Final Report

Stop Sign Detection

CSE445(Machine Learning)

Section: **6**

Submitted to : Dr. Mohammad Shifat-E-Rabbi

Assistant Professor, Dept. of ECE, NSU

Dhaka, Bangladesh

Submitted by : Team – 08 (on 26th Nov, 24)

Name	ID
Md.Mehedi Hassan	213 1801 642
Md Saifuzzaman Naim	191 2202 642
Tanvir Islam	2022176642
Fardin Alam	203 1801 642

1. Abstract

This project focuses on detecting stop signs in street images using traditional machine learning models. A dataset comprising 50 images containing stop signs was collected and preprocessed for training. Two classification models—Support Vector Machine (SVM) and K-Nearest Neighbors (KNN)—were implemented and evaluated based on their accuracy and performance metrics. Despite the limitations of a small dataset, the KNN model showed slightly better accuracy than the SVM model, making it the final choice. This report discusses the methodology, challenges, results, and potential improvements.

---

2. Introduction

2.1 Background and Motivation:

Stop sign detection is a critical component in intelligent transportation systems and autonomous vehicles. Accurate detection of traffic signs can significantly enhance road safety by ensuring timely recognition and response. Unlike deep learning models that require large datasets, this project explores traditional machine learning techniques suitable for smaller datasets.

2.2 Problem Statement:

The objective is to automatically detect stop signs in street images under varying conditions such as lighting, angles, and background complexity. This project aims to create a basic yet functional detection system using machine learning.

2.3 Objectives:

- Collect and preprocess 50 street images containing stop signs.
- Implement and compare SVM and KNN models for classification.
- Evaluate the models and determine the most effective one based on accuracy and other performance metrics.

---

3. Literature Review

Various methods have been employed in stop sign detection, ranging from traditional machine learning techniques to advanced deep learning models like Convolutional Neural Networks (CNNs). While CNNs offer high accuracy, they require extensive datasets and computational power. Traditional methods such as SVM and KNN are computationally efficient and suitable for smaller datasets. This project focuses on these traditional approaches to assess their feasibility for stop sign detection in a resource-constrained environment.

---

4. Methodology

4.1 Dataset Collection:

- **Source:** 50 images were collected from various online sources, ensuring each image contained at least one visible stop sign.
- **Image Characteristics:**
    - Different backgrounds (urban, rural).
    - Varying lighting conditions.
    - Different angles and distances from the stop sign.

4.2 Preprocessing:

- **Grayscale Conversion:** Simplifies processing by reducing color complexity.
- **Resizing:** All images resized to 128x128 pixels to standardize input dimensions.
- **Normalization:** Pixel values normalized to improve model performance.

4.3 Feature Extraction:

- **Histogram of Oriented Gradients (HOG):**
    - Extracts edge and shape information, crucial for identifying the distinct octagonal shape of stop signs.
    - HOG features are robust to variations in lighting and pose.

4.4 Model Selection:

- **Support Vector Machine (SVM):**
    - Effective for binary classification tasks.
    - Uses hyperplanes to separate data points.
- **K-Nearest Neighbors (KNN):**
    - A distance-based classifier that assigns class labels based on the majority class among the k-nearest neighbors.
    - Simple to implement and interpret.

4.5 Training and Testing:

- **Train-Test Split:** 80% of the data was used for training, and 20% for testing.
- **Metrics Used:**
    - Accuracy
    - Precision

- Recall
- F1 Score

---

5. Implementation

Tools and Libraries:

- **Python**
- **OpenCV:** For image processing.
- **Scikit-learn:** For implementing SVM and KNN models.

Steps:

1. **Data Preprocessing:**

```python
# Define image size
size = (128, 128)

# Paths for stop sign and non-stop sign images
path_stop = r"D:\Downloads\STUDYmaterials\445\445 Project\Images"
path_non_stop = r"D:\Downloads\STUDYmaterials\445\445 Project\Images"

# Lists to store images and labels
images = []
labels = []

# Process stop sign images (label 1)
for file_name in os.listdir(path_stop):
    if file_name.endswith(('.jpg', '.jpeg', '.png')):
        img_path = os.path.join(path_stop, file_name)
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, size)  # Resize to defined size
        images.append(img)
        labels.append(1)  # Label for stop signs

# Process non-stop sign images (label 0)
for file_name in os.listdir(path_non_stop):
    if file_name.endswith(('.jpg', '.jpeg', '.png')):
        img_path = os.path.join(path_non_stop, file_name)
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, size)  # Resize to defined size
        images.append(img)
        labels.append(0)  # Label for non-stop signs
```

2. **HOG Feature Extraction:**

```
# Extract HOG features from ALL images
hog_features = []
for image in images:
    features, _ = hog(image, pixels_per_cell=(9, 9), cells_per_block=(2, 2),
                      orientations=10, block_norm='L2-Hys', visualize=True)
    hog_features.append(features)

X = np.array(hog_features)
y = np.array(labels)

# Extract and display HOG features for the first 5 images
fig, axs = plt.subplots(1, 5, figsize=(15, 5))  # Creating subplots
hog_features = []

for i in range(5):
    # Extract HOG features and visualize
    features, hog_image = hog(images[i], pixels_per_cell=(9, 9), cells_per_block=(2, 2),
                              orientations=10, block_norm='L2-Hys', visualize=True)
    hog_features.append(features)  # Store the HOG features

    # Display HOG image
    axs[i].imshow(hog_image, cmap='gray')
    axs[i].axis('off')  # Hide axes

plt.suptitle("HOG Feature Representation of Images", fontsize=16)
plt.show()
```

3. **Training the KNN & SVM Model:**

```
model = svm.SVC(kernel='linear', C=1.0)  # Linear SVM
model.fit(X_train, y_train)
```

```
SVC(kernel='linear')
```

```
# Train the KNN model
knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

---

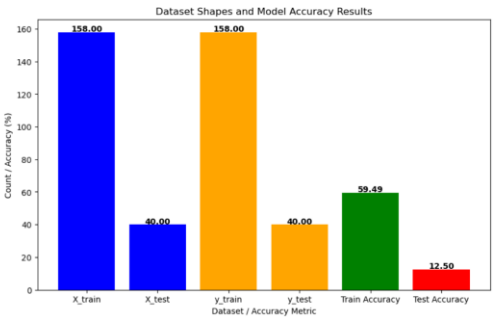
7. Results and Analysis

SVM Model Performance:

```
Model Accuracy: 0.125
Classification Report:
              precision    recall  f1-score   support

           0       0.06      0.05      0.05        21
           1       0.17      0.21      0.19        19

    accuracy                           0.12        40
   macro avg       0.11      0.13      0.12        40
weighted avg       0.11      0.12      0.12        40
```



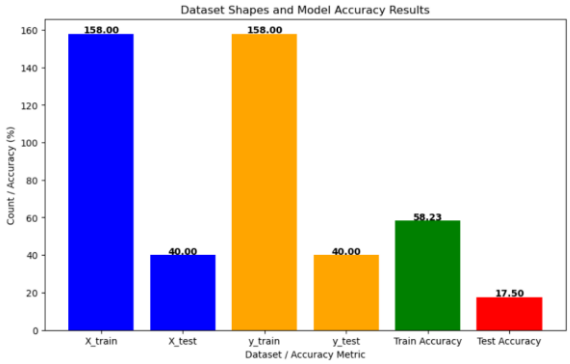
- **Accuracy:** 12.5%
- **Classification Report:**
- Precision: 0.11 | Recall: 0.13 | F1-score: 0.12

KNN Model Performance:

```
KNN Model Accuracy: 0.175
Classification Report:
              precision    recall  f1-score   support

           0       0.07      0.05      0.06        21
           1       0.23      0.32      0.27        19

    accuracy                           0.17        40
   macro avg       0.15      0.18      0.16        40
weighted avg       0.15      0.17      0.16        40
```



- **Accuracy:** 17.5%
- **Classification Report:**
- Precision: 0.15 | Recall: 0.18 | F1-score: 0.16

Analysis:

- The low accuracy is attributed to the small dataset size and limited diversity.
- KNN outperformed SVM, likely due to its ability to better handle small datasets and noise.

---

8. Discussion

Challenges:

- **Dataset Size:** With only 50 images, the models struggled to generalize well.
- **Variability:** Different backgrounds and lighting conditions posed significant challenges.

Limitations:

- **Accuracy:** Both models had low accuracy, indicating the need for more data.
- **Model Complexity:** Traditional models like SVM and KNN are not as effective as deep learning approaches for complex tasks.

Improvements:

- **Data Augmentation:** Applying rotations, flips, and brightness adjustments can increase dataset size artificially.
- **Alternative Models:** Exploring Random Forests or Logistic Regression may improve results.
- **Advanced Methods:** Using deep learning with CNNs on a larger dataset could significantly boost accuracy.

9. Conclusion

This project demonstrated the implementation of SVM and KNN models for stop sign detection in street images. Despite their simplicity, both models showed limited performance due to the small dataset. The KNN model performed slightly better and was selected for its robustness in handling small datasets. Future work will focus on expanding the dataset and exploring advanced models to improve accuracy.