Forest Fire Detection using Computer Vision

1. Problem Statement

Imagine you are in charge of watching a big forest. You want to keep everyone safe, but sometimes wildfires can start without warning. These fires can spread quickly and cause a lot of damage. So, we want to build a computer program that can look at pictures of forests and tell us if there's a fire starting or not. This program can help people react faster to stop wildfires!

2. Objectives

This project will:

```
Teach the computer to recognize pictures with wildfires and without wildfires.
Build a model (a smart program) that looks at forest pictures and decides if there's a fire.
Show us a simple screen (a GUI) where we can upload forest pictures and get quick answers on fire danger.
Help keep forests and people safe by giving early warnings about fires.
```

3. Dataset Information

To teach the computer about wildfires, we'll use a special set of pictures called a dataset. This dataset has pictures of forests divided into two types: with a fire (wildfire) and without a fire (no wildfire). These images help the computer learn to tell the difference.

```
Where to Get the Dataset: You can download the dataset from Kaggle
https://www.kaggle.com/datasets/abdelghaniaaba/wildfire-prediction-dataset
Dataset Folders:
    Inside, there are three main folders: train, test, and valid.
    Each folder has two subfolders:
        Wildfire: Contains pictures showing wildfires in forests.
        NoWildfire: Contains pictures showing forests without any fire.
```

4. Understanding the CNN Model

CNN stands for Convolutional Neural Network. It's a special type of program that can look at pictures and learn to find important details. Imagine how you look at a picture and notice trees, animals, or fire — CNN does something similar!

Here's how it works, step-by-step:

```
Looking at Patterns: The CNN looks at lots of tiny patterns in pictures, like shapes and colors.
Learning from Examples: It looks at many images of wildfires and no wildfires to learn what makes them different.
Making Predictions: Once trained, the CNN can look at a new picture and guess if it shows a fire or not.
```

5. Code Implementation

```
# Import libraries
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk, ImageFile
import numpy as np
import os
```

```
# This line lets TensorFlow ignore corrupted images
ImageFile.LOAD_TRUNCATED_IMAGES = True

# Paths to dataset folders
train_dir = r"C:\Users\user\Desktop\Edunet_Content_Usha\Computer Vision\Datasets\Forest fire classification data\train"
valid_dir = r"C:\Users\user\Desktop\Edunet_Content_Usha\Computer Vision\Datasets\Forest fire classification data\valid"
test_dir = r"C:\Users\user\Desktop\Edunet_Content_Usha\Computer Vision\Datasets\Forest fire classification data\test"


# Loading and preprocessing images with ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255)
valid_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir, target_size=(64, 64), batch_size=32, class_mode='binary'
)
valid_generator = valid_datagen.flow_from_directory(
    valid_dir, target_size=(64, 64), batch_size=32, class_mode='binary'
)

# Building the CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')  # Output layer for binary classification
])

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

# Training the model
history = model.fit(train_generator, validation_data=valid_generator, epochs=10, verbose=1)

# GUI for predicting an image
def predict_image():
    # Open file dialog to select an image
    file_path = filedialog.askopenfilename()
    if file_path:
        # Display the image in the GUI
        img = Image.open(file_path).resize((200, 200))
        img_tk = ImageTk.PhotoImage(img)
        image_label.configure(image=img_tk)
        image_label.image = img_tk

        # Preprocess the image for the model
        img_for_model = Image.open(file_path).resize((64, 64))
        img_array = np.array(img_for_model) / 255.0
        img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension

        # Make a prediction
        prediction = model.predict(img_array)[0][0]
        result = "Wildfire" if prediction > 0.5 else "No Wildfire"
        result_label.config(text="Prediction: " + result)

# Setting up the GUI window
root = tk.Tk()
root.title("Forest Fire Detection")
root.geometry("400x400")

# Adding widgets
btn = tk.Button(root, text="Upload Image", command=predict_image)
btn.pack(pady=20)

image_label = tk.Label(root)
image_label.pack()

result_label = tk.Label(root, text="Prediction: ", font=("Helvetica", 16))
result_label.pack(pady=20)

# Start the GUI event loop
root.mainloop()
```

```
Found 30250 images belonging to 2 classes.
Found 6300 images belonging to 2 classes.
Epoch 1/10
946/946 [==============================] - 283s 297ms/step - loss: 0.2306 - accuracy: 0.9102 - val_loss: 0.1583 - val_accuracy: 0.9443
Epoch 2/10
946/946 [==============================] - 286s 303ms/step - loss: 0.1777 - accuracy: 0.9348 - val_loss: 0.1888 - val_accuracy: 0.9195
Epoch 3/10
946/946 [==============================] - 282s 298ms/step - loss: 0.1619 - accuracy: 0.9408 - val_loss: 0.1253 - val_accuracy: 0.9546
Epoch 4/10
946/946 [==============================] - 286s 302ms/step - loss: 0.1461 - accuracy: 0.9470 - val_loss: 0.1413 - val_accuracy: 0.9495
Epoch 5/10
946/946 [==============================] - 285s 301ms/step - loss: 0.1330 - accuracy: 0.9501 - val_loss: 0.1140 - val_accuracy: 0.9576
Epoch 6/10
946/946 [==============================] - 288s 305ms/step - loss: 0.1200 - accuracy: 0.9570 - val_loss: 0.1182 - val_accuracy: 0.9600
Epoch 7/10
946/946 [==============================] - 337s 356ms/step - loss: 0.1110 - accuracy: 0.9590 - val_loss: 0.1112 - val_accuracy: 0.9595
Epoch 8/10
946/946 [==============================] - 239s 252ms/step - loss: 0.0963 - accuracy: 0.9641 - val_loss: 0.1075 - val_accuracy: 0.9624
Epoch 9/10
946/946 [==============================] - 221s 234ms/step - loss: 0.0851 - accuracy: 0.9689 - val_loss: 0.1020 - val_accuracy: 0.9625
Epoch 10/10
946/946 [==============================] - 240s 254ms/step - loss: 0.0764 - accuracy: 0.9718 - val_loss: 0.1358 - val_accuracy: 0.9543
1/1 [==============================] - 0s 221ms/step
```

5. Conclusion

With this project, we have created a simple computer program that helps spot wildfires early by looking at pictures of forests. This project shows how computer vision, a field that teaches computers to see and understand images, can help us make safer decisions for people and nature. Using CNN, we taught the computer to recognize fires, giving us a way to respond faster and prevent fires from spreading.