Automatic Number Plate Recognition (ANPR) using CNN

Authors:
Urmi Ganguly

Project Objective:
The aim of this project is to develop an Automatic Number Plate Recognition (ANPR) system using Convolutional Neural Networks (CNN) and OpenCV. The system detects vehicle number plates from an image, segments individual characters, and recognize them using a trained CNN model.

Tools & Technologies Used

- Python 3.x
- TensorFlow / Keras
- OpenCV
- NumPy, Matplotlib

Workflow & Pipeline

1. Data Input: User provides paths for training/validation datasets organized by character folders.

2. Preprocessing: Resize images, convert to grayscale, normalize.

3. Model Architecture: CNN with Conv2D, MaxPooling2D, Dense, Dropout layers.

4. Training: CNN trained on character data and saved in .keras format.

5. License Plate Detection: OpenCV contour detection isolates the plate.

6. Character Segmentation: Thresholding and morphology extract individual characters.

7. Character Recognition: CNN classifies each character, builds the plate string.

8. Output: Plate number printed in console.

Code

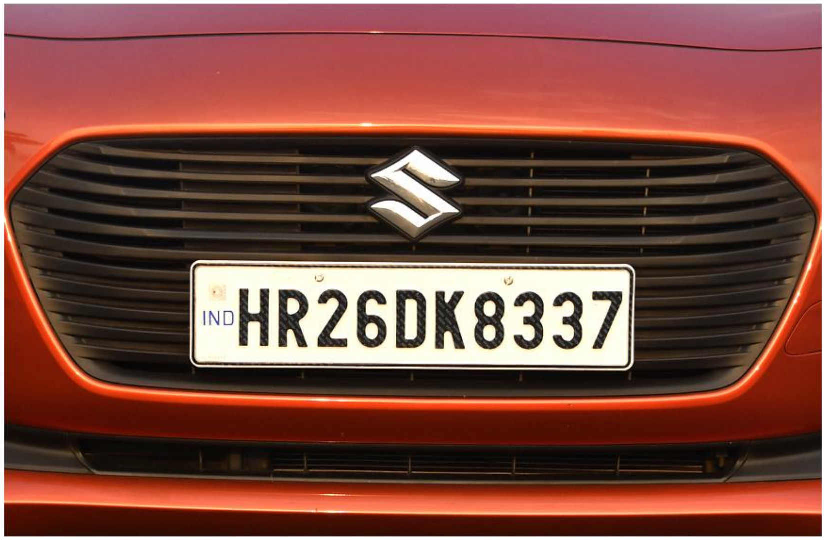
```
# Automatic Number Plate Recognition (ANPR) using CNN
# Author: Urmi Ganguly
# Description: This script performs number plate recognition using OpenCV and a custom CNN.

import os
import cv2
import numpy as np
import matplotlib.pyplot as plt

# -------------------------------
# TensorFlow Import with Error Handling
# -------------------------------
try:
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    from tensorflow.keras.models import load_model
    from tensorflow.keras.preprocessing.image import img_to_array
except ImportError as e:
    print("TensorFlow is not installed or not configured correctly.")
    print("Error:", e)
    print("Please ensure TensorFlow is installed and compatible with your Python version.")
    exit(1)

# -------------------------------
# 1. User Input for Paths
# -------------------------------
try:
    train_dir = input("Enter training data path: ").strip().strip('"').strip("'")
    val_dir = input("Enter validation data path: ").strip().strip('"').strip("'")
except Exception as e:
    print("Error reading input paths:", e)
    exit(1)

# -------------------------------
# 2. Image Generators
# -------------------------------
img_width, img_height = 50, 50
batch_size = 32

try:
    datagen = ImageDataGenerator(rescale=1./255)
    train_generator = datagen.flow_from_directory(
        train_dir,
        target_size=(img_width, img_height),
        color_mode='grayscale',
        batch_size=batch_size,
        class_mode='categorical')

    val_generator = datagen.flow_from_directory(
        val_dir,
        target_size=(img_width, img_height),
        color_mode='grayscale',
        batch_size=batch_size,
        class_mode='categorical')
except Exception as e:
    print("Error loading image data:", e)
    exit(1)

# -------------------------------
# 3. CNN Model
# -------------------------------
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(img_width, img_height, 1)),
    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(len(train_generator.class_indices), activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()

# -------------------------------
# 4. Model Training
# -------------------------------
epochs = 10
model.fit(train_generator, validation_data=val_generator, epochs=epochs)
model.save("character_recognition_cnn.keras", save_format="keras")

# -------------------------------
# 5. Plate Detection
# -------------------------------
```

```
def detect_plate(image_path):
    image = cv2.imread(image_path)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    filtered = cv2.bilateralFilter(gray, 11, 17, 17)
    edged = cv2.Canny(filtered, 30, 200)
    contours, _ = cv2.findContours(edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10]

    for c in contours:
        peri = cv2.arcLength(c, True)
        approx = cv2.approxPolyDP(c, 0.018 * peri, True)
        if len(approx) == 4:
            x, y, w, h = cv2.boundingRect(approx)
            plate = image[y:y+h, x:x+w]
            cv2.imshow("Detected Plate", plate)
            cv2.waitKey(0)
            cv2.destroyAllWindows()
            return plate
    return None

# -------------------------------
# 6. Character Segmentation
# -------------------------------
def segment_characters(plate_img):
    gray = cv2.cvtColor(plate_img, cv2.COLOR_BGR2GRAY)
    _, thresh = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY_INV)
    contours, _ = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    characters = []
    for i, cnt in enumerate(sorted(contours, key=lambda ctr: cv2.boundingRect(ctr)[0])):
        x, y, w, h = cv2.boundingRect(cnt)
        if h / plate_img.shape[0] >= 0.5 and w / plate_img.shape[1] <= 0.5:
            char = thresh[y:y+h, x:x+w]
            resized = cv2.resize(char, (img_width, img_height))
            characters.append(resized)

            # Debug display
            cv2.imshow(f"Character {i+1}", resized)
            cv2.waitKey(0)
            cv2.destroyAllWindows()
    return characters

# -------------------------------
# 7. Character Recognition
# -------------------------------
def recognize_characters(characters, model, class_labels):
    result = ""
    for char in characters:
        char = char.astype("float32") / 255.0
        char = np.expand_dims(char, axis=-1)
        char = np.expand_dims(char, axis=0)
        pred = model.predict(char, verbose=0)
        label = class_labels[np.argmax(pred)]
        result += label
    return result

# -------------------------------
# 8. Predict Plate Number
# -------------------------------
def predict_plate_number(image_path, model, class_indices):
    plate_img = detect_plate(image_path)
    if plate_img is not None:
        chars = segment_characters(plate_img)
        labels_map = {v: k.split("_")[1] for k, v in class_indices.items()}
        return recognize_characters(chars, model, labels_map)
    else:
        return "Plate not detected"

# -------------------------------
# 9. User Testing
# -------------------------------
test_image_path = input("Enter path to a test image: ").strip().strip('"').strip("'")

if os.path.exists(test_image_path):
    model = load_model("character_recognition_cnn.keras")
    predicted_plate = predict_plate_number(test_image_path, model, train_generator.class_indices)
    print("Detected Plate Number:", predicted_plate)
else:
    print("Test image path is invalid or does not exist.")
```

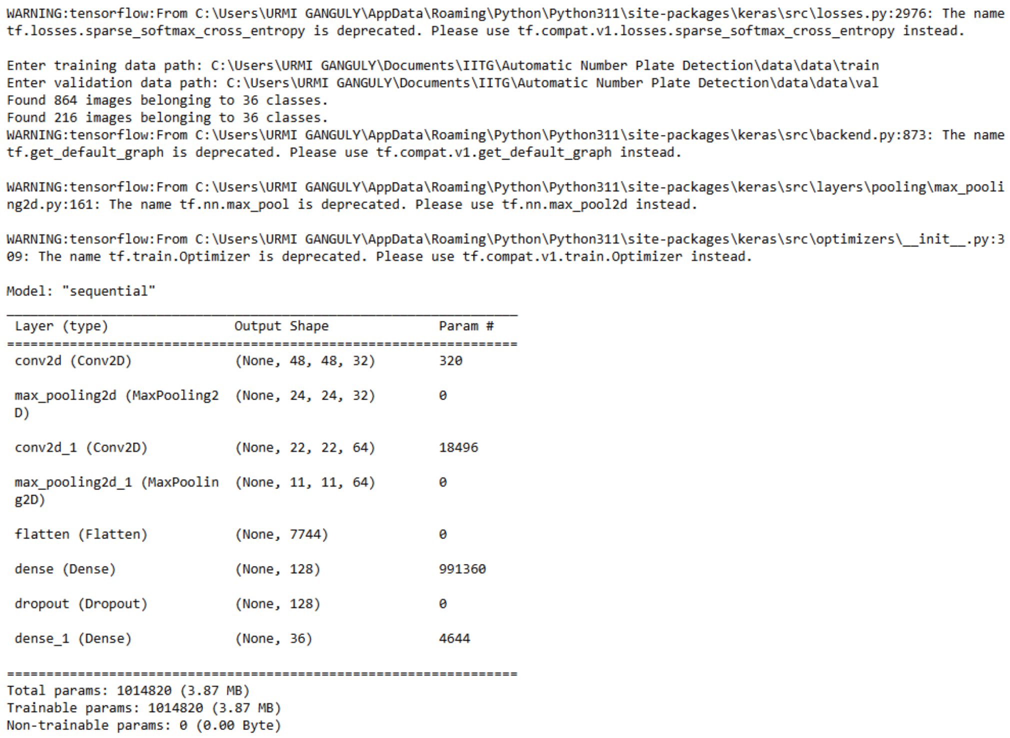
Dataset

Synthetic dataset of grayscale character images (A-Z, 0-9) structured into labeled folders for training and validation.

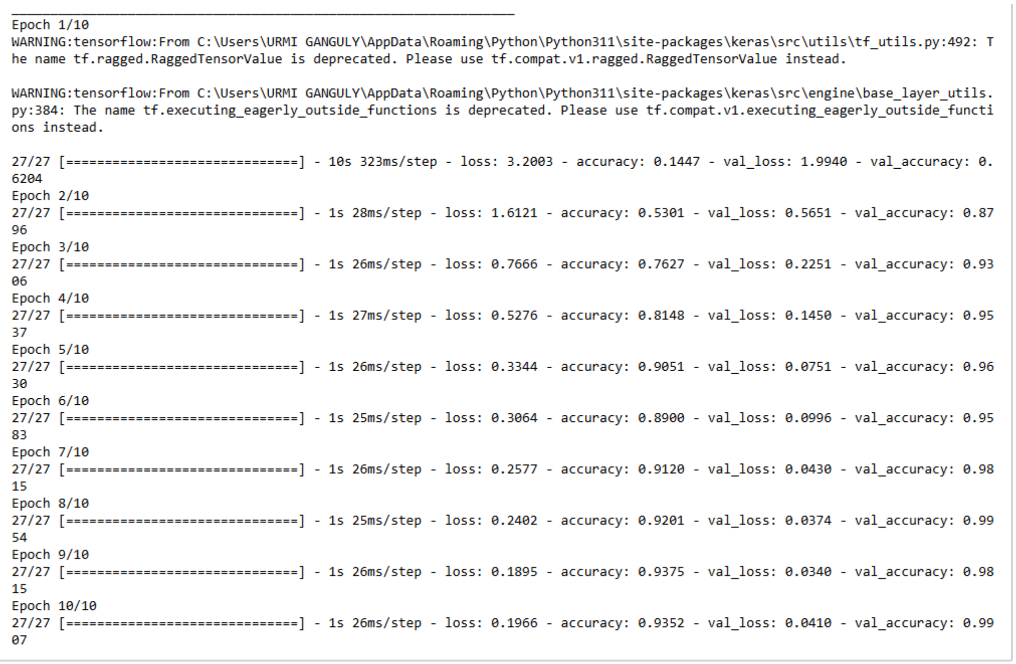
Input Sample:



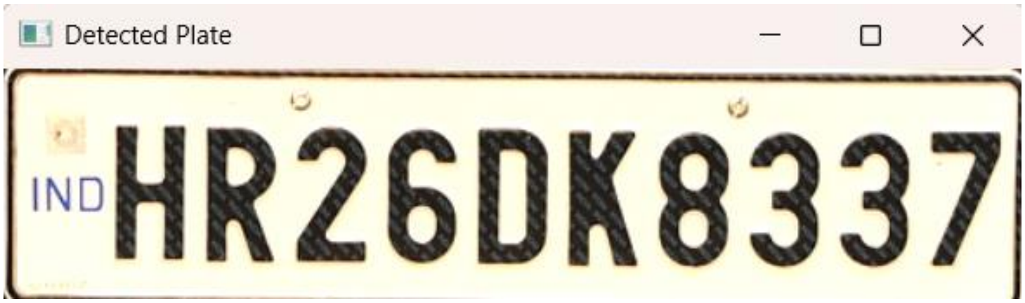
Output Sample:



```
WARNING:tensorflow:From C:\Users\URMI GANGULY\AppData\Roaming\Python\Python311\site-packages\keras\src\losses.py:2976: The name
tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

Enter training data path: C:\Users\URMI GANGULY\Documents\IITG\Automatic Number Plate Detection\data\data\train
Enter validation data path: C:\Users\URMI GANGULY\Documents\IITG\Automatic Number Plate Detection\data\data\val
Found 864 images belonging to 36 classes.
Found 216 images belonging to 36 classes.
WARNING:tensorflow:From C:\Users\URMI GANGULY\AppData\Roaming\Python\Python311\site-packages\keras\src\backend.py:873: The name
tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\URMI GANGULY\AppData\Roaming\Python\Python311\site-packages\keras\src\layers\pooling\max_pooli
ng2d.py:161: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From C:\Users\URMI GANGULY\AppData\Roaming\Python\Python311\site-packages\keras\src\optimizers\__init__.py:3
09: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 48, 48, 32)        320

 max_pooling2d (MaxPooling2  (None, 24, 24, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 22, 22, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 11, 11, 64)        0
 g2D)

 flatten (Flatten)           (None, 7744)              0

 dense (Dense)               (None, 128)               991360

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 36)                4644

=================================================================
Total params: 1014820 (3.87 MB)
Trainable params: 1014820 (3.87 MB)
Non-trainable params: 0 (0.00 Byte)
```



```
Epoch 1/10
WARNING:tensorflow:From C:\Users\URMI GANGULY\AppData\Roaming\Python\Python311\site-packages\keras\src\utils\tf_utils.py:492: T
he name tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\URMI GANGULY\AppData\Roaming\Python\Python311\site-packages\keras\src\engine\base_layer_utils.
py:384: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functi
ons instead.

27/27 [==============================] - 10s 323ms/step - loss: 3.2003 - accuracy: 0.1447 - val_loss: 1.9940 - val_accuracy: 0.
6204
Epoch 2/10
27/27 [==============================] - 1s 28ms/step - loss: 1.6121 - accuracy: 0.5301 - val_loss: 0.5651 - val_accuracy: 0.87
96
Epoch 3/10
27/27 [==============================] - 1s 26ms/step - loss: 0.7666 - accuracy: 0.7627 - val_loss: 0.2251 - val_accuracy: 0.93
06
Epoch 4/10
27/27 [==============================] - 1s 27ms/step - loss: 0.5276 - accuracy: 0.8148 - val_loss: 0.1450 - val_accuracy: 0.95
37
Epoch 5/10
27/27 [==============================] - 1s 26ms/step - loss: 0.3344 - accuracy: 0.9051 - val_loss: 0.0751 - val_accuracy: 0.96
30
Epoch 6/10
27/27 [==============================] - 1s 25ms/step - loss: 0.3064 - accuracy: 0.8900 - val_loss: 0.0996 - val_accuracy: 0.95
83
Epoch 7/10
27/27 [==============================] - 1s 26ms/step - loss: 0.2577 - accuracy: 0.9120 - val_loss: 0.0430 - val_accuracy: 0.98
15
Epoch 8/10
27/27 [==============================] - 1s 25ms/step - loss: 0.2402 - accuracy: 0.9201 - val_loss: 0.0374 - val_accuracy: 0.99
54
Epoch 9/10
27/27 [==============================] - 1s 26ms/step - loss: 0.1895 - accuracy: 0.9375 - val_loss: 0.0340 - val_accuracy: 0.98
15
Epoch 10/10
27/27 [==============================] - 1s 26ms/step - loss: 0.1966 - accuracy: 0.9352 - val_loss: 0.0410 - val_accuracy: 0.99
07
```



Limitations & Future Scope

- Works best with clear, front-facing plates
- Sensitive to image quality

Conclusion

This project successfully demonstrates the core functionality of a number plate recognition system using deep learning and computer vision. It highlights the integration of CNNs for character recognition and OpenCV for image processing, providing a hands-on understanding of practical AI applications.