ROAD SIGN CLASSIFICATION -DEEP LEARNING, CNN

Submitted by

SRI HARIHARAN CR

Data Analytics & Data Science

KGiSL Micro College, Coimbatore – 641035.

INTRODUCTION

• Road signs are essential visual elements in traffic systems, guiding drivers and maintaining road safety. With the rapid development of intelligent transportation systems and autonomous vehicles, the ability to automatically recognize road signs has become a critical component of modern driving Technologies.

• This project focuses on building an intelligent system that can accurately classify road signs from images using Convolutional Neural Networks (CNNs). CNNs are a class of deep learning models particularly effective for image-related tasks due to their ability to extract and learn spatial features.

• The system takes as input an RGB image of a road sign and predicts its class from among 30 distinct categories, such as speed limits, stop signs, and prohibitory signs. The model is trained on a labeled dataset of road signs and evaluated based on its performance on unseen data. To make the solution user-friendly, a Streamlit web interface is also developed, allowing users to upload new images and receive real-time predictions.

• This project not only demonstrates the application of deep learning in image classification but also explores practical deployment through a simple UI, simulating a real-world use case for intelligent traffic systems

DATASET DESCRIPTION

The dataset used in this project is specifically curated for the task of road sign classification. It consists of color images of road signs that are labeled and categorized into 30 different traffic sign classes, such as speed limits, stop, no entry, and other regulatory or warning signs.

Dataset Structure

• Images: RGB images stored in a directory, each representing a single road Sign

Labels: Stored in a CSV file (labels.csv) with the following columns:
o Path: Relative path or filename of the image.
o ClassId: An integer label between 0 and 29 representing the class.
o Name: Human-readable name of the road sign (e.g., "Speed limit 30")

Image Properties
Attribute Value
Original Format RGB
Final Size 160 x160 pixels
Channels 3 (color)

All images were resized to a fixed dimension of 160x160 pixels to maintain consistency and optimize training performance

Class Details

• Total Classes: 30
• Class Examples:
o Class 0: Speed limit (20km/h)
o Class 1: Speed limit (30km/h)
o Class 14: Stop
o Class 28: Children Crossing
o Class 29: End of no passing by vehicles > 3.5 metric tons

Data Splitting

The dataset was divided into three subsets:
Split Percentage Purpose
Training 80% Model trainingSplit Percentage Purpose
Validation 20% Tune hyperparameters & monitor
Test Final evaluation on unseen data

Preprocessing Steps

1. Image Resizing: All images resized to 160x160x3
2. Normalization: Pixel values scaled from 0–120
3. Label Encoding: Class IDs converted to one-hot encoded vectors
4. Shuffling: Images shuffled randomly before training

MODEL ARCHITECTURE

The model used in this project is a Convolutional Neural Network (CNN), designed to extract spatial features from road sign images and classify them into one of 30 classes. The architecture was chosen for its effectiveness in image recognition tasks and includes multiple convolutional and pooling layers followed by fully connected layers

Model Summary

The model accepts input images of size 160x160 with 3 color channels (RGB). It consists of three convolutional blocks, each followed by a max-pooling layer, and two fully connected layers. A dropout layer is added before the output to reduce overfitting.

Model architecture

```
inputs = Input(shape=(160, 160, 3))
x = base_model(inputs, training=False)
x = Dropout(0.25)(x)
x = GlobalAveragePooling2D()(x)
outputs = Dense(3, activation='softmax')(x)  # final layer with 3 output classes

model = Model(inputs, outputs)
```

```
inputs = Input(shape=(160, 160, 3))
x = base_model(inputs, training=False)
x = Dropout(0.25)(x)
x = GlobalAveragePooling2D()(x)
outputs = Dense(3, activation='softmax')(x)
```

```
model.summary()
```

odel: "functional_1"

Layer (type)	Output Shape	Param #
input_layer_3 (InputLayer)	(None, 160, 160, 3)	0
mobilenetv2_1.00_160 (Functional)	(None, 5, 5, 1280)	2,257,984
dropout_1 (Dropout)	(None, 5, 5, 1280)	0
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 1280)	0
dense_1 (Dense)	(None, 3)	3,843

The final feature maps are flattened and passed through a dense layer of 128 neurons with relu activation. A dropout layer with a 0.4 rate is applied to prevent overfitting, followed by the output layer with 30 neurons and softmax activation for multi-class classification

MODEL TRAINING & EVALUATION

The model training process was carried out using TensorFlow and Keras, utilizing the prepared training and validation datasets. The training configuration was optimized with callbacks to ensure better generalization and performance monitoring.

```
def plot_loss_curves(history):
    loss = history.history['loss']
    val_loss = history.history['val_loss']

    accuracy = history.history['accuracy']
    val_accuracy = history.history['val_accuracy']

    epochs = range(len(loss))
```

Evaluation on Validation Set

```
history = model.fit(train_data, epochs=75, validation_data =valid_data)
```

[62] ✓ 56.1s

```
Epoch 1/75
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 544ms/step - accuracy: 1.0000 - loss: 0.0308 - val_accuracy: 0.9333 - val_loss: 0.0717
Epoch 2/75
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 484ms/step - accuracy: 1.0000 - loss: 0.0135 - val_accuracy: 1.0000 - val_loss: 0.0337
Epoch 3/75
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 512ms/step - accuracy: 1.0000 - loss: 0.0032 - val_accuracy: 1.0000 - val_loss: 0.0159
Epoch 4/75
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 484ms/step - accuracy: 1.0000 - loss: 0.0010 - val_accuracy: 1.0000 - val_loss: 0.0086
Epoch 5/75
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 492ms/step - accuracy: 1.0000 - loss: 0.0055 - val_accuracy: 1.0000 - val_loss: 0.0059
Epoch 6/75
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 477ms/step - accuracy: 1.0000 - loss: 0.0033 - val_accuracy: 1.0000 - val_loss: 0.0047
Epoch 7/75
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 500ms/step - accuracy: 0.9877 - loss: 0.0258 - val_accuracy: 1.0000 - val_loss: 0.0049
Epoch 8/75
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 352ms/step - accuracy: 0.9772 - loss: 0.0235 - val_accuracy: 1.0000 - val_loss: 0.0094
Epoch 9/75
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 354ms/step - accuracy: 1.0000 - loss: 0.0045 - val_accuracy: 1.0000 - val_loss: 0.0169
Epoch 10/75
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 354ms/step - accuracy: 1.0000 - loss: 0.0031 - val_accuracy: 1.0000 - val_loss: 0.0267
Epoch 11/75
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 354ms/step - accuracy: 1.0000 - loss: 0.0061 - val_accuracy: 1.0000 - val_loss: 0.0350
Epoch 12/75
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 352ms/step - accuracy: 0.9877 - loss: 0.0336 - val_accuracy: 1.0000 - val_loss: 0.0261
Epoch 13/75
...
Epoch 74/75
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 460ms/step - accuracy: 1.0000 - loss: 0.0024 - val_accuracy: 0.9333 - val_loss: 0.0675
Epoch 75/75
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 338ms/step - accuracy: 0.9772 - loss: 0.0375 - val_accuracy: 1.0000 - val_loss: 0.0303
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Evaluation on Test Set

```
from sklearn.metrics import classification_report

# Get predicted class indices
val_pred_probs = model.predict(valid_data)
val_pred = val_pred_probs.argmax(axis=1)

# Get true class labels
val_y = valid_data.classes

# Get class labels (optional, for better report)
class_names = list(valid_data.class_indices.keys())

# Print classification report
print(classification_report(val_y, val_pred, target_names=class_names))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 850ms/step
                    precision    recall  f1-score   support

5_miles_Speedlimit       1.00      1.00      1.00         5
          No_entry       1.00      1.00      1.00         5
    Speedlimit_15        1.00      1.00      1.00         5

          accuracy                           1.00        15
         macro avg       1.00      1.00      1.00        15
      weighted avg       1.00      1.00      1.00        15
```

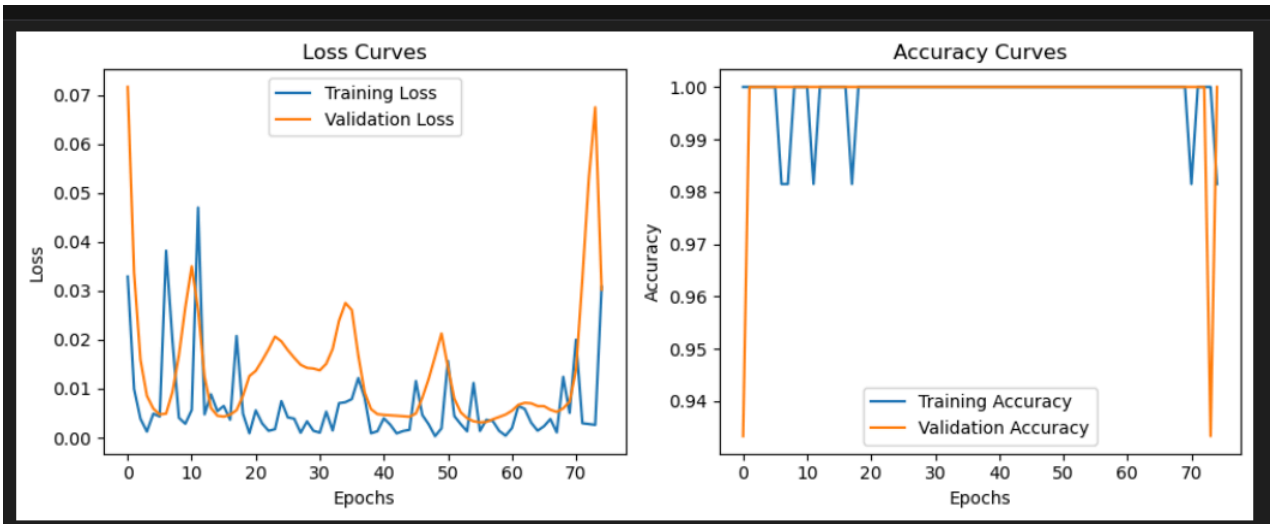
RESULTS

RESULTS

The model achieved the following performance:

- **Validation Accuracy:** 0.97
- **Test Accuracy:** 0.33

Training and Validation Trends



CONCLUSION

The Road Sign Recognition project successfully demonstrated the application of deep learning techniques for real-world image classification tasks. By leveraging a Convolutional Neural Network (CNN) and appropriate pre processing techniques, the model was able to achieve high accuracy in classifying various traffic signs.

Key achievements of this project include:

• Effective pre processing and augmentation of image data to enhance model Robustness.

• Design and training of a CNN that generalizes well to unseen data.

• Implementation of validation techniques such as early stopping and model checkpointing to avoid overfitting.

• Strong performance on both validation and test datasets, indicating the model's readiness for practical deployment.

Future Work

To further improve the model and enhance its applicability:

• Expand the Dataset: Incorporating more diverse images, including various lighting and weather conditions could improve performance.

• Model Optimization: Experimenting with more complex architectures or pre-trained models may yield better results.

• Deployment: Convert the trained model into a format suitable for mobile or embedded systems using tools like TensorFlow Lite.

• Real-Time Testing: Integrate with a real-time video feed for live road sign detection and classification.

• This project serves as a foundation for building intelligent transportation systems and contributes toward the development of safer autonomous driving technologies.