Traffic Sign Recognition with Data Augmentation

📌 Overview

This project implements a **deep learning pipeline** for the recognition of **German Traffic Signs** using **Convolutional Neural Networks (CNNs)**.
The goal is to **accurately classify traffic signs into 43 categories**, a crucial step in developing **autonomous driving systems** and **advanced driver-assistance systems (ADAS)**.

Traffic sign recognition plays a critical role in **road safety**, enabling vehicles to automatically detect and interpret signs such as **speed limits, stop signs, and warnings**.

🎯 Objectives

- Build a **robust classification model** for the German Traffic Sign Recognition Benchmark (GTSRB).
- Apply **data augmentation** to improve generalization.
- Compare baseline and augmented results.
- Evaluate model with **accuracy, precision, recall, and F1-score**.
- Provide a **reproducible training pipeline** for researchers and engineers.

🗁 Dataset

- **Name**: German Traffic Sign Recognition Benchmark (GTSRB)
- **Number of classes**: 43
- **Training images**: ~39,000
- **Test images**: ~12,000
- **Image size**: Resized to 32×32×3

Example Images

Original **Augmented**

Class Distribution

The dataset is **imbalanced**, with some classes having thousands of samples (e.g., speed limits) while others have fewer (e.g., rare signs).

🔧 Methodology

1. Data Preprocessing

- **Resizing** all images to 32×32×3.
- **Normalization** to range [0, 1].
- **One-hot encoding** of labels.

2. Data Augmentation

To improve generalization, the following transformations were applied:

- Random rotations (±20°)
- Zoom (0.8–1.2×)
- Horizontal/vertical shifts (±10%)
- Brightness variation (±20%)
- Shear transformations

- Horizontal flips

3. Model Architecture (CNN)

```
Input: 32x32x3
↓
Conv2D (32 filters, 3x3) + ReLU
↓
Conv2D (64 filters, 3x3) + ReLU
↓
MaxPooling (2x2)
↓
Dropout (0.25)
↓
Conv2D (128 filters, 3x3) + ReLU
↓
MaxPooling (2x2)
↓
Dropout (0.25)
↓
Flatten
↓
Dense (256 units) + ReLU
↓
Dropout (0.5)
↓
Dense (43 units) + Softmax
```

⚙️ Training Setup

- **Loss Function**: Categorical Crossentropy
- **Optimizer**: Adam (`lr=0.001`)
- **Batch Size**: 32 / 64
- **Epochs**: 30–50
- **Callbacks**:
    - `EarlyStopping` (patience=5)

- `ModelCheckpoint` (best model saved)

📊 Results

Accuracy

- **Baseline CNN**: ~92% test accuracy
- **With Augmentation**: **95–97% test accuracy**

Evaluation Metrics

Metric	Score
Accuracy	95%
Precision	94%
Recall	94%
F1-score	94%

Confusion Matrix

📈 Training Curves

🚀 Reproducibility

1. Clone Repo

```
git clone https://github.com/your-username/traffic-signs-classification.git
cd traffic-signs-classification
```

2. Install Dependencies

```
pip install -r requirements.txt
```

3. Run Training Notebook

```
jupyter notebook augmented_traffic_signs.ipynb
```

4. Evaluate Model

```
from tensorflow.keras.models import load_model
model = load_model("best_model.h5")
```

📦 Dependencies

- Python 3.8+
- TensorFlow / Keras
- NumPy
- Matplotlib
- OpenCV
- Scikit-learn

Install with:

```
pip install tensorflow keras numpy matplotlib opencv-python scikit-learn
```

🔮 Future Work

- Apply **transfer learning** with ResNet, VGG, or EfficientNet.
- Experiment with **semi-supervised learning** for rare signs.
- Deploy model using **Flask, FastAPI, or Streamlit**.

- Optimize for **edge devices** (Jetson Nano, Raspberry Pi).

🙌 Acknowledgements

- GTSRB Dataset
- TensorFlow & Keras documentation
- OpenCV tutorials