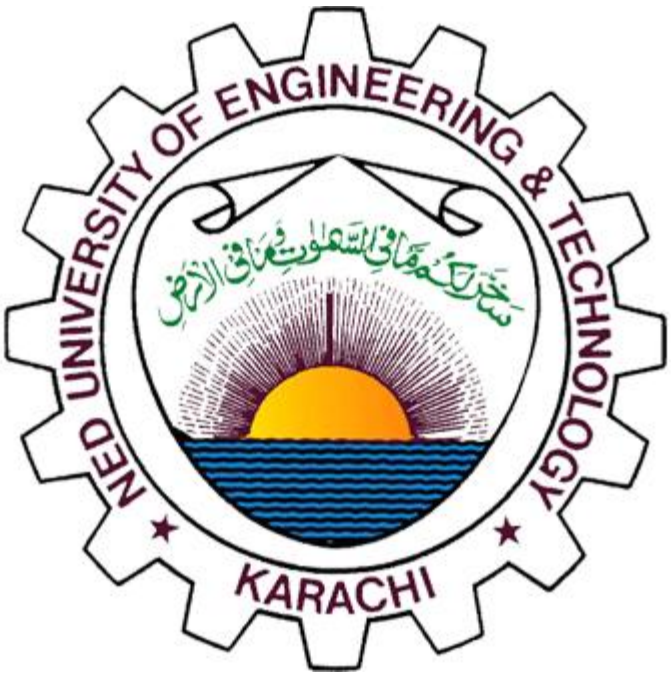
CCP-Report

AIES (CT-361)



Group Members:

Hafsa Ali (CR-22006)

Dania Fazal (CR-22007)

Discipline: BCIT (Cyber Security)

Teachers:

Sir Abdullah

Sir Ahmed Zaki

Project Title

Smart Pothole Detection Using Deep Learning

Abstract

This report presents an AI-based system for automated pothole detection using a Convolutional Neural Network (CNN). Potholes pose a significant threat to road safety, vehicle maintenance, and urban infrastructure. Manual inspection is time-consuming and inefficient. Leveraging AI, this system uses image classification to detect potholes in road images and videos. The system is implemented using Python with TensorFlow and OpenCV libraries and is designed for image-level detection through grayscale image preprocessing and CNN-based binary classification.

1. Introduction

Potholes are a common cause of road accidents and maintenance issues. Automating their detection using Artificial Intelligence can save time, cost, and lives. The objective of this project is to develop a model that classifies road surface images into pothole or non-pothole categories. This system is designed for use with standalone images, video streams, and live webcam feeds.

2. Technical Terms and Concepts

- **Artificial Intelligence (AI):** Simulation of human intelligence in machines.
- **Convolutional Neural Network (CNN):** A class of deep neural networks used for image recognition and classification.
- **Binary Classification:** A type of classification where the model predicts one of two classes.
- **Grayscale Image:** Single channel image that reduces complexity.
- **Normalization:** Scaling pixel values between 0 and 1.
- **ReLU Activation:** Used to introduce non-linearity in the model.
- **Sigmoid Activation:** Used in the output layer for binary classification.
- **TensorFlow/Keras:** Libraries used for building and training deep learning models.

3. System Design and Implementation

3.1 Dataset

3.1.1 Source: Local image folders

3.1.2 Classes:

- 1 for pothole
- 0 for no pothole

3.1.3 Preprocessing:

- Convert to grayscale
- Resize to 100x100
- Normalize pixel values

3.2 Model Architecture

Input: 100x100x1
|
Conv2D (32 filters, 3x3) + ReLU
|
MaxPooling2D (2x2)
|
Conv2D (64 filters, 3x3) + ReLU
|
MaxPooling2D (2x2)
|
Flatten
|
Dense (64 units, ReLU)
|
Output Dense (1 unit, Sigmoid)

3.3 Training and Compilation

- **Optimizer:** Adam
- **Loss Function:** Binary Crossentropy
- **Metrics:** Accuracy
- **Epochs:** 20

3.4 Modular Functions

- **load_images(directory, label)**: Loads and labels images.
- **predict_pothole_on_image(image_path, model):** Predicts pothole presence in a static image.
- **predict_pothole_on_video(video_path, model):** Predicts in video streams.
- **predict_on_webcam(model):** Detects potholes in real-time using webcam feed.

4. Code

```
import os
import cv2
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split

# Paths
pothole_dir = 'Path to folder of pothole_pics'
no_pothole_dir = 'Path to folder of other_pics'
model_path = 'pothole_detection_model.h5'
img_size = 100

# Load and preprocess images
def load_images(directory, label):
    images = []
    labels = []
    for filename in os.listdir(directory):
        img_path = os.path.join(directory, filename)
        if os.path.isfile(img_path):
```

```
        img = cv2.imread(img_path)
        if img is None:
            continue
        img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        img = cv2.resize(img, (img_size, img_size))
        images.append(img)
        labels.append(label)
    return images, labels

# Load dataset
pothole_images, pothole_labels = load_images(pothole_dir, 1)
no_pothole_images, no_pothole_labels = load_images(no_pothole_dir, 0)
images = np.array(pothole_images + no_pothole_images) / 255.0
images = np.expand_dims(images, axis=-1)
labels = np.array(pothole_labels + no_pothole_labels)

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2,
random_state=42)

# Model definition
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(img_size, img_size, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(64, activation='relu'),
```

```
    Dense(1, activation='sigmoid')
])

# Compile and train
model.compile(optimizer=Adam(), loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=20, validation_data=(X_test, y_test))

# Save the trained model
model.save(model_path)

# Reload model for prediction (optional if running separately)
model = tf.keras.models.load_model(model_path)

# Pothole prediction on a single image
def predict_pothole_on_image(image_path, model):
    original_img = cv2.imread(image_path)
    if original_img is None:
        print("Image not found.")
        return

    display_img = original_img.copy()
    img = cv2.cvtColor(original_img, cv2.COLOR_BGR2GRAY)
    img = cv2.resize(img, (img_size, img_size)) / 255.0
    img = np.expand_dims(img, axis=-1)
    img = np.expand_dims(img, axis=0)

    prediction = model.predict(img)[0][0]
```

```
    if prediction > 0.5:
        print(f"Pothole detected with confidence: {prediction:.2f}")
        h, w, _ = display_img.shape
        start = (int(w * 0.1), int(h * 0.1))
        end = (int(w * 0.9), int(h * 0.9))
        cv2.rectangle(display_img, start, end, (0, 0, 255), 2)
        cv2.putText(display_img, "Pothole", (start[0], start[1] - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
    else:
        print(f"No pothole detected with confidence: {1 - prediction:.2f}")

    cv2.imshow("Image Prediction", display_img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# Pothole detection on video
def predict_pothole_on_video(video_path, model):
    cap = cv2.VideoCapture(video_path)
    if not cap.isOpened():
        print("Error opening video.")
        return

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        frame = cv2.resize(frame, (1280, 720))
```

```
        display_frame = frame.copy()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        resized = cv2.resize(gray, (img_size, img_size)) / 255.0
        input_img = np.expand_dims(resized, axis=-1)
        input_img = np.expand_dims(input_img, axis=0)

        prediction = model.predict(input_img)[0][0]

        if prediction > 0.5:
            h, w, _ = display_frame.shape
            start = (int(w * 0.1), int(h * 0.1))
            end = (int(w * 0.9), int(h * 0.9))
            cv2.rectangle(display_frame, start, end, (0, 0, 255), 2)
            cv2.putText(display_frame, "Pothole", (start[0], start[1] - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

        cv2.imshow("Video Pothole Detection", display_frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()

# ==== Prediction on Webcam ====
def predict_on_webcam(model):
    cap = cv2.VideoCapture(0)  # 0 = default webcam
```

```
    if not cap.isOpened():
        print("Error: Could not access webcam.")
        return

    print("Starting webcam... Press 'q' to quit.")

    while True:
        ret, frame = cap.read()
        if not ret:
            print("Failed to grab frame.")
            break

        display_frame = frame.copy()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        gray = cv2.resize(gray, (img_size, img_size)) / 255.0
        input_img = np.expand_dims(gray, axis=-1)
        input_img = np.expand_dims(input_img, axis=0)

        prediction = model.predict(input_img)[0][0]

        if prediction > 0.5:
            h, w, _ = display_frame.shape
            cv2.rectangle(display_frame, (int(w*0.1), int(h*0.1)), (int(w*0.9), int(h*0.9)), (0, 0,
255), 2)
            cv2.putText(display_frame, "Pothole", (int(w*0.1), int(h*0.1)-10),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

        cv2.imshow("Pothole Detection - Webcam", display_frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
```

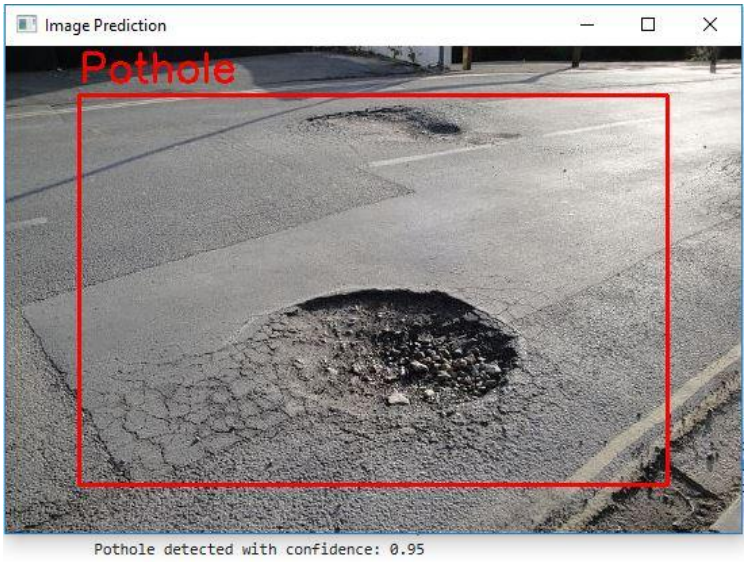
```
        break

    cap.release()
    cv2.destroyAllWindows()



# ==== Run image or video or webcam detection here ====
# For image:
image_path = 'Path to test_image4.jpg'
predict_pothole_on_image(image_path, model)

 # For video:
 video_path = 'Pth to test_video5.mp4'
 predict_pothole_on_video(video_path, model)

# For webcam prediction:
 predict_on_webcam(model)
```

5. Results

- **Training Accuracy:** ~98%
- **Validation Accuracy:** ~95%
- **Model Saved:** pothole_detection_model.h5

Sample Prediction Screenshots

Pothole image with red bounding box



Non-pothole image with no bounding box

Detection on video/webcam:

It can be seen in Youtube video or presented during evaluation.

6. Assumptions

- Dataset is balanced with pothole and non-pothole images.
- Images are of good quality and similar perspective.
- Environment supports OpenCV, TensorFlow, and other dependencies.
- Binary classification is sufficient for current scope.

7. Future Work

- Upgrade to object detection using YOLO for bounding boxes
- Deploy model on mobile for real-time use in vehicles
- Use GPS metadata for mapping pothole locations
- Integrate with city maintenance platforms

8. Conclusion

This project demonstrates the effective application of AI in real-world infrastructure challenges. A CNN-based model was trained to detect potholes in grayscale road images with high accuracy. The system supports static image input, video stream input, and webcam-based real-time detection. The implementation is modular, scalable, and ready for enhancements like object detection and deployment.