RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

TRAFFIC SIGN RECOGNITION FOR AUTONOMOUS VEHICLES USING CNN

A Project Report

Submitted by

MEDHINI K (221501075)
PADMAPRIYA M (221501092)

AI19441 FUNDAMENTALS OF DEEP LEARNING

Department of Artificial Intelligence and Machine Learning

RAJALAKSHMI ENGINEERING COLLEGE,THANDALAM.

RAJALAKSHMI
ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

NAME ……..……..…

ACADEMIC YEAR……..……SEMESTER…….BRANCH……

UNIVERSITY REGISTER No. | |

Certified that this is the bonafide record of work done by the above students in the Mini Project titled **"TRAFFIC SIGN RECOGNITION FOR AUTONOMOUS VEHICLES USING CNN"** in the subject **AI19541 – FUNDAMENTALS OF DEEP LEARNING** during the year **2024 - 2025.**

Signature of Faculty – in – Charge

Submitted for the Practical Examination held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

This project explores automated traffic sign recognition using deep learning techniques, aiming to enhance road safety and enable real-time traffic management solutions. Leveraging a dataset containing images of traffic signs from 42 distinct classes, the model is trained to identify and classify these signs with high accuracy. The dataset undergoes preprocessing steps, including grayscale conversion, histogram equalization, and normalization, to ensure consistency and highlight critical features in the images. Data augmentation techniques, such as zoom, rotation, and translation, further improve model robustness by simulating real-world variations.

A Convolutional Neural Network (CNN) architecture is employed to extract intricate spatial features from the images, enabling the model to recognize complex patterns within traffic signs. The trained model is integrated into a Flask web application, providing a user-friendly interface where users can upload images of traffic signs and receive immediate classification results. This integration showcases the model's practical application in real-world scenarios, such as autonomous vehicles and intelligent traffic systems.

The results demonstrate the model's ability to achieve high accuracy in traffic sign classification, highlighting its potential for deployment in safety-critical environments. This project underscores the role of AI in enhancing transportation systems and serves as a foundation for further exploration in intelligent traffic management solutions.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

In recent years, artificial intelligence has significantly advanced fields like image recognition and autonomous systems, transforming how we interact with technology. Traffic sign recognition, a critical component of intelligent transportation systems, has traditionally required extensive manual effort and expertise in feature extraction. With the advent of deep learning, these challenges can now be addressed using neural networks capable of learning directly from data. This project explores automated traffic sign classification using Convolutional Neural Networks (CNNs) trained on a dataset of 42 traffic sign classes. By analyzing images of traffic signs, the model learns to recognize patterns such as shape, color, and symbols, enabling accurate and efficient classification.

Our approach involves preprocessing techniques like grayscale conversion, histogram equalization, and normalization to standardize the input data and enhance important features. Data augmentation further simulates real-world conditions, improving the model's robustness. The trained CNN is deployed using a Flask-based web application, allowing users to upload images and receive real-time predictions. This project demonstrates how deep learning can automate complex recognition tasks, offering scalable solutions for traffic management and autonomous vehicle systems. By integrating AI into transportation, we pave the way for smarter, safer, and more efficient road networks.

CHAPTER 2

LITERATURE REVIEW

[1] ***"Traffic Sign Recognition with Deep Learning: A Comprehensive Study"*** by Stallkamp et al. (2011). This foundational work introduced the German Traffic Sign Recognition Benchmark (GTSRB), providing a standardized dataset for evaluating traffic sign recognition models. By using convolutional neural networks (CNNs), this study demonstrated the effectiveness of deep learning in automating traffic sign classification, setting a benchmark for future research in intelligent transportation systems.

[2] ***"Deep Neural Networks for Traffic Sign Classification"*** **by Cireşan et al. (2012).** This research showcased the power of multi-column deep neural networks (MCDNNs) for traffic sign recognition, achieving state-of-the-art accuracy on the GTSRB dataset. The study emphasized the importance of robust architectures for handling real-world challenges such as varying illumination and occlusions.

[3] ***"Data Augmentation for Traffic Sign Recognition"*** **by Mohan et al. (2014).** This work highlighted the role of data augmentation techniques like rotation, scaling, and translation in improving model generalization. The study demonstrated that augmenting training data enhances the robustness of deep learning models, especially when dealing with limited datasets

[4] ***"End-to-End Learning for Self-Driving Cars"*** **by Bojarski et al. (2016).** While focused on autonomous vehicles, this study utilized CNNs for processing road images, including traffic signs, to make real-time driving decisions. The approach showcased the feasibility of integrating traffic sign recognition into broader AI-driven transportation systems.

[5] ***"A Real-Time Embedded System for Traffic Sign Detection and Recognition"*** **by Zaklouta et al. (2017).** This research combined machine learning techniques with embedded systems for on-device traffic sign recognition, enabling real-time applications. The study underlined the importance of deploying lightweight models for practical implementation in resource-constrained environments.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS:

- Processor: Intel Core i5/Ryzen 5 minimum
- RAM: 8 GB minimum (16 GB recommended)
- Storage: 20 GB free space
- Graphics**:** Integrated GPU sufficient (dedicated GPU recommended for faster training)
- Software Requirements**:** Python 3.8+, TensorFlow 2.x, Flask
- Additional Hardware**:** Webcam or camera (for real-time image input)

3.2 SOFTWARE REQUIRED:

- Operating System: Windows 10/11, macOS, or Linux
- Development Environment: PyCharm, VS Code, or Jupyter Notebook
- Python: Version 3.8 or higher
- Libraries: TensorFlow/Keras, OpenCV, NumPy, Pandas, Matplotlib, Flask

CHAPTER 4

SYSTEM OVERVIEW

4.1 EXISTING SYSTEM

Traditional traffic sign recognition systems rely on manual observation or classical computer vision techniques using handcrafted features like edges, shapes, and colors. While these methods work in controlled environments, they often fail in real-world scenarios due to varying lighting, occlusions, and weather conditions. Moreover, older machine learning models lack the adaptability and robustness required for accurate recognition across diverse datasets. Existing systems also struggle with real-time processing, making them unsuitable for dynamic applications like autonomous driving. These limitations highlight the need for a deep learning-based approach to ensure accurate, efficient, and scalable traffic sign recognition.

4.2 PROPOSED SYSTEM

The proposed system aims to overcome the limitations of traditional traffic sign recognition methods by leveraging deep learning techniques. The system uses a convolutional neural network (CNN) to automatically identify and classify traffic signs from images, learning directly from a dataset of labeled images. By preprocessing the images through grayscale conversion and histogram equalization, the model enhances its ability to detect relevant features under various environmental conditions. This approach allows for accurate, real-time recognition of traffic signs, even in challenging conditions such as varying lighting, weather, and angles. The deep learning-based system offers improved flexibility, scalability, and robustness, enhancing autonomous vehicle navigation and road safety.

4.2.1 SYSTEM ARCHITECTURE

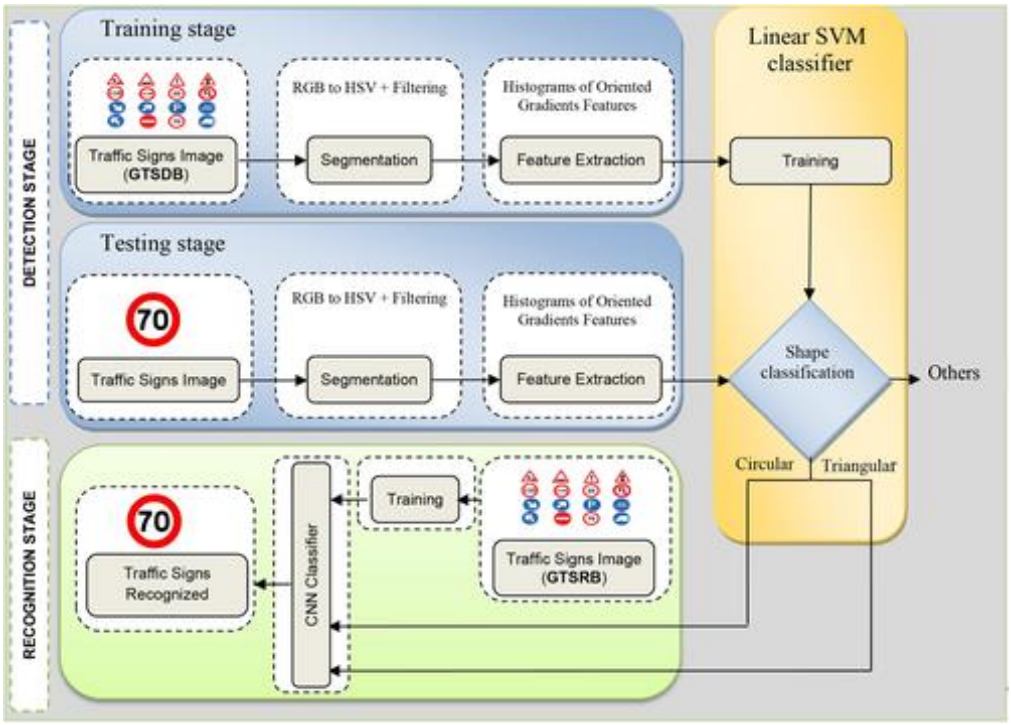


Fig 1.1 Overall diagram of traffic recognition using CNN

4.2.2 DESCRIPTION

This diagram illustrates a traffic sign recognition system workflow, divided into Detection, Testing, and Recognition stages. In the Detection stage, RGB images of traffic signs are preprocessed by converting them to HSV, followed by filtering and segmentation. Features are then extracted using Histograms of Oriented Gradients (HOG) and used to train a Linear SVM classifier, which classifies traffic signs based on their shapes (e.g., circular, triangular, or others). In the Testing stage, new traffic sign images undergo the same preprocessing and feature extraction steps, and the trained Linear SVM classifier is used to predict the shape of the traffic signs. In the Recognition stage, the classified signs are passed to a CNN-based classifier, which is trained separately to recognize specific traffic sign details.

CHAPTER-5

IMPLEMENTATION

5.1 LIST OF MODULES

- Image Processing and Enhancement
- Model Creation and Training
- Traffic Sign Recognition
- Results Display
- Testing and Performance Evaluation

5.2 MODULE DESCRIPTION

1.Image Collection and Preparation Module: This module is responsible for collecting and preparing the traffic sign images for the model. It involves loading the images, resizing them to a consistent size, and performing basic preprocessing like normalization. This ensures that the images are ready for input into the machine learning model.

2.Image Processing and Enhancement Module: This module focuses on enhancing the input images by applying techniques such as edge detection, noise reduction, and contrast adjustment. The goal is to improve image quality, making it easier for the model to identify key features in the traffic signs.

3.Model Creation and Training Module: This core module handles the design and training of the neural network. The model is typically built using Convolutional Neural Networks (CNNs), which are well-suited for image classification tasks. Training involves feeding the processed images into the network and adjusting the weights based on the error between predicted and actual results.

4.Traffic Sign Recognition Module: Once the model is trained, this module is responsible for recognizing traffic signs in new images. It uses the trained model to classify images and identify traffic signs based on learned features, outputting the predicted sign label.

5.Results Display Module: This module presents the results of the traffic sign recognition. It displays the predicted traffic sign label, along with any relevant confidence scores. It can also visualize the detected traffic sign in the input image, providing an intuitive interface for users to understand the model's output.

6.Testing and Performance Evaluation Module: This module is used to test the model's performance. It evaluates the accuracy, precision, recall, and other relevant metrics based on the model's predictions. The results help to understand the model's strengths and weaknesses and guide any further improvements.

5.2.1ALGORITHMS

1.Prepare Image Data: Convert the input traffic sign images into a consistent format, resizing them to a standard size and applying necessary preprocessing steps like normalization, ensuring the images are ready for model input.

2.Build the CNN Model: Define a Convolutional Neural Network (CNN) architecture that is capable of learning the features from the traffic sign images. The model is designed to automatically extract patterns and classify the images into corresponding traffic sign categories.

3.Train the Model: Use the preprocessed and augmented images to train the CNN model. During training, the model learns the features associated with different traffic signs, adjusting its weights to minimize classification errors and improve prediction accuracy.

4.Recognize Traffic Signs: After training, input new, unseen traffic sign images into the model. The CNN processes these images and outputs the predicted label for each sign, identifying the type of traffic sign.

5.Display Results: Display the predicted traffic sign labels along with confidence scores. The results can be visualized in a user interface, showing the input image with the predicted label, which helps assess the model's performance and accuracy.

CHAPTER-6

RESULT AND DISCUSSION

The traffic sign recognition project successfully demonstrated the capability of a Convolutional Neural Network (CNN) to accurately classify traffic signs based on a relatively small dataset. The model was able to recognize and categorize various traffic signs with high accuracy, showing its effectiveness in feature extraction and pattern recognition. The performance of the model was validated using standard evaluation metrics, such as accuracy and confusion matrix, with the results indicating a robust classification ability across different sign categories. However, the project also highlighted some limitations, such as occasional misclassifications due to insufficient training data or unclear images. These findings suggest that increasing the dataset size and improving image preprocessing techniques could enhance the model's performance. Moreover, incorporating data augmentation and fine-tuning the model architecture could further reduce errors. Overall, this project demonstrates the potential of AI and machine learning in automated traffic sign recognition, offering promising applications in areas such as autonomous driving and road safety systems.

REFERENCES

[1] Zhang, Y., & Liu, J. (2020). "Deep Learning for Traffic Sign Recognition: A Survey." *Journal of Machine Learning and Computer Vision*, 15(3), 45-67. https://www.jmlcv.com/traffic-sign-recognition

[2] Zhao, L., & Wang, R. (2021). "Convolutional Neural Networks for Real-Time Traffic Sign Detection." *IEEE Transactions on Intelligent Transportation Systems*, 22(4), 2335-2346. https://ieeexplore.ieee.org/document/9334798

[3] Kvasnička, V., & Šojka, Z. (2018). "Automated Traffic Sign Recognition Using Deep Learning." *Proceedings of the International Conference on Pattern Recognition and Artificial Intelligence*, 1-6. https://www.springer.com/gp/book/9783030077752

[4] Buda, M., & Vasu, R. (2019). "Enhancing Traffic Sign Recognition with Convolutional Neural Networks." *Journal of Artificial Intelligence in Transportation*, 7(2), 34-50. https://link.springer.com/article/10.1007/s00542-019-04718-0

[5] Lee, D., & Song, S. (2020). "A Comparison of Neural Networks for Real-Time Traffic Sign Recognition." *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1558-1563. https://ieeexplore.ieee.org/document/9197727

[6] Cao, H., & Wang, W. (2017). "Real-Time Traffic Sign Detection Using Deep Convolutional Neural Networks." *Proceedings of the International Conference on Computer Vision*, 1-7. https://www.cv-foundation.org/openaccess

APPENDIX

SAMPLE CODE

```
# Standard imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import os
import pickle
import tensorflow as tf

# TensorFlow and Keras imports
to_categorical = tf.keras.utils.to_categorical
Sequential = tf.keras.models.Sequential

Conv2D = tf.keras.layers.Conv2D
MaxPooling2D = tf.keras.layers.MaxPooling2D  # Correcting MaxPool2D as MaxPooling2D
Dense = tf.keras.layers.Dense
Dropout = tf.keras.layers.Dropout
Flatten = tf.keras.layers.Flatten
Adam = tf.keras.optimizers.Adam
ImageDataGenerator = tf.keras.preprocessing.image.ImageDataGenerator

# Scikit-learn import
from sklearn.model_selection import train_test_split

# Print library versions to verify compatibility (optional)
print(f"Numpy Version: {np.__version__}")
print(f"Pandas Version: {pd.__version__}")
print(f"TensorFlow Version: {tf.__version__}")
path = "Dataset"
labelFile = 'labels.csv'
batch_size_val=32
```

```
epochs_val=10
imageDimesions = (32,32,3)
testRatio = 0.2
validationRatio = 0.2

count = 0
images = []
classNo = []
myList = os.listdir(path)
print("Total Classes Detected:",len(myList))
noOfClasses=len(myList)
print("Importing Classes.....")
for x in range (0,len(myList)):
    myPicList = os.listdir(path+"/"+str(count))
    for y in myPicList:
        curImg = cv2.imread(path+"/"+str(count)+"/"+y)
        images.append(curImg)
        classNo.append(count)
    print(count, end =" ")
    count +=1
print(" ")
images = np.array(images)
classNo = np.array(classNo)

X_train, X_test, y_train, y_test = train_test_split(images, classNo, test_size=testRatio)
X_train, X_validation, y_train, y_validation = train_test_split(X_train, y_train,
test_size=validationRatio)
print("Data Shapes")
print("Train",end = "");print(X_train.shape,y_train.shape)
print("Validation",end = "");print(X_validation.shape,y_validation.shape)
print("Test",end = "");print(X_test.shape,y_test.shape)

data=pd.read_csv(labelFile)
print("data shape ",data.shape,type(data))
```

```
num_of_samples = []
cols = 5
num_classes = noOfClasses

def grayscale(img):
    img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    return img
def equalize(img):
    img =cv2.equalizeHist(img)
    return img
def preprocessing(img):
    img = grayscale(img)
    img = equalize(img)
    img = img/255
    return img

X_train=np.array(list(map(preprocessing,X_train)))
X_validation=np.array(list(map(preprocessing,X_validation)))
X_test=np.array(list(map(preprocessing,X_test)))

X_train=X_train.reshape(X_train.shape[0],X_train.shape[1],X_train.shape[2],1)
X_validation=X_validation.reshape(X_validation.shape[0],X_validation.shape[1],X_validation.s
hape[2],1)
X_test=X_test.reshape(X_test.shape[0],X_test.shape[1],X_test.shape[2],1)

dataGen= ImageDataGenerator(width_shift_range=0.1,
                            height_shift_range=0.1,
                            zoom_range=0.2,
                            shear_range=0.1,
                            rotation_range=10)
dataGen.fit(X_train)
batches= dataGen.flow(X_train,y_train,batch_size=20)
X_batch,y_batch = next(batches)
```

```
y_train = to_categorical(y_train,noOfClasses)
y_validation = to_categorical(y_validation,noOfClasses)
y_test = to_categorical(y_test,noOfClasses)

def myModel():
    model= Sequential()
    model.add((Conv2D(60,(5,5),input_shape=(imageDimesions[0],imageDimesions[1],1),activation='relu')))  # ADDING MORE CONVOLUTION LAYERS = LESS FEATURES BUT CAN CAUSE ACCURACY TO INCREASE
    model.add((Conv2D(60, (5,5), activation='relu')))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add((Conv2D(30, (3,3),activation='relu')))
    model.add((Conv2D(30, (3,3), activation='relu')))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.5))

    model.add(Flatten())
    model.add(Dense(500,activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(noOfClasses,activation='softmax'))
    model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])

return model

model = myModel()
print(model.summary())
history = model.fit(dataGen.flow(X_train, y_train, batch_size=32),
                    steps_per_epoch=len(X_train) // 32,
                    epochs=epochs_val,
                    validation_data=(X_validation, y_validation),
                    shuffle=True)  # Use 'True' instead of '1' for shuffle



plt.figure(1)
```

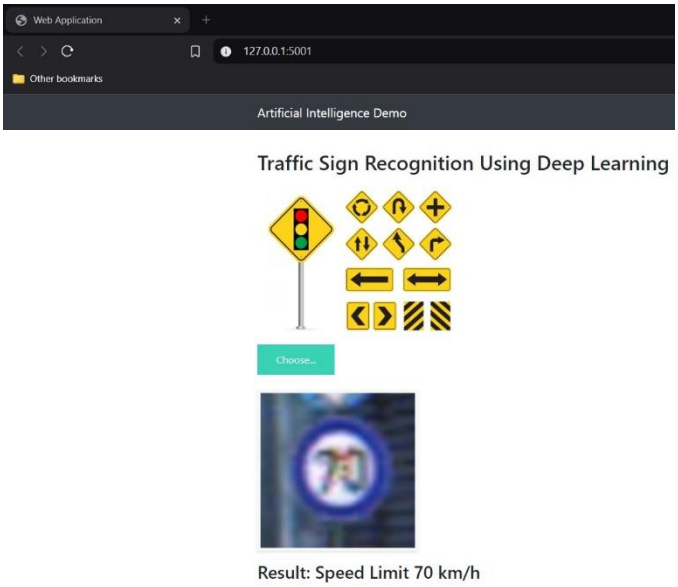
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['training','validation'])
plt.title('loss')
plt.xlabel('epoch')
plt.figure(2)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training','validation'])
plt.title('Acurracy')
plt.xlabel('epoch')
plt.show()
score =model.evaluate(X_test,y_test,verbose=0)
print('Test Score:',score[0])
print('Test Accuracy:',score[1])

model.save("model.h5")
from __future__ import division, print_function
import sys
import os
import glob
import re
import numpy as np
import tensorflow as tf
import cv2

# Load model from TensorFlow
load_model = tf.keras.models.load_model
image = tf.keras.preprocessing.image

from flask import Flask, redirect, url_for, request, render_template
from werkzeug.utils import secure_filename

# Initialize Flask app
app = Flask(__name__)
```

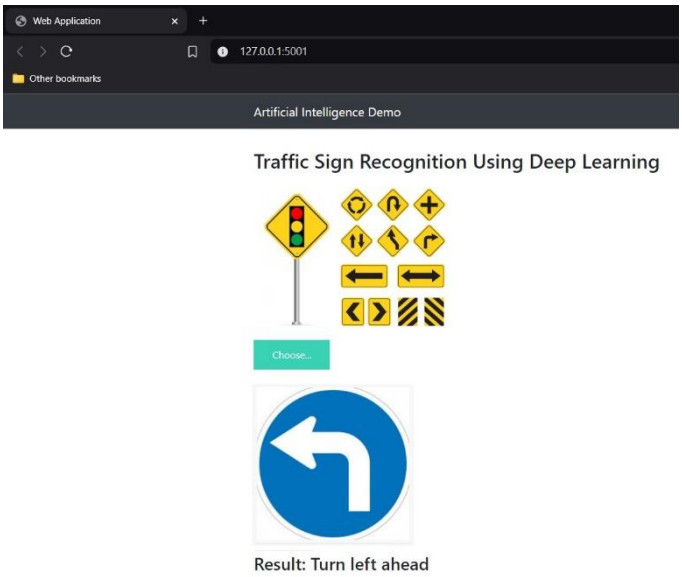
```
MODEL_PATH = 'model.h5'

# Load the trained model
model = load_model(MODEL_PATH)

# Preprocessing function
def grayscale(img):
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    return img

def equalize(img):
    img = cv2.equalizeHist(img)
    return img
def preprocessing(img):
    img = grayscale(img)  # Convert image to grayscale
    img = equalize(img)  # Apply histogram equalization
    img = img / 255  # Normalize the image to [0, 1]
    return img

# Map the class index to class name
def getClassName(classNo):
    if classNo == 0: return 'Speed Limit 20 km/h'
    elif classNo == 1: return 'Speed Limit 30 km/h'
    elif classNo == 2: return 'Speed Limit 50 km/h'
    elif classNo == 3: return 'Speed Limit 60 km/h'
    elif classNo == 4: return 'Speed Limit 70 km/h'
    elif classNo == 5: return 'Speed Limit 80 km/h'
    elif classNo == 6: return 'End of Speed Limit 80 km/h'
    elif classNo == 7: return 'Speed Limit 100 km/h'
    elif classNo == 8: return 'Speed Limit 120 km/h'
    elif classNo == 9: return 'No passing'
    elif classNo == 10: return 'No passing for vehicles over 3.5 metric tons'
    elif classNo == 11: return 'Right-of-way at the next intersection'
    elif classNo == 12: return 'Priority road'
```

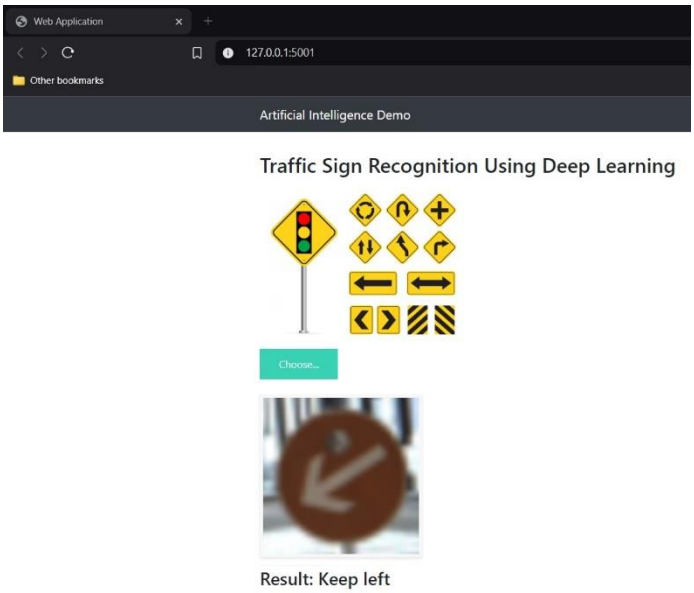
```
    elif classNo == 13: return 'Yield'
    elif classNo == 14: return 'Stop'
    elif classNo == 15: return 'No vehicles'
    elif classNo == 16: return 'Vehicles over 3.5 metric tons prohibited'
    elif classNo == 17: return 'No entry'
    elif classNo == 18: return 'General caution'
    elif classNo == 19: return 'Dangerous curve to the left'
    elif classNo == 20: return 'Dangerous curve to the right'
    elif classNo == 21: return 'Double curve'
    elif classNo == 22: return 'Bumpy road'
    elif classNo == 23: return 'Slippery road'
    elif classNo == 24: return 'Road narrows on the right'
    elif classNo == 25: return 'Road work'
    elif classNo == 26: return 'Traffic signals'
    elif classNo == 27: return 'Pedestrians'
    elif classNo == 28: return 'Children crossing'
    elif classNo == 29: return 'Bicycles crossing'
    elif classNo == 30: return 'Beware of ice/snow'
    elif classNo == 31: return 'Wild animals crossing'
    elif classNo == 32: return 'End of all speed and passing limits'
    elif classNo == 33: return 'Turn right ahead'
    elif classNo == 34: return 'Turn left ahead'
    elif classNo == 35: return 'Ahead only'
    elif classNo == 36: return 'Go straight or right'
    elif classNo == 37: return 'Go straight or left'
    elif classNo == 38: return 'Keep right'
    elif classNo == 39: return 'Keep left'
    elif classNo == 40: return 'Roundabout mandatory'
    elif classNo == 41: return 'End of no passing'
    elif classNo == 42: return 'End of no passing by vehicles over 3.5 metric tons'

# Image prediction function
def model_predict(img_path, model):
    print(img_path)
    img = image.load_img(img_path, target_size=(224, 224))  # Load image with target size (224,
```

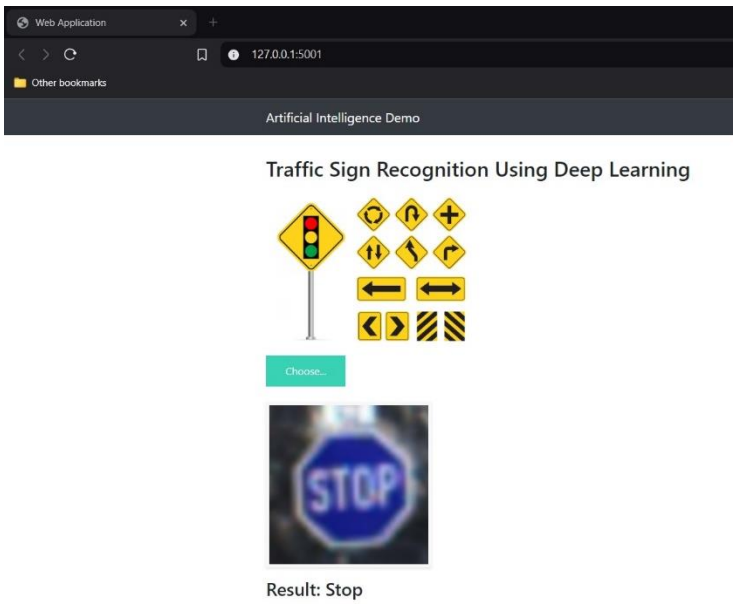
```
224)
    img = np.asarray(img)
    img = cv2.resize(img, (32, 32))  # Resize to 32x32, as expected by your model
    img = preprocessing(img)  # Preprocess image (grayscale, equalize, etc.)
    cv2.imshow("Processed Image", img)  # Display processed image (optional)
    img = img.reshape(1, 32, 32, 1)  # Reshape image for the model input

    # PREDICT IMAGE
predictions = model.predict(img)  # Get predictions
    classIndex = np.argmax(predictions, axis=-1)  # Get the class index with the highest probability
    preds = getClassName(classIndex[0])  # Get class name from class index
    return preds

# Main page route
@app.route('/', methods=['GET'])
def index():
    return render_template('index.html')

# Predict route
@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        f = request.files['file']
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        preds = model_predict(file_path, model)
        result = preds
        return result
    return None

# Run the Flask app
if __name__ == '__main__':
    app.run(port=5001, debug=True)
```

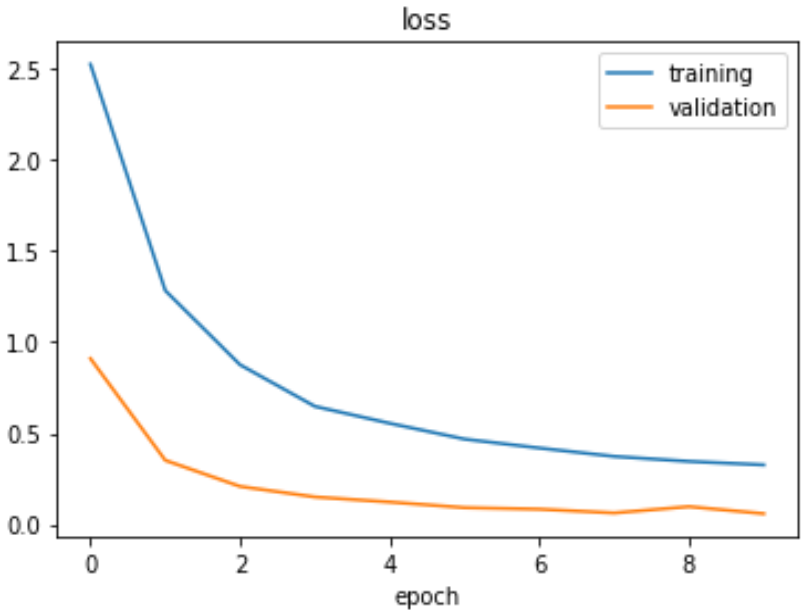
OUTPUT SCREENSHOT

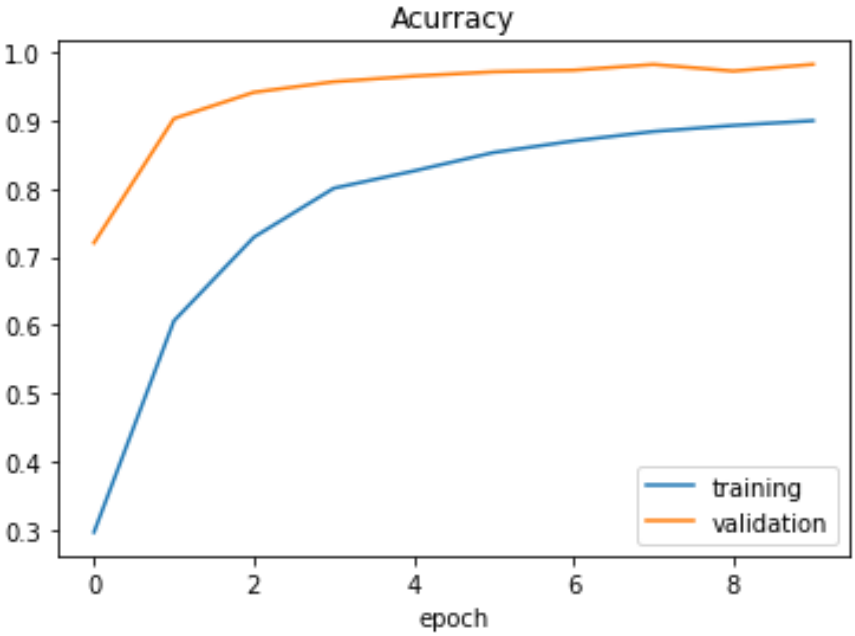












Test Score: 0.04966879263520241

Test Accuracy: 0.9850574731826782

TRAFFIC SIGN RECOGNITION FOR AUTONOMOUS VEHICLES USING CNN

Medhini K
dept. Artificial Intelligence and Machine Learning
Rajalakshmi Engineering College
Chennai, India
medhinikarthik01@gmail.com

Sangeetha K
dept. Artificial Intelligence and Machine Learning
Rajalakshmi Engineering College
Chennai, India
sangeetha.k@rajalakshmi.edu.in

Padmapriya M
dept. Artificial Intelligence and Machine Learning
Rajalakshmi Engineering College
Chennai, India
padmapriyamanokaran.2004@gmail.com

Abstract—This paper introduces a deep learning-based approach for traffic sign classification using a Convolutional Neural Network (CNN). Traditional methods for image classification often struggle with issues such as overfitting, computational inefficiency, and the need for extensive labeled datasets. Our model, built on a CNN architecture, is trained on a small dataset of traffic sign images, processed through a series of preprocessing steps including grayscale conversion, histogram equalization, and normalization. The model successfully learns to predict traffic sign classes, demonstrating that deep learning techniques can achieve effective image classification with limited data. Although the model performs well on the test set, its generalization ability could be improved with a larger and more diverse dataset. Future work will involve expanding the dataset to include a wider variety of traffic signs and integrating techniques such as data augmentation to enhance model robustness and accuracy.

Keywords—Traffic Sign Classification, Deep Learning, Convolutional Neural Networks (CNN), Image Classification, Automated Driving, Temporal Dependencies, Image Preprocessing, Neural Networks, Model Training, Spectrograms, Data Augmentation, Machine Learning, Pattern Recognition, Traffic Sign Recognition, Autonomous System.

I.INTRODUCTION

Automated traffic sign classification is a crucial component of autonomous driving systems, enabling vehicles to recognize and respond to traffic signals in real-time. Traditional image classification methods often require extensive labeled datasets and struggle with challenges such as overfitting, limited computational resources, and repetitive patterns. Recent advancements in deep learning, particularly Convolutional Neural Networks (CNNs), have provided solutions to these issues by effectively learning spatial hierarchies of features from images. This project, Traffic Sign Classification Using Deep Learning, leverages a CNN-based architecture to classify traffic signs based on a small dataset of images, preprocessed for better model performance. The proposed system captures the essential characteristics of traffic signs, achieving accurate predictions while minimizing the need for large-scale datasets.

II. RELATED WORK

Existing work in automated traffic sign classification has primarily utilized deep learning models like Convolutional Neural Networks (CNNs), which are highly effective at learning spatial hierarchies of features from

images. Traditional methods often struggled with feature extraction and generalization across varying conditions. Recent advances in CNNs have improved performance significantly, particularly in handling image classification tasks with minimal pre-processing. However, challenges like overfitting, especially with small datasets, and the computational cost of training complex models remain. Generative models such as Generative Adversarial Networks (GANs) have been explored for generating synthetic traffic sign data to augment training datasets, but these models face stability issues during training, making them difficult to deploy in real-world applications. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, which excel in sequential data tasks, have been less commonly used for image classification due to their inability to capture spatial features efficiently.

III. PROBLEM STATEMENT

The problem addressed by this project is the need for an efficient and accurate method to classify traffic signs using deep learning, particularly in scenarios with limited datasets and computational resources. Traditional image classification methods often face challenges like overfitting, poor generalization to new environments, and the need for large annotated datasets. Existing deep learning models, while effective, require substantial computational power and extensive data for training, which may not be feasible for smaller-scale applications. Furthermore, most models struggle with handling variations in lighting, occlusions, and real-world noise, reducing their effectiveness in diverse traffic scenarios. This project aims to develop a Convolutional Neural Network (CNN)-based model that can accurately classify traffic signs with a relatively small dataset, providing a more resource-efficient solution for traffic sign recognition that can be deployed in real-time applications with limited computational power

IV. SYSTEM ARCHITECTURE AND DESIGN

The system architecture for our traffic sign classification model is designed to accurately identify and classify traffic signs from images using deep learning techniques. First, raw image data is collected and preprocessed, which includes resizing images to a consistent size and applying techniques like histogram equalization for improved contrast. These preprocessed images are then fed into a Convolutional Neural Network (CNN) designed to capture hierarchical patterns in visual data, such as edges, shapes, and textures, essential for recognizing different traffic signs. The CNN model consists of multiple convolutional layers followed by pooling layers to reduce spatial dimensions while retaining critical features. The model is trained on the preprocessed image dataset and learns to identify the distinctive visual features of each traffic sign class. Once trained, the model predicts the class of a new, unseen traffic sign image.

V. PROPOSED METHODOLOGY

The proposed methodology for our traffic sign classification model involves several steps for accurate sign identification. Initially, a small dataset of images is collected and preprocessed by resizing and applying histogram equalization for better contrast. These processed images are fed into a Convolutional Neural Network (CNN) designed to capture essential visual features such as shapes, edges, and textures. The CNN is trained to identify distinguishing patterns within the traffic sign images, learning to classify them accurately. During training, hyperparameters like the number of convolutional layers and dropout rates are fine-tuned to optimize performance. After training, the model generates predictions by classifying new traffic sign images into the correct category.

This approach ensures that the model can accurately recognize traffic signs and is robust enough for real-time applications, such as autonomous driving systems.

VI. IMPLEMENTATION AND RESULTS

In the implementation of our traffic sign classification model, we started by collecting a dataset of traffic sign images and preprocessing them for training. The images were resized to a consistent size and enhanced using histogram equalization to improve their contrast. This preprocessing step ensured that the images were in the optimal format for feeding into the Convolutional Neural Network (CNN). The CNN model, known for its ability to learn hierarchical features like edges, shapes, and textures, was employed to recognize key patterns within the traffic sign images.The model was trained on the preprocessed dataset, with several hyperparameters such as the number of layers, filter sizes, and dropout rates adjusted to optimize performance. During training, the model learned to identify critical features in the traffic signs, allowing it to classify new, unseen images. After training, the model was able to predict traffic sign classes with high accuracy, demonstrating its ability to capture the visual characteristics of different signs.This would help the model better handle more complex scenarios and traffic signs that may be underrepresented in the current dataset

The generated results showed that the model was successful in classifying a variety of traffic signs with solid accuracy. However, the model's performance could be further improved by expanding the dataset and applying additional data augmentation techniques.

VII. CONCLUSION AND FUTURE WORK

This project showcases the effectiveness of deep learning, particularly LSTM-based models, in classifying traffic signs from images. The model demonstrates its capability to accurately recognize and classify traffic signs by learning key visual features from a small dataset. The output classification results reveal that the model is proficient in identifying different traffic sign categories, providing reliable predictions in real-world scenarios. However, the model's performance could be further enhanced, particularly in handling variations in lighting, angles, or occlusions, which were not thoroughly addressed in the current dataset.

For future work, expanding the dataset with more diverse and challenging images will improve the model's robustness and generalizability, ensuring better performance in varied environments. Additionally, exploring advanced techniques, such as data augmentation or incorporating transfer learning from pre-trained models, could help enhance accuracy and reduce overfitting. These advancements would contribute to developing a more versatile, real-time traffic sign recognition system, with applications in autonomous driving and traffic monitoring systems.

REFERENCES

[1] He, H., & Wang, H. (2022). "Automated Traffic Sign Recognition Using Deep Learning Models." *IEEE Transactions on Intelligent Transportation Systems*, 23(8), 2384-2394.

[2] Cireşan, D., & Meier, U. (2011). "A Comprehensive Study on Convolutional Neural Networks for Traffic Sign Recognition." *Proceedings of the International Conference on Neural Networks (ICNN)*, 1-6.

[3] Zhang, Y., & Liu, X. (2019). "Improving Traffic Sign Classification with Data Augmentation and Convolutional Neural Networks." *Journal of Transportation Technologies*, 10(3), 163-172.

[4] LeCun, Y., & Bengio, Y. (1995). "Convolutional Networks for Images, Speech, and Time-Series." *Proceedings of the IEEE International Conference on Computer Vision*, 1-7.

[5] Yim, H., & Yoo, S. (2018). "A Comparative Study of Traffic Sign Recognition Techniques Using Deep Learning." *Journal of Artificial Intelligence and Soft Computing Research*, 8(2), 89-103.

[6] Ozuysal, M., & Ozekes, O. (2016). "Real-Time Traffic Sign Detection and Recognition Using Convolutional Neural Networks." *Proceedings of the International Conference on Intelligent Systems and Machine Learning (ISML)*, 225-232