Road Sign Classification - Using CNN

Prepared by: Akilan J

Batch: DADS November 2024 Batch

1. PROJECT OVERVIEW

This project focuses on developing a deep learning model using Convolutional Neural Networks (CNNs) to classify road signs into 30 categories. The goal is to enhance traffic automation and support autonomous driving systems through automatic image-based sign recognition.

2. OBJECTIVE

The primary objectives of the project are to:

• Automate the classification of road signs using deep learning.

• Improve the scalability and reliability of road sign recognition.

• Deploy the model in a web app for real-time user interaction.

• Understand the impact of preprocessing and model architecture on classification accuracy.

3. DATASET DESCRIPTION

The dataset contains labelled images of various traffic signs. The data distribution and properties are:

Total Images: 743

Classes: 30 (e.g., speed limits, prohibitions, directional signs, warnings)

Labels: Provided in an Excel file.

4. DATA PREPROCESSING

To prepare the images for input into the CNN, the following preprocessing steps were applied:

• Resizing: All images were resized to a fixed dimension which is 160×160 to ensure consistency in input shape.

• ImageDataGenerator: Used Keras's ImageDataGenerator to load and label images based on folder structure.

• Data Splitting: The dataset was split into:

- Training Set: 631 images
- Validation Set: 112 images
- Test Set: 285 images

• Autotuning: Implemented tf.data.AUTOTUNE to prefetch batches during training, improving speed and efficiency.

5. CNN MODEL ARCHITECTURE:

The model was built using TensorFlow/Keras with the following layers:
• Convolutional Layers: Apply filters to extract features using ReLU activation.
• MaxPooling Layers: Downsample feature maps to retain key information.
• Dropout Layers: Reduce overfitting by randomly deactivating neurons.
• Batch Normalization: Stabilize and accelerate training.
• Dense Layers: Map extracted features to final outputs.
• Softmax Layer: Outputs class probabilities for 30 categories.

6. MODEL TRAINING PROCESS

The CNN model was trained using the following configuration:

Parameter Value

Epochs: 30

Optimizer: Adam (learning rate = 0.001)

Loss Function: Sparse Categorical Crossentropy (for integer labels)

Evaluation Metrics: Accuracy, Loss

The model's performance was monitored using training and validation accuracy/loss across the epochs.

7. EVALUATION AND RESULTS

After training, the following results were obtained:

• Training Accuracy: 100%

• Test Accuracy: 59%

Observations:

• High training accuracy suggests the model learned the training data well.

• Good validation accuracy shows generalization to unseen data.

• Lower test accuracy indicates potential overfitting and dataset limitations.

Performance Tools:

• Confusion Matrix: Identified misclassified categories.

• Accuracy Curve: Tracked learning progress.

• Loss Curve: Helped detect overfitting or underfitting.

8. MODEL DEPLOYMENT

To demonstrate practical use, the model was deployed with:

1. Model Persistence: Saved using .h5 format for reuse.
2. Streamlit App: Allowed users to upload images and get predictions.
3. Global Access: Enabled via Ngrok-generated public URL.

9. KEY FINDINGS AND INSIGHTS

- CNNs are effective in extracting features from image data.
- Dataset size constrained generalization.
- Similar signs led to confusion and misclassification.
- Data augmentation was not used, limiting robustness.
- Training stabilized around 20 epochs.

10. CONCLUSION AND FUTURE WORK

The project proves CNNs can be successfully applied to road sign recognition, with high validation accuracy.

Future Enhancements:

- Use data augmentation (rotation, flipping, zoom, brightness).
- Expand dataset for improved diversity and generalization.
- Experiment with advanced CNN architectures (ResNet, InceptionNet, EfficientNet).
- Perform hyperparameter tuning (batch size, dropout, learning rate).
- Apply transfer learning using pretrained models like MobileNet or VGG.