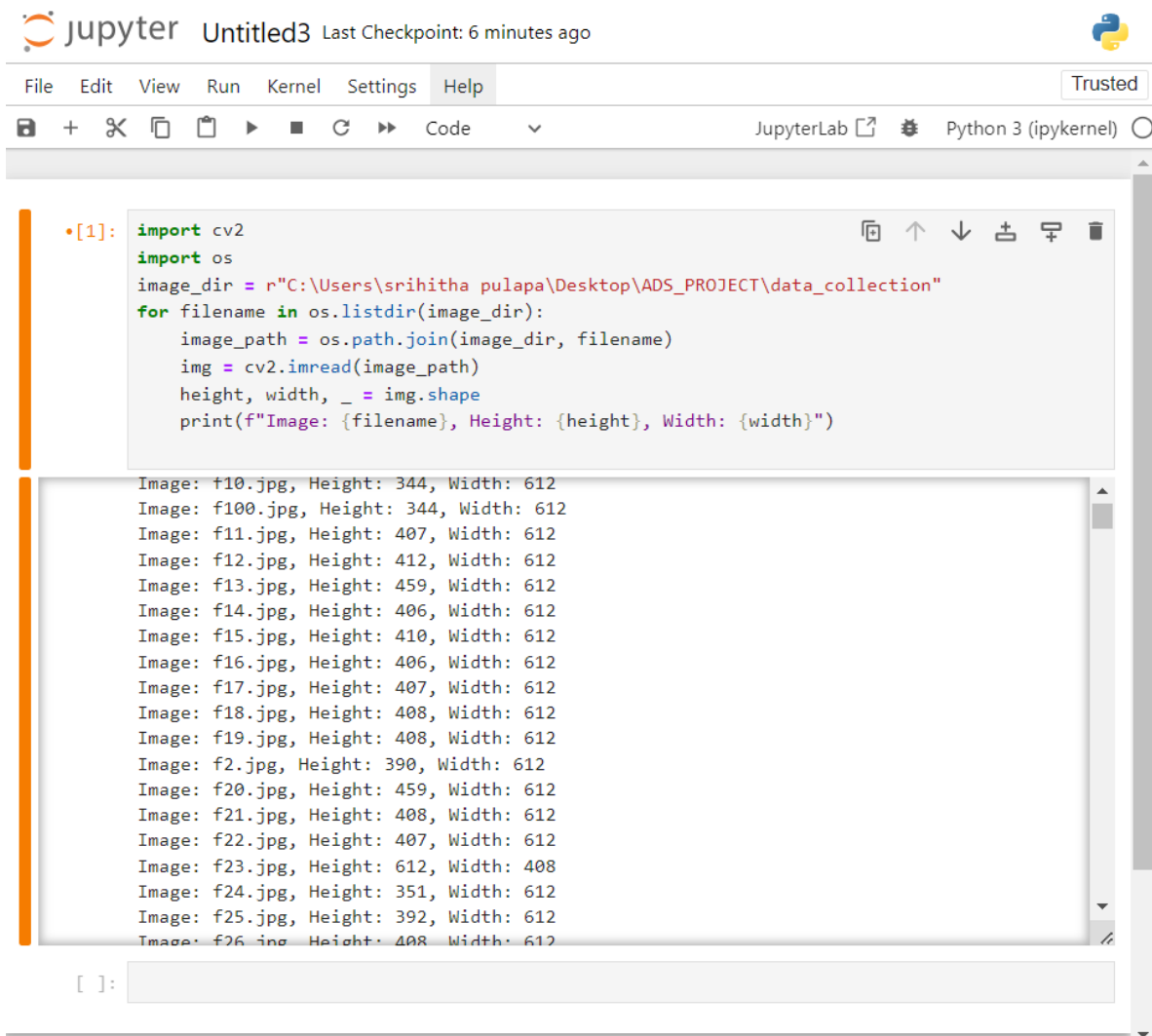
The output is given in the subsequent page

5 Data Pre-Processing

5.1 Data resizing

All the images are resized into a size of 256*256 using an online bulk image resizing tool available at the following URL:

<https://imageresizer.com/bulk-resize>



JupyterLab interface showing a Python script and its output. The script lists image files and their dimensions.

```
[1]: import cv2
import os
image_dir = r"C:\Users\srihitha pulapa\Desktop\ADS_PROJECT\data_collection"
for filename in os.listdir(image_dir):
    image_path = os.path.join(image_dir, filename)
    img = cv2.imread(image_path)
    height, width, _ = img.shape
    print(f"Image: {filename}, Height: {height}, Width: {width}")
```

Output:

```
Image: f10.jpg, Height: 344, Width: 612
Image: f100.jpg, Height: 344, Width: 612
Image: f11.jpg, Height: 407, Width: 612
Image: f12.jpg, Height: 412, Width: 612
Image: f13.jpg, Height: 459, Width: 612
Image: f14.jpg, Height: 406, Width: 612
Image: f15.jpg, Height: 410, Width: 612
Image: f16.jpg, Height: 406, Width: 612
Image: f17.jpg, Height: 407, Width: 612
Image: f18.jpg, Height: 408, Width: 612
Image: f19.jpg, Height: 408, Width: 612
Image: f2.jpg, Height: 390, Width: 612
Image: f20.jpg, Height: 459, Width: 612
Image: f21.jpg, Height: 408, Width: 612
Image: f22.jpg, Height: 407, Width: 612
Image: f23.jpg, Height: 612, Width: 408
Image: f24.jpg, Height: 351, Width: 612
Image: f25.jpg, Height: 392, Width: 612
Image: f26.jpg, Height: 408, Width: 612
```

Figure 1:

5.2 Data Labeling

The dataset has been classified into two distinct labels: Forest Fire Images and Forest Images. There are 436 images categorized under Forest Fire and 200 images categorized under Forest.

5.3 Data Balancing

Upon analysis, an imbalance was identified between Forest Fire images and Forest images. Consequently, we opted to address this imbalance through resampling. Given the already limited size of our dataset, we implemented an oversampling strategy by incorporating 240 new images of forests, appropriately resized. As a result, our revised dataset now comprises 436 images of forest fire and 440 images of forest, achieving a balanced distribution and yielding a total dataset size of 876.

6 Data Splitting

The data has been partitioned into three sets: training set, validation set, and test set. We adhered to a 70-15-15 split for training, testing, and validation sets, respectively. The training set encompasses 70% of the total data, while the remaining data is evenly divided between the testing and validation sets, each comprising 15%. The distribution of images under each label in the three sets is detailed below:

6.1 Training set:

Forest : 308
Forest Fire:305
Total : 613

6.2 Validation Set :

Forest : 66
Forest Fire: 66
Total : 132

6.3 Testing Set :

Forest : 66
Forest Fire:65
Total : 131

7 Model selection

Choosing a Convolutional Neural Network (CNN) for forest fire detection is rooted in its capacity to automatically learn hierarchical features from images. Forest fire images contain critical local patterns, such as the color, texture, and shape of flames or smoke, which CNNs excel at capturing. The translation-invariant nature of CNNs is advantageous for recognizing patterns irrespective of their location in the image, addressing potential variations in fire occurrences. Additionally, CNNs offer adaptability to diverse image conditions and eliminate the need for manual feature engineering.

8 Model Development:

The code for the model is written in python using TensorFlow and Keras Integration because using Python and TensorFlow-Keras integration simplifies machine learning model development with an easy-to-learn language, extensive libraries (NumPy, Pandas). Keras, as a high-level API, is known for its simplicity and ease of use . TensorFlow, a popular deep learning framework, ensures flexibility, scalability, and deployment options. Overall, this combination offers a user-friendly and well-supported environment for efficient model building and deployment.

8.1 Importing the necessary libraries:

Listing 2: Python Code To Import Necessary Libraries

```
import numpy as np
import os
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten
import cv2
import matplotlib.pyplot as plt
```

The numpy library is imported to check the size and resize the images, and to check the shape of images and to reshape them. it is also used to concatenate two files and to normalise pixel values to be between 0 and 1.

The os module is imported to list files in a directory, to join path component for all files, and to check if the files end with a particular extension.

Sequential is imported because The **Sequential** model in Keras allows us to build a neural network layer by layer in a step-by-step fashion.

The **Conv2D** layer in Keras is used to create a convolutional layer in a neural network.

The **MaxPooling2D** layer in Keras is commonly used for down-sampling spatial dimensions.

The **Dense** layer is used for learning non-linear transformations in the data.

The **Flatten** layer is used to flatten the input, transforming it into a one-dimensional array.

OpenCV (cv2) is used for reading and processing images , to resize images to a specified target height and width and Checking if Image Loading is Successful .

8.2 Loading and processing the data sets :

Listing 3: Python Code To Load The Data Sets

```
f_train_path = r"C:\Users\Sandeep\Desktop\ads1\train_1"
f_train = []
for filename in os.listdir(f_train_path):
    if filename.endswith(".jpg") or filename.endswith(".jpeg") or
filename.endswith(".webp"):
    image_path = os.path.join(f_train_path, filename)
    img = cv2.imread(image_path)
    if img is not None:
        f_train.append(img)
    else:
        print(f"Could not load {image_path}")
if not f_train:
    print("f_train not successful")
else:
    print("f_train successful")
f_test_path = r"C:\Users\Sandeep\Desktop\ads1\test_1"
f_test = []
for filename in os.listdir(f_test_path):
    if filename.endswith(".jpg") or filename.endswith(".jpeg") or
filename.endswith(".webp"):
    image_path = os.path.join(f_test_path, filename)
    img = cv2.imread(image_path)
    if img is not None:
        f_test.append(img)
    else:
        print(f"Could not load {image_path}")
if not f_test:
    print("f_test not successful")
else:
```

```

print("f_test-successfull")
nf_train_path = r"C:\Users\Sandeep\Desktop\ads1\train"
nf_train = []
for filename in os.listdir(nf_train_path):
    if filename.endswith(".jpg") or filename.endswith(".jpeg") or
filename.endswith(".webp"):
    image_path = os.path.join(nf_train_path , filename)
    img = cv2.imread(image_path)
    if img is not None:
        nf_train.append(img)
    else:
print(f"Could not load {image_path}")

```

Listing 4: Python Code To Convert Lists into numpy arrays

```

f_train=np.array(f_train)
f_test=np.array(f_test)
nf_train=np.array(nf_train)
nf_test=np.array(nf_test)
if isinstance(f_train , np.ndarray):
    print("f_train-is-a-Numpy-array.")
else:
    print("f_train-is-not-a-Numpy-array.")
if isinstance(f_test , np.ndarray):
    print("f_test-is-a-Numpy-array.")
else:
    print("f_test-is-not-a-Numpy-array.")
if isinstance(nf_train , np.ndarray):
    print("nf_train-is-a-Numpy-array.")
else:
    print("nf_train-is-not-a-Numpy-array.")
if isinstance(nf_test , np.ndarray):
    print("nf_test-is-a-Numpy-array.")
else:
    print("nf_test-is-not-a-Numpy-array.")

```

Listing 5: Python Code To resize the images

```

target_height = 256
target_width = 256
resized_images = []
for img in f_train:
    resized_img = cv2.resize(img, (target_width , target_height))
    resized_images.append(resized_img)
f_train = np.array(resized_images)

```

Listing 6: Python Code To check the shape of images

```

print("shape-of-f_train:-", f_train.shape)
print("shape-of-nf_train:-", nf_train.shape)
print("shape-of-f_test:-", f_test.shape)
print("shape-of-nf_test:-", nf_test.shape)

```

Listing 7: Python Code To Prepare the data and labels

```

X_train = np.concatenate((f_train , nf_train), axis=0)
X_test = np.concatenate((f_test , nf_test), axis=0)
y_train = np.concatenate((np.ones(len(f_train)),
np.zeros(len(nf_train))), axis=0)

```

```
y_test = np.concatenate((np.ones(len(f_test)),
np.zeros(len(nf_test))), axis=0)
```

Listing 8: Python Code To Normalize pixel values to be between 0 and 1

```
X_train = X_train / 255.0
X_test = X_test / 255.0
```

8.3 Creating and Training The Model

Listing 9: Python Code To create and train the model

```
model = Sequential([
    Conv2D(16, (3,3), activation='relu',
input_shape=X_train.shape[1:]),
    MaxPooling2D(2,2),
    Conv2D(32, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, validation_data=(X_test,
y_test))
```

9 Evaluation of the Model

9.1 Code

```
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
print(f'Test accuracy: {test_acc}')
```

9.2 Output

The total accuracy obtained is approximately 0.886 i.e 88.6%

10 References :

- International Journal of Research in Engineering, Science and Management Volume-3, Issue-3, March-2020
- <https://journals.sagepub.com/doi/full/10.1177/1748302619887689>
- <https://journals.sagepub.com/doi/full/10.1177/1748302619887689>
- <https://ieeexplore.ieee.org/document/10125611>


```

Epoch 1/10
20/20 [=====] - 70s 2s/step - loss: 1.6042 -
accuracy: 0.5997 - val_loss: 0.4630 - val_accuracy: 0.7803
Epoch 2/10
20/20 [=====] - 38s 2s/step - loss: 0.4712 -
accuracy: 0.7827 - val_loss: 0.3482 - val_accuracy: 0.8561
Epoch 3/10
20/20 [=====] - 30s 1s/step - loss: 0.3095 -
accuracy: 0.8595 - val_loss: 0.3194 - val_accuracy: 0.9015
Epoch 4/10
20/20 [=====] - 29s 1s/step - loss: 0.2457 -
accuracy: 0.8954 - val_loss: 0.2636 - val_accuracy: 0.8864
Epoch 5/10
20/20 [=====] - 29s 1s/step - loss: 0.1865 -
accuracy: 0.9297 - val_loss: 0.2496 - val_accuracy: 0.8939
Epoch 6/10
20/20 [=====] - 30s 1s/step - loss: 0.1835 -
accuracy: 0.9281 - val_loss: 0.3048 - val_accuracy: 0.8864
Epoch 7/10
20/20 [=====] - 29s 1s/step - loss: 0.1663 -
accuracy: 0.9330 - val_loss: 0.2610 - val_accuracy: 0.8864
Epoch 8/10
20/20 [=====] - 30s 2s/step - loss: 0.1160 -
accuracy: 0.9624 - val_loss: 0.4704 - val_accuracy: 0.8939
Epoch 9/10
20/20 [=====] - 30s 1s/step - loss: 0.1278 -
accuracy: 0.9575 - val_loss: 0.3162 - val_accuracy: 0.9015
Epoch 10/10
20/20 [=====] - 29s 1s/step - loss: 0.0780 -
accuracy: 0.9673 - val_loss: 0.3863 - val_accuracy: 0.8864
5/5 - 1s - loss: 0.3863 - accuracy: 0.8864 - 1s/epoch - 267ms/step
Test accuracy: 0.8863636255264282

```

Figure 2: