Traffic Sign Detection System using Convolutional Neural Network and Computer Vision

Nathanael Adrian T. Cua, Kristina Diana L. Ocampo, Cecilia Angeline J. Roque, Daniel Iñigo M. Soriano

Department of Electronics and Computer Engineering
Gokongwei College of Engineering
De La Salle University Manila
2401 Taft Avenue, Malate, Manila, Philippines 1004
nathanael_cua@dlsu.edu.ph
cecilia_roque@dlsu.edu.ph
kristina_ocampo@dlsu.edu.ph
daniel_inigo_soriano@dlsu.edu.ph

***Abstract* - Traffic Sign Recognition Systems are important for enhancing road safety and supporting autonomous vehicle navigation. This study focuses on designing and implementing a real-time TSRS using computer vision techniques. Developed using OpenCV and Convolutional Neural Networks (CNNs), the system detects and classifies traffic signs such as Stop, Pedestrian Crossing, and Parking signs under dynamic environmental conditions. The program utilizes preprocessing, feature extraction, and classification stages, culminating in a user-friendly interface that highlights recognized signs. Testing reveals the system's robustness, accuracy, and potential for real-world applications, contributing to safer roads, autonomous vehicle development, and advanced urban traffic management. With the different trials done, the developed neural network was able to accurately detect the different signages with various conditions such as different style, lighting, distance and angle.**

Index Terms - Road Signs, Sign Detection, Sign Recognition, Computer Vision

I. INTRODUCTION

Traffic sign recognition systems (TSRS) play important roles in road safety by aiding drivers and autonomous navigation systems of vehicles. These systems help ensure that drivers are aware of important road signs, even if they miss them, thereby reducing the risk of accidents and improving overall traffic management. The growth of powerful and efficient image processing tools like OpenCV has enabled the creation of robust, real time image recognition [1]. These systems utilize live camera feeds to detect and classify various road signs under dynamic environmental conditions, such as varying lighting, motion blur, and occlusion [2]. Furthermore, the implementation of TSRS aligns with global efforts to standardize traffic signs and improve road safety. For instance, the Vienna Convention on Road Signs and Signals has established standardized traffic signs across many countries, facilitating the development of TSRS that can be used internationally. From this, TSRS systems contribute to the unification of traffic signs internationally.

II. OBJECTIVES

The primary objective of this project are as follows:

To design and implement a traffic sign recognition system that utilizes computer vision to assist in autonomous vehicle navigation and advanced driver assistance through real time detection and classification of traffic signs.

To further break down the objectives, the group formulated the following specific objectives:

1. To develop a robust real-time traffic sign recognition system using a live camera feed.
2. To leverage OpenCV functions for traffic sign detection and classification.
3. To enhance the accuracy of recognition by addressing environmental factors such as lighting, motion blur, and occlusion.
4. To provide visual feedback by highlighting detected signs in the application window for real-time usability.

III. DESIGN OF THE PROJECT

A. Theoretical Background

1. Image Preprocessing: Techniques to prepare images for analysis, including converting to grayscale (reducing complexity) and applying Gaussian blur (smoothing and noise reduction).
2. Edge Detection: Identifies boundaries of objects in an image by detecting areas of significant intensity changes, typically using the Canny algorithm.
3. Contour Detection: Finds continuous curves around objects in an image, which help isolate and define the region of interest (e.g., traffic signs).
4. Convolutional Neural Network (CNN): A deep learning model designed for image classification that learns spatial hierarchies of features through convolution and pooling layers.

5. Convolutional Layers: Layers in a CNN that apply filters to input images, detecting basic features like edges and textures.
6. Pooling Layers: Reduce the dimensionality of feature maps in a CNN, making computations more efficient while retaining essential features.
7. Softmax Activation: A function used in the output layer of a neural network that converts raw outputs into probabilities for each class.
8. L2 Regularization: A technique to prevent overfitting by penalizing large weights in a neural network.

B. System Architecture

The system design involves multiple stages to ensure accurate and efficient traffic sign recognition:

1. Input Capture: The live camera feed is continuously monitored using *cv2.VideoCapture* to acquire frames.

2. Preprocessing: The frames undergo preprocessing steps, including resizing, noise reduction, and conversion to a suitable color space. The program uses grayscale conversion (cv2.COLOR_BGR2GRAY) and Gaussian blur (cv2.GaussianBlur), as well as edge detection (cv2.Canny) to complete the preprocessing stage.

3. Feature Detection: Traffic signs are detected using shape-based (e.g., circular, triangular, rectangular) and color-based segmentation techniques. The main function used for the shape detection is *cv2.findContours* to find the most prominent contoured area on the frame.

4. Classification: Detected features are matched against a trained classifier, *road_sign_model_updated.h5,* to identify specific traffic signs. The model is trained using a dataset of pictures of the same traffic sign. *model.predict(reshaped)* is then used to classify and detect the traffic signs that match the dataset.

5. Output Visualization: Recognized signs are detected in the live feed with a text output, providing an easy-to-use interface. The function *cv2.putText* is used to output the text corresponding to the sign detected.

IV. Codes and Explanation

The program developed is a Python-based application integrating image processing techniques such as Gaussian Blur, Canny Edge Detection, and Contour Analysis. Provided that this project also utilizes machine learning in order to accurately detect and identify particular signages, it is crucial that the system would create and make use of a model. For its results to achieve the set objectives, the model must be thoroughly trained. In order to do so, a dataset is required which consists of a diverse set of images for each category (Stop Sign, Ped Xing Sign, and Parking Sign). These pictures are carefully organized into their respective folders for easy access for the program. This project followed the structure displayed in Figure 1 to ensure proper labelling is adhered to.

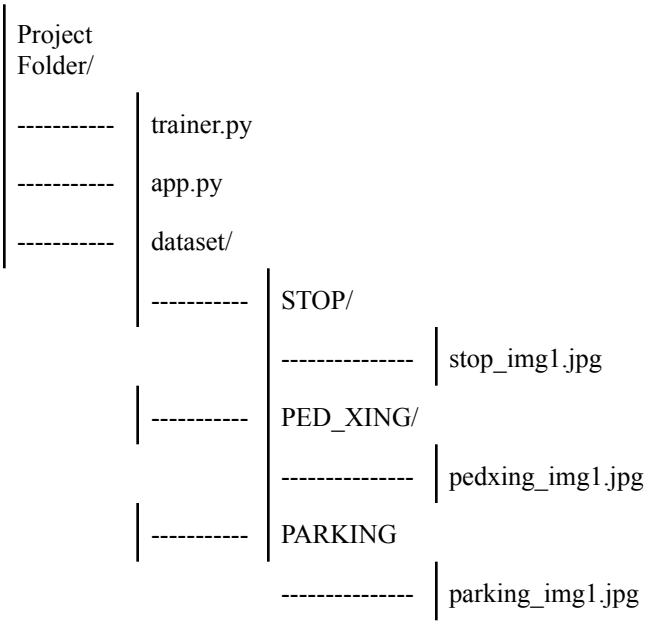

Figure 1. Dataset Folder and File Structure

As seen in Figure 1, there are two existing Python scripts for the entire system namely “trainer.py” and “app.py”. From their assigned names, trainer.py is the Python code implemented when creating, training, and saving a model. This model would then be fed into the main program, app.py. Table 1 and Table 2 present the raw codes for each script, respectively.

```python
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import
Conv2D, MaxPooling2D, Flatten, Dense
from
tensorflow.keras.preprocessing.image
import ImageDataGenerator
from tensorflow.keras.layers import
Dense
```

```
from tensorflow.keras.regularizers
import l2


# Data Preprocessing
data_gen =
ImageDataGenerator(rescale=1./255,
validation_split=0.2)


# Define the training and validation
generators using the ImageDataGenerator
train_gen =
data_gen.flow_from_directory(
    r'File Path',  # Replace with
correct path
    target_size=(64, 64),  # Resizing
to 64x64
    batch_size=32,
    class_mode='categorical',
    subset='training'
)


val_gen = data_gen.flow_from_directory(
    r'File Path',  # Replace with
correct path
    target_size=(64, 64),  # Resizing
to 64x64
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)


# Model Definition
model = Sequential([
    Conv2D(32, (3, 3),
activation='relu', input_shape=(64, 64,
3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3),
activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu',
kernel_regularizer=l2(0.01)),
    Dense(3, activation='softmax')  # 3
classes
])
```

```
# Compile Model
model.compile(optimizer='adam',
loss='categorical_crossentropy',
metrics=['accuracy'])


# Train Model
history = model.fit(train_gen,
validation_data=val_gen, epochs=10)
# Save Model
model.save('road_sign_model.h5')
```

Table 1. trainer.py

Python

```
import cv2
import numpy as np
from tensorflow.keras.models import
load_model

# Load the trained model
model =
load_model('road_sign_model_updated.h5'
)
class_names = ['PARKING', 'PED XING',
'STOP']

def preprocess_frame(frame):
    # Convert to grayscale
    gray = cv2.cvtColor(frame,
cv2.COLOR_BGR2GRAY)
    # Apply Gaussian blur
    blurred = cv2.GaussianBlur(gray,
(5, 5), 0)
    # Perform edge detection
    edges = cv2.Canny(blurred, 50, 150)
    return edges

def extract_contours(edges, frame):
    # Find contours
    contours, _ =
cv2.findContours(edges,
cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    # Select the largest contour with
area threshold
    largest_contour = max(contours,
key=cv2.contourArea, default=None)
    if largest_contour is not None and
cv2.contourArea(largest_contour) >
1500:  # Adjust threshold as needed
```

```
        x, y, w, h =
cv2.boundingRect(largest_contour)
        cropped = frame[y:y+h, x:x+w]
        return cropped
    return None

cap = cv2.VideoCapture(0)

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

    # Preprocess frame
    edges = preprocess_frame(frame)
    sign_region =
extract_contours(edges, frame)

    if sign_region is not None:
        # Resize and normalize for
model
        resized =
cv2.resize(sign_region, (64, 64))
        normalized = resized / 255.0
        reshaped =
np.expand_dims(normalized, axis=0)

        # Predict
        predictions =
model.predict(reshaped)
        class_index =
np.argmax(predictions)
        label =
class_names[class_index]

        # Display label
        cv2.putText(frame, label, (10,
50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,
255, 0), 2)

    # Display the processed frame
    cv2.imshow('Road Sign Detection',
frame)

    # Exit on pressing 'q'
    if cv2.waitKey(1) & 0xFF ==
ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Table 2. app.py

Within the trainer.py, there are 10 epochs wherein the accuracy of the system is presented per trial. As it progresses through the epochs, it continually increases its accuracy, reducing its overall loss. Analyzing the given codes, it can be observed that the model created in trainer.py is based on the dataset folder containing all the necessary images. This model is named and saved into a file "road_sign_model.h5". This same file is stored within the project folder once the trainer.py program is run, in the same file path as trainer.py, app.py, and the dataset folder.

With the model created, the app.py can then be implemented. The program would take a few seconds to load before prompting a Camera window to appear on the user's screen as shown in Figure 2. For the purpose of this demonstration, the proponents tested the efficiency of the system by displaying different images of each road sign using a mobile phone. These results are presented in Figures 3, 4, and 5.

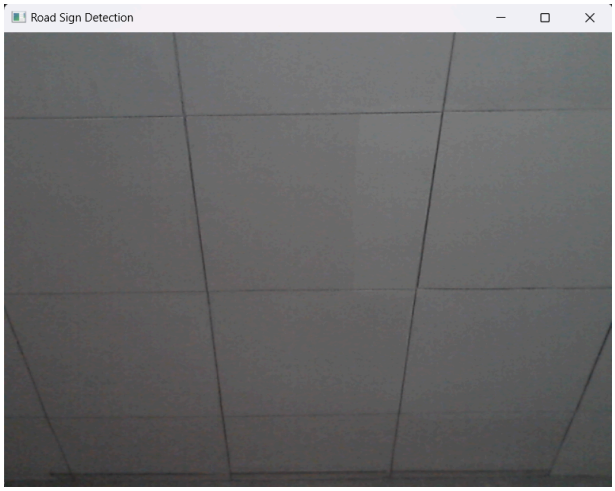

Figure 2. Application Window

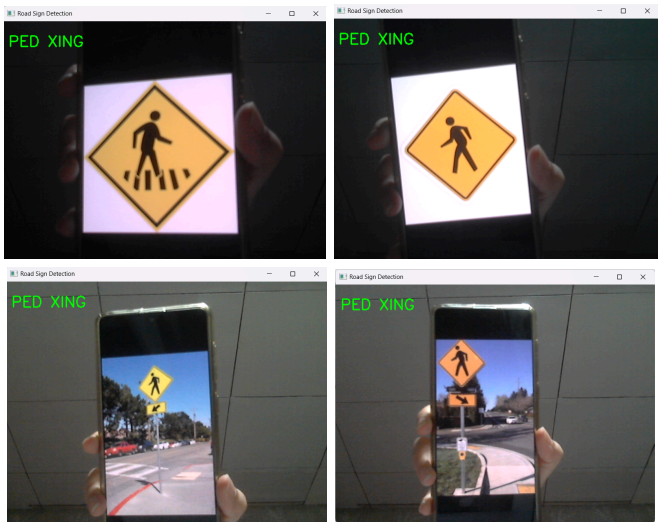

Figure 3. Pedestrian Crossing Sign Detected

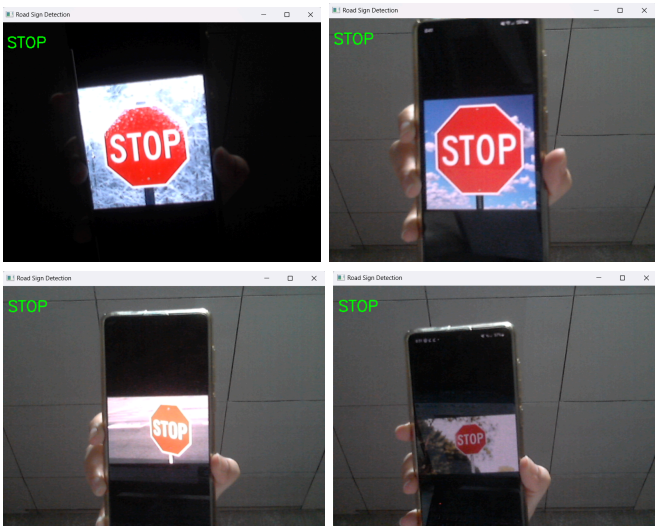

Figure 4. Stop Sign Detected

Figure 5. Parking Sign Detected

From rigorous testing and experimentation, the proponents observed that the classes of Stop and Ped Xing were equally accurate with one another, successfully detecting and recognizing their respective signages. These would also account for non-ideal conditions as well such as inadequate lighting and unsatisfactory angles. On the other hand, although the Parking class was able to fulfill the necessary requirements, few errors would still occur depending on any present discrepancies.

V. Significance of the Project

A traffic sign recognition system (TSRS) is of great importance in a multitude of fields. Development and implementation of the TSRS give contributions to the following:

1. *Road Safety*

Human errors could be significantly lowered on the road through TSRS. The system ensures that drivers obey traffic regulations and stay clear of potential hazards by identifying and correctly interpreting traffic signs in real-time. This is especially helpful to prevent accidents in high-risk areas and in difficult driving circumstances.

2. *Autonomous Vehicles Support*

The automotive industry needs a system for the recognition of traffic signs, since this sector is inching toward full automation. TSRS would improve the ability of autonomous vehicles to make defensible decisions and allow these vehicles to handle the navigation task in complex road situations on its own. This development is important for starting safer, more intelligent transportation networks and for the broader use of autonomous driving technologies.

3. *Urban Traffic Management*

The TSRS can be incorporated into the urban traffic management systems to reduce congestion and streamline traffic flow. The system's useful data can be used by traffic control centers. It allows for a more effective routing of vehicles and synchronization of traffic lights. It results in commute time being shortened and emissions being decreased.

4. *Accessibility and Inclusivity*

The TSRS could also function as an assistive technology, helping individuals with vision problems or even cognitive impairments drive safely and independently. This is the way to increase inclusiveness, enabling access to opportunities and personal mobility for a much wider segment of society.

5. *Research and Development*

Lastly, research in TSRS helps in developing computer vision, machine learning, and artificial intelligence in academic and research circles. New methodologies and algorithms developed from such research have very useful applications in these fields. Similarly, these can also be applied in many other technological fields.

VI. Recommendations

To guarantee the system's accuracy and efficiency, several modifications and improvements can be made by future researchers and developers. Some of these would involve the optimization of the CNN by experimenting with hyperparameters and exploring pretrained models instead of creating a model from scratch. Furthermore, it is suggested

that a higher and more equal number of images are used for each classification. By employing these, the system would significantly improve. Additionally, other road signages could also be included to cater to a wider variety of uses.

VII. Conclusion

The project successfully developed a real-time Traffic Sign Recognition System capable of detecting and classifying specific road signs under varying conditions. By integrating advanced image processing and machine learning techniques, the system demonstrates high accuracy for Stop and Pedestrian Crossing signs, with minor limitations for Parking signs under certain conditions. The TSRS contributes significantly to road safety, autonomous vehicle support, and urban traffic optimization. Future improvements, such as incorporating additional signage and refining the CNN model, could enhance system efficiency and applicability. The study emphasizes the critical role of TSRS in modern transportation systems and its potential to promote inclusivity and technological innovation.

References

[1] Development of road sign recognition for ADAS using OpenCV," 2017 International Conference on Intelligent Computing and Control (I2C2), 2017, pp. 1-5, doi: 10.1109/I2C2.2017.8321941

[2] Traffic Sign Detection and Recognition Using OpenCV," 2014 International Conference on Computational Intelligence and Communication Networks (CICN), 2014, pp. 1-5, doi: 10.1109/CICN.2014.703381

[3] A. Raeburn, "What is Extreme Programming (XP)?," Asana, Nov. 28, 2022. https://asana.com/resources/extreme-programming-xp

[4] Y. Sun, P. Ge, and D. Liu, "Traffic Sign Detection and Recognition Based on Convolutional Neural Network," IEEE Xplore, Nov. 01, 2019. https://ieeexplore.ieee.org/document/8997240

[5] E. Oruklu, D. Pesty, J. Neveux, and J.-E. Guebey, "Real-time traffic sign detection and recognition for in-car driver assistance systems," Aug. 2012, doi: https://doi.org/10.1109/mwscas.2012.6292185.

[6] A. Santos, P. A. Abu, C. Oppus, and R. Reyes, "Traffic Sign Detection and Recognition for Assistive Driving," IEEE Xplore, Aug. 01, 2019. https://ieeexplore.ieee.org/document/8836161

[7] N. P. Botekar and M. N. Mahalakshmi, "Development of road sign recognition for ADAS using OpenCV," pp. 1–4, Jun. 2017, doi: https://doi.org/10.1109/i2c2.2017.8321941.

[8] P. Shopa, N. Sumitha, and P. S. K. Patra, "Traffic sign detection and recognition using OpenCV," IEEE Xplore, Feb. 01, 2014. https://ieeexplore.ieee.org/document/7033810

[9] I. Siniosoglou, P. Sarigiannidis, Y. Spyridis, A. Khadka, G. Efstathopoulos, and T. Lagkas, "Synthetic Traffic Signs Dataset for Traffic Sign Detection & Recognition In Distributed Smart Systems," Zenodo (CERN European Organization for Nuclear Research), Jul. 2021, doi: https://doi.org/10.1109/dcoss52077.2021.00056.

[10] N. Youssouf, "Traffic sign classification using CNN and detection using faster-RCNN and YOLOV4," Heliyon, vol. 8, no. 12, p. e11792, Dec. 2022, doi: https://doi.org/10.1016/j.heliyon.2022.e11792.

[11] H. B. Fredj, A. Chabbah, J. Baili, H. Faiedh, and C. Souani, "An efficient implementation of traffic signs recognition system using CNN," Microprocessors and Microsystems, vol. 98, p. 104791, Apr. 2023, doi: https://doi.org/10.1016/j.micpro.2023.104791.

[12] X. Qiao, "Research on Traffic sign recognition based on CNN Deep Learning Network," Procedia Computer Science, vol. 228, pp. 826–837, 2023, doi: https://doi.org/10.1016/j.procs.2023.11.102.

[13] Y. Zhu and W. Q. Yan, "Traffic sign recognition based on deep learning," Multimedia Tools and Applications, Mar. 2022, doi: https://doi.org/10.1007/s11042-022-12163-0.

[14] A. Juyal, S. Sharma, and P. Matta, "Traffic Sign Detection using Deep Learning Techniques in Autonomous Vehicles," IEEE Xplore, Sep. 01, 2021. https://ieeexplore.ieee.org/abstract/document/9633959/

[15] M. M. William et al., "Traffic Signs Detection and Recognition System using Deep Learning," IEEE Xplore, Dec. 01, 2019. https://ieeexplore.ieee.org/document/9014763 (accessed Aug. 04, 2022).

[16] S. Pathak, R. Rane, G. Chavan, and S. Pundkar, "Traffic Sign Recognition System," International Research Journal of Engineering and Technology, vol. 9(5). 234-237, 2022.

[17] P. Liu, Z. Xie, and T. Li, "UCN-YOLOv5: Traffic Sign Object Detection Algorithm Based on Deep Learning," IEEE Access, vol. 11, pp. 110039–110050, Jan. 2023, doi: https://doi.org/10.1109/access.2023.3322371.

[18] P. S. Kondamari and A. Itha, "A Deep Learning Application for Traffic Sign Classification [BS thesis, Blekinge Institute of Technology]. ."