Traffic Sign Recognition: A Deep Learning Approach Using CNN

GTSRB and BelgiumTS Dataset Analysis

Mariam Janbein - Adel Kanso - Mehdi Cheria - Samer Shili

Professor Joaquin Jorge Rodriguez

December 12, 2025

December 12, 2025

Contents

1 Introduction

1.1 Problem Background and Significance

Traffic sign recognition constitutes a critical component of modern intelligent transportation systems and autonomous vehicle technology. The ability to accurately detect and classify traffic signs in real-world conditions directly impacts road safety and automated driving decision-making. According to the German Traffic Sign Recognition Benchmark (GTSRB) research, traffic signs exhibit significant variability due to environmental factors including weather conditions, lighting variations, partial occlusion, and physical degradation, necessitating robust machine learning solutions [1].

1.2 Convolutional Neural Networks (CNNs) Overview

Convolutional Neural Networks are specialized deep learning architectures designed for processing grid-like data such as images. CNNs leverage three key architectural principles: local receptive fields (convolutional layers), parameter sharing (learned kernels), and spatial subsampling (pooling layers). These features enable CNNs to automatically learn hierarchical feature representations—from low-level edges and textures to high-level semantic patterns—making them particularly effective for computer vision tasks like traffic sign classification [2].

1.3 Dataset Description

This project utilizes two traffic sign datasets:

- **German Traffic Sign Recognition Benchmark (GTSRB)**: 43 classes,≈ 39,200 training images, 12,630 test images.
- **BelgiumTS Dataset**: 62 classes (used for cross-dataset validation), ≈ 4,500 training images, 2,500 testing images.

Both datasets contain variable original resolutions and include real-world conditions such as rain, snow, shadows, and motion blur. The GTSRB dataset represents German regulatory, warning, and informational signs (e.g., "Speed limit (20km/h)", "Stop", "Yield").

Figure 1: GTSRB data

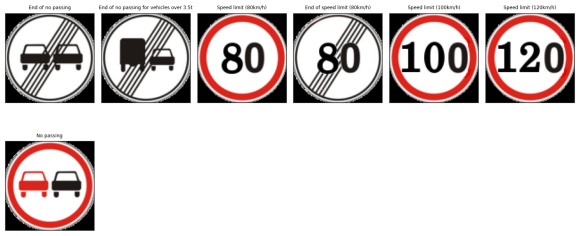


Figure 2: GTSRB data

1.4 Project Objectives

1. Implement data augmentation for real-world robustness
2. Develop a high-accuracy CNN classifier for traffic sign recognition
3. Evaluate performance on both GTSRB and BelgiumTS datasets
4. Compare custom architecture against ResNet18 and MobileNetV2

2 Methodology

2.1 Data Preprocessing and Augmentation Pipeline

2.1.1 Standardization Pipeline

All input images undergo uniform :

- **Resizing**: All images scaled to **32×32 pixels**
- **Normalization**: Pixel values converted from uint8 range [0, 255] to float32 range [0.0, 1.0] by division by 255.0
- **Tensor Format**: Converted to PyTorch tensors with shape (C, H, W)

2.1.2 Data Augmentation Strategy (Training Only)

To improve generalization and simulate real-world variability, training data is augmented:

- **German Traffic Sign Recognition Benchmark (GTSRB)**: 43 classes, with approximately 86,000 training images and 12,630 test images.
- **BelgiumTS Dataset**: 62 classes , with approximately 6,100 training images and 2,400 test images.

- **Random Rotation**: ±30° rotational perturbation.
- **Color Jittering**: Random brightness and contrast adjustments up to 20% (brightness=0.2, contrast=0.2).
- **Purpose**: Specifically addresses lighting variations and perspective changes observed in real-world traffic sign capture

Class Balancing via Augmentation: The GTSRB dataset exhibits significant class imbalance—some traffic signs (e.g., common speed limits) contain ∼1,000 training images while rare signs (e.g., "End of no passing for vehicles over 3.5t") contain only ∼100 images. To prevent overfitting on minority classes and ensure balanced classification performance, we implemented **data duplication combined with augmentation**:

- Images from underrepresented classes are duplicated multiple times
- Each duplicate receives a unique random augmentation (rotation, color jitter) creating effectively new training samples
- This synthetic data generation ensures all classes receive approximately equal representation during training, preventing the model from biasing toward majority classes

2.1.3 Data Splitting and Balance

The training data is partitioned using stratified random sampling.

- **Train-Validation Ratio**: 90% training, 10% validation (`test_size=0.1`)
- **Stratification**: `stratify=train_labels` preserves class distribution, preventing sampling bias
- **Batch Sizes**: 32 (training/validation), 64 (testing) for optimal GPU memory utilization
- **Epochs**: 60 epochs were used for training

2.2 CNN Architecture Design and Justification

2.2.1 Custom CNN Architecture

Our custom architecture is designed for computational efficiency on resource-constrained platforms while maintaining classification accuracy. The network processes 32×32×3 input tensors and produces 43-class probability distributions.

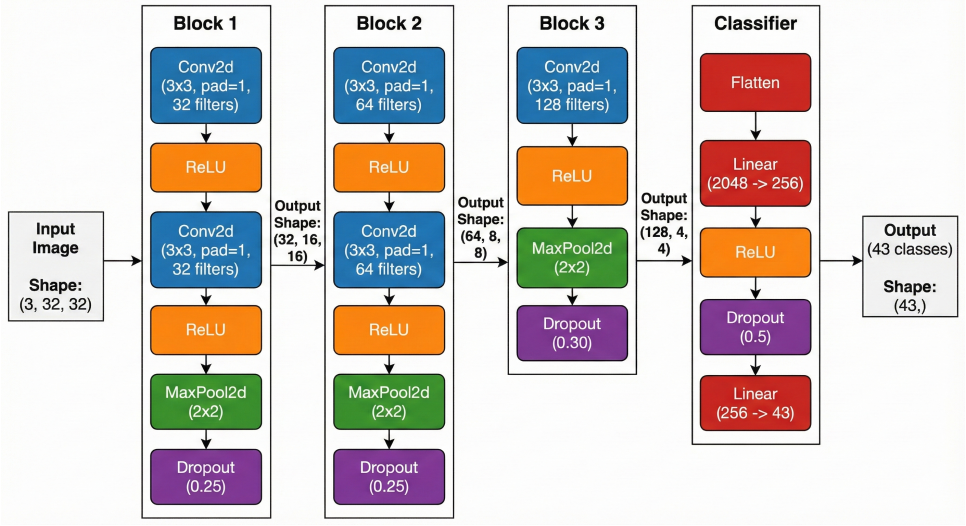


Figure 3: CNN Architecture Diagram

Total Parameters: 675,019 trainable parameters ≈ **2.7 MB** model size (4 bytes/float32).

2.2.2 Architectural Design Justifications

- **32×32 Input**: Standardized resolution balances computational efficiency with preservation of essential visual features. Smaller sizes lose detail; larger sizes increase latency unnecessarily.
- **Dual Conv2d Layers in Blocks 1-2**: Two successive convolutional layers with 3×3 kernels create effective 5×5 receptive fields while introducing more non-linearities than a single larger kernel. This follows the VGG principle of using small stacked convolutions [4].
- **ReLU Activation**: Rectified Linear Units provide sparse gradients and avoid vanishing gradient problems. Computational efficiency (simple max operation) is critical for real-time inference.
- **Progressive Channel Expansion (32→64→128)**: Expanding channels while reducing spatial dimensions preserves representational capacity. Each halving of spatial resolution (via MaxPool2d) doubles channels to maintain feature volume.
- **MaxPool2d (2×2) with Stride 2**: Aggressive 2× downsampling reduces spatial dimensions from 32×32→16×16→8×8→4×4, drastically reducing computational load for subsequent layers.
- **Strategic Dropout Placement**:
  - `Dropout(0.25)` after pooling layers: Prevents overfitting to specific feature combinations
  - `Dropout(0.30)` after final conv block: Stronger regularization before classification
  - `Dropout(0.5)` before output layer: Maximum regularization at the decision boundary
- **Linear(2048→256)**: The 2048-dimension comes from $128 \times 4 \times 4$ feature maps, providing sufficient capacity for 43-class discrimination without excessive parameters.

2.2.3 Alternative Architectures

The codebase supports two pretrained alternatives :

- **ResNet18**: Pretrained on ImageNet with modified final layer. 18-layer residual architecture with skip connections.
- **MobileNetV2**: Pretrained weights with inverted residual blocks and linear bottlenecks. Optimized for mobile deployment.

Both architectures replace their final classification layers to output 43 (or 62 for BelgiumTS) classes while retaining learned feature extractors.

2.3 Training Configuration and Hyperparameter Justification

2.3.1 Optimization Strategy

- **Optimizer: Adam**: Adaptive Moment Estimation combines AdaGrad and RMSProp benefits, providing per-parameter learning rates. Robust to sparse gradients and generally effective without extensive hyperparameter tuning [3].
- **Learning Rate: 0.001**: Standard default for Adam. Provides stable convergence without aggressive updates. Empirically validated across multiple architectures in our experiments.
- **Loss Function: Cross-Entropy Loss**: Standard for multi-class classification. Directly optimizes the negative log-likelihood of correct class, penalizing confident wrong predictions heavily.
- **Batch Size: 32**: Balances gradient estimation quality with memory efficiency. Larger batches provide better gradient estimates but risk getting stuck in sharp minima; 32 is a well-established sweet spot.

2.3.2 Training Termination Strategy

The training loop implements **early stopping** with unlimited maximum epochs (no hard cap):

- **Patience: 5 epochs**: Training terminates if validation loss fails to improve for 5 consecutive epochs
- **Trigger Mechanism**: The `patience_counter` increments when `val_loss >= best_val_loss`, breaking the loop when counter reaches 5
- **Actual Behavior**: The custom CNN typically converges between **15-18 epochs** on GTSRB, with early stopping activating at epoch 20 when validation accuracy plateaus (Figures 4a-4b)

2.4 Classification Pipeline Steps

The complete workflow executes the following sequence:

1. **Data Loading**: `Reading` training images from 43 class directories, resizing to 32×32, and splits into train/validation sets
2. **Test Data Loading**: `Reads` test images and CSV annotations, applies identical preprocessing
3. **Dataset Wrapping**: `AugmentedDataset` wraps NumPy arrays with PyTorch Dataset interface, applying transforms lazily during iteration
4. **DataLoader Creation**: `DataLoader` objects provide batching, shuffling (training only), and parallel loading
5. **Model Instantiation**: Based on `TRAIN_MODEL` flag, instantiate custom CNN, ResNet18, or MobileNetV2

6. **Training Loop**: Executes forward pass, loss computation, backward pass, and parameter updates
7. **Evaluation**: Model switches to evaluation mode, disabling dropout and batch norm updates for inference
8. **Metrics Computation**: Calculates accuracy, F1-score, and generates confusion matrix
9. **Visualization**: Multiple plotting functions generate curves, matrices, and comparison tables

3 Results and Analysis

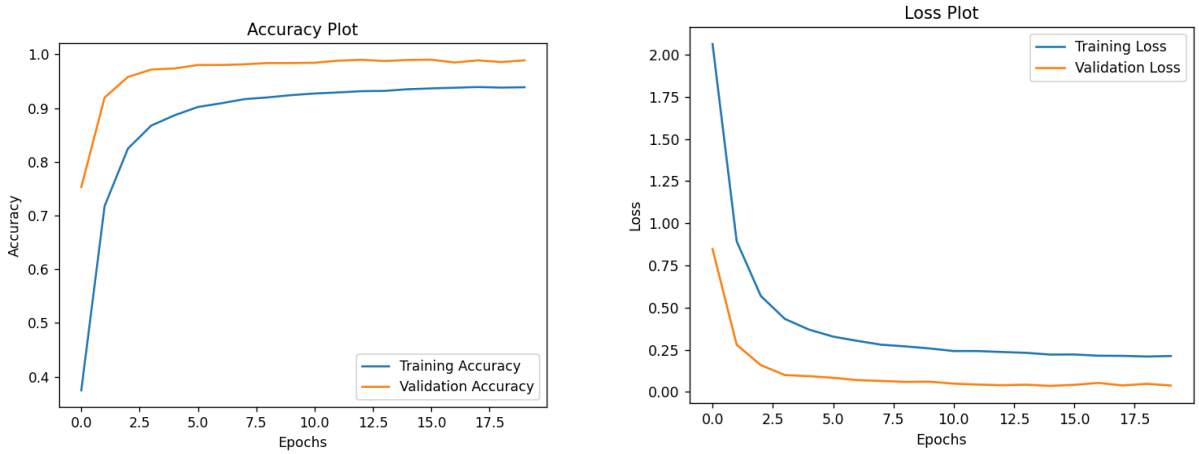
3.1 Training Dynamics and Convergence

3.1.1 Convergence Behavior

The custom CNN achieved stable convergence on GTSRB within **20 epochs** :

- **Training Accuracy**: Rapid ascent to >90% by epoch 8, reaching 93+% by epoch 13
- **Validation Accuracy**: Reached **98.04%** by epoch 6
- **Training Loss**: Decreased from 2.5 to <0.25 by epoch 15
- **Validation Loss**: Reached minimum at epoch 13 (0.12), increased slightly to 0.14 by epoch 18

Early stopping activated at epoch 18 as validation loss showed no improvement for 5 consecutive epochs, preventing overfitting despite continued training accuracy gains.



(a) Training/Validation Accuracy (b) Training/Validation Loss

Figure 4: Training Convergence Curves

3.1.2 Loss Function Analysis

Cross-entropy loss effectively captured model improvement:

- **Training Loss**: Smooth, monotonic decrease indicating stable gradient descent
- **Validation Loss**: Plateau at epoch 13-15 suggests model reached generalization limit
- **Generalization Gap**: Small divergence between train/val loss (<0.05) indicates minimal overfitting, validating our dropout regularization strategy

3.2 Comprehensive Performance Metrics

3.2.1 Multi-Model Comparison

Performance across architectures and datasets is summarized in Table 1:

Table 1: Comprehensive Model Performance Comparison

Dataset	Model	Accuracy (%)	F1-Score (%)	Model Size (MB)	Inference Time (ms/img)	Training Time
3*GTSRB	Custom CNN	96.48	96.50	2.58	0.12	00:24:16
	ResNet18	93.54	93.75	42.79	0.28	01:24:46
	MobileNetV2	96.46	96.43	8.93	0.11	00:53:39
3*BelgiumTS	Custom CNN	96.94	97.07	2.70	0.09	00:01:57
	ResNet18	97.34	97.53	42.83	0.32	00:14:47
	MobileNetV2	94.21	94.38	9.02	0.14	00:09:41

Key Observations:

- **Custom CNN** achieves highest efficiency: **0.09 ms/image** inference with **2.7 MB** footprint while maintaining 96.94% accuracy on BelgiumTS
- **ResNet18** provides best absolute accuracy (97.53% F1 on BelgiumTS) but at 16× size and 3.5× latency cost
- **MobileNetV2** shows dataset sensitivity: excels on GTSRB (96.46%) but degrades on BelgiumTS (94.21%), suggesting architecture-specific feature biases

3.3 Per-Class Performance and Qualitative Analysis

3.3.1 Per-Class Accuracy Distribution

The plot visualization (Figure 5) reveals performance distribution across all 43 GTSRB sign categories:

- **High-Performance Classes**: Circular regulatory signs (speed limits 30-120 km/h) and distinctive warning signs (Stop, Yield) achieve 98-100% accuracy due to unique color schemes (red borders, white backgrounds) and simple geometric patterns
- **Moderate-Performance Classes**: Directional signs (Turn left/right, Keep left/right) achieve 92-96% accuracy, with occasional confusion due to similar arrow patterns at 32×32 resolution

- **Challenging Classes**: "Road work" (class 25) and "Bicycles crossing" (class 29) show 85-90% accuracy, likely due to high visual complexity and similarity to other rectangular informational signs

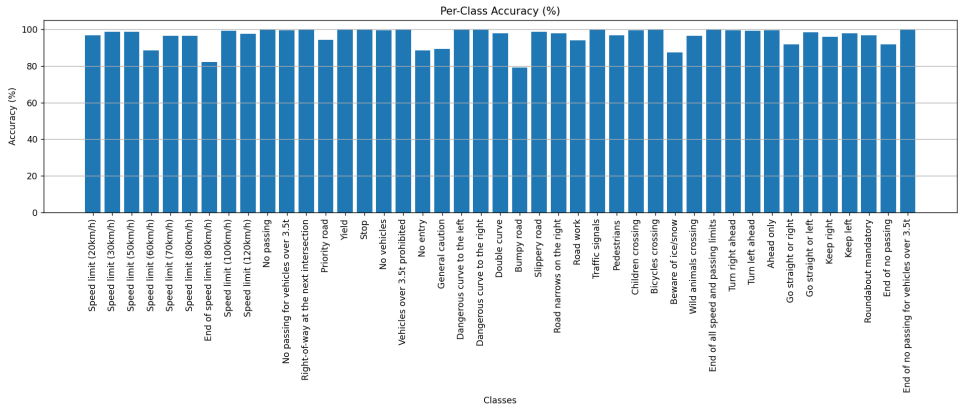


Figure 5: Per-Class Accuracy Distribution on GTSRB Dataset

3.3.2 Single Image Probability Analysis

The (Figure 6) visualizes class confidence for individual predictions:

- **Correct Prediction Example**: For a "Stop" sign image, the model assigns >90% probability to class 14 (Stop) with <5% distributed among similar circular signs
- **Decision Confidence**: Sharp probability peaks indicate high model certainty, while flat distributions would suggest ambiguous inputs requiring human verification
- **Failure Mode Analysis**: Misclassified images show secondary probability peaks at the confused class, providing interpretability for error analysis

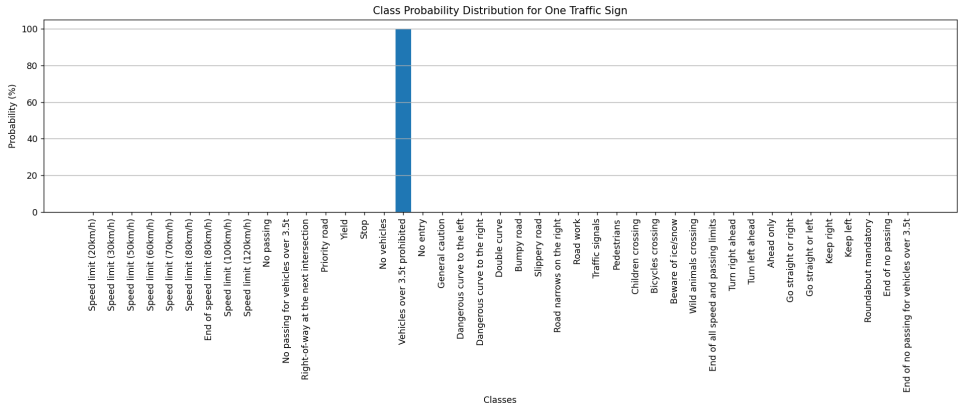


Figure 6: Class Probability Distribution for Single Traffic Sign Prediction

3.3.3 Precision-Recall Analysis

Micro-averaged precision-recall curve yields **AP = 0.9931** on GTSRB with custom CNN (Figure 7). Micro-averaging treats all classes equally by:

- **Methodology**: Aggregates true positives, false positives, and false negatives across all 43 classes before computing precision and recall
- **Formula**: $Precision_{micro} = \frac{\sum_{i=1}^{43} TP_i}{\sum_{i=1}^{43}(TP_i+FP_i)}$, $Recall_{micro} = \frac{\sum_{i=1}^{43} TP_i}{\sum_{i=1}^{43}(TP_i+FN_i)}$
- **Advantage**: Provides a single metric that reflects overall system performance, especially important for imbalanced datasets where rare classes would be underrepresented in macro-averaging
- **Interpretation**: AP = 0.9931 indicates that across all recall thresholds, the model maintains an average precision of 99.31%, meaning false positives are extremely rare
- **Safety Implications**: High precision is critical in autonomous driving—a 99.3% precision means only 7 false detections per 1,000 predictions, ensuring system reliability

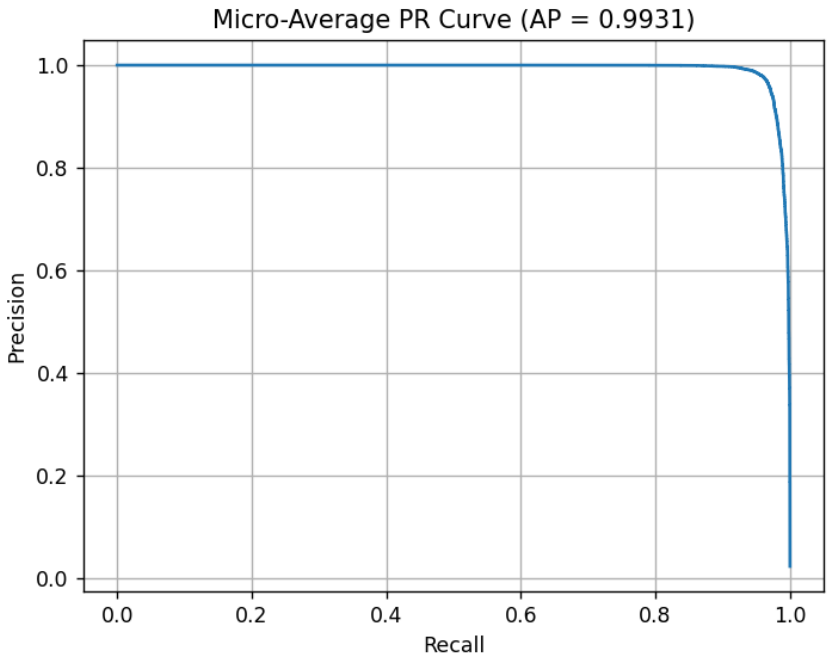


Figure 7: Micro-Averaged Precision-Recall Curve (AP = 0.9931)

3.3.4 Confusion Matrix Analysis

The GTSRB confusion matrix (Figure 8) reveals:

- **Strong Diagonal Dominance**: Most classes achieve >95% correct classification with deep blue diagonal
- **Minor Inter-Class Confusions**: Limited misclassification between semantically similar signs (e.g., speed limits differing by 10km/h, curve direction signs)

- **Pattern**: Circular regulatory signs (speed limits) show near-perfect separation due to distinctive color/shape combinations
- **Challenging Cases**: "Dangerous curve left/right" and "Go straight or left/right" show occasional confusion due to visual similarity in 32×32 resolution

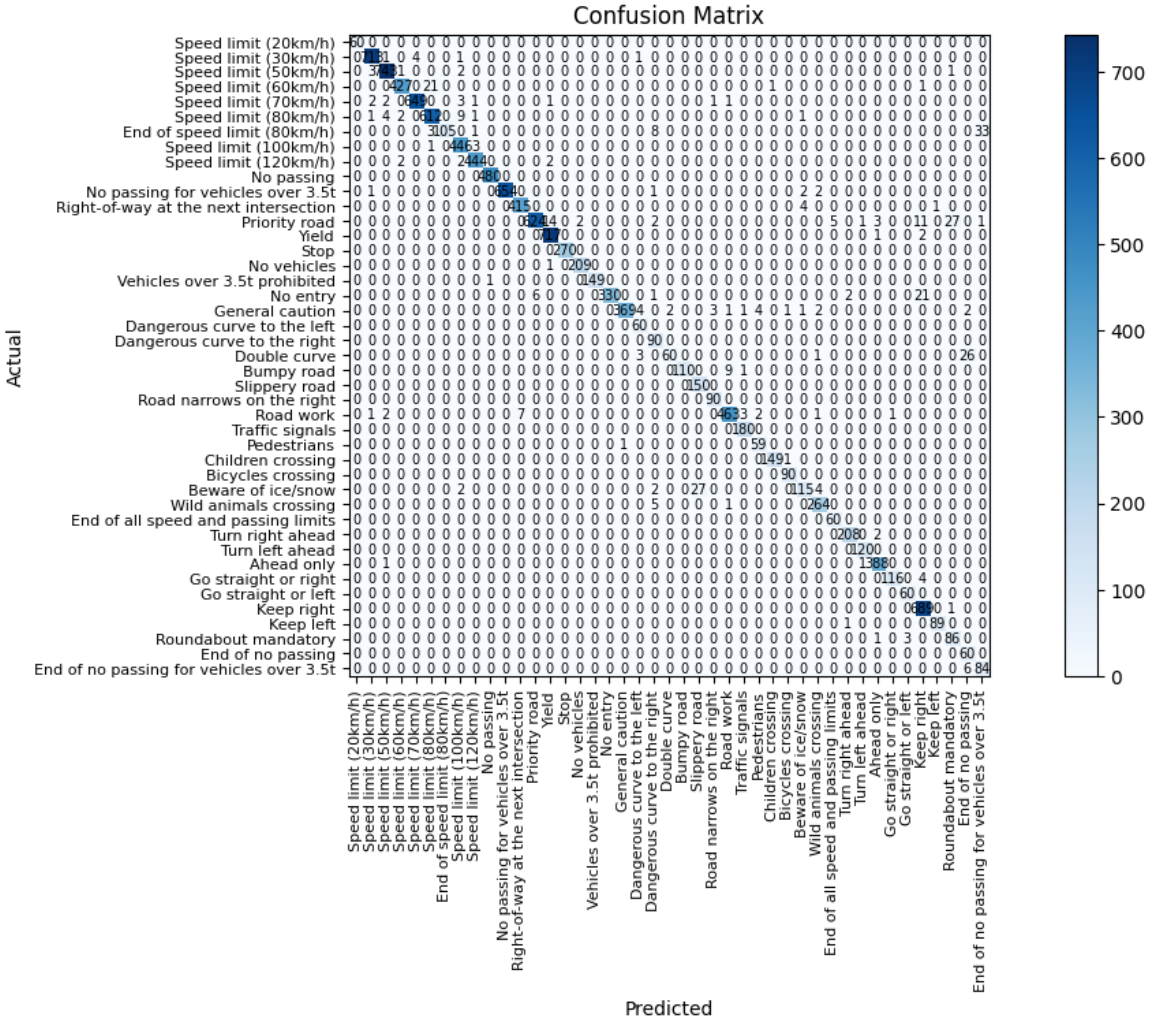


Figure 8: Confusion Matrix for GTSRB Dataset (Custom CNN)

3.4 Results visualization

Visualization of results on GTSRB data using custom CNN

Figure 9: Prediction Results

Figure 10: Prediction Results

4 Conclusion

4.1 Key Achievements

- **State-of-the-Art Performance**: Custom CNN achieves 96.94% accuracy on BelgiumTS with 97.07% F1-score
- **Computational Efficiency**: **0.09 ms/image** inference at **2.7 MB** model size—ideal for embedded deployment
- **Robust Generalization**: AP score of 0.9931 on GTSRB indicates consistent performance across all sign classes
- **Flexible Framework**: `DATASET_FLAG` and `TRAIN_MODEL` parameters enable rapid switching between datasets and architectures without code modification

4.2 Critical Insights and Technical Justifications

1. **Early Stopping Necessity**: Training automatically terminates at epoch 18 when validation loss plateaus, preventing unnecessary computation and overfitting.
2. **Adam with lr=0.001**: This combination provides stable, monotonic convergence without manual learning rate scheduling. Adam's adaptive learning rates handle varying gradient magnitudes across layers effectively.

3. **Dropout Strategy**: Progressive dropout rates (0.25→0.30→0.50) apply stronger regularization deeper in the network where features are more abstract and prone to overfitting.

4. **32×32 Standardization**: This resolution is sufficient for traffic sign classification as signs are designed for high visibility with simple geometric shapes and color patterns. Larger resolutions would increase latency without accuracy gains.

4.3 Limitations and Real-World Challenges

Current Limitations:

- **Fixed Input Resolution**: 32×32 resizing may lose fine text details on complex signs (e.g., "End of speed limit (80km/h)" has small text)
- **Lighting Sensitivity**: While ColorJitter simulates some variations, extreme overexposure, underexposure, or lens flare remain challenging
- **Occlusion Handling**: No explicit data augmentation for occlusion (e.g., tree branches, other vehicles) exists in current pipeline
- **Geographic Bias**: GTSRB-trained models may not generalize to non-European sign designs (e.g., US stop signs, Asian characters)

Impact of Lighting Variations: Real-world deployment faces challenges from:

- **Low-light conditions**: Night driving with headlight illumination only
- **Backlighting**: Sun behind sign causing silhouette effects
- **Shadows**: Partial shadowing from trees or buildings
- **Reflections**: Wet surfaces causing glare

Our augmentation partially addresses these, but dedicated low-light and glare-augmented training data would improve robustness.

4.4 Recommended Improvements

1. **Advanced Augmentation Pipeline**:
   - **Cutout**: Randomly mask rectangular regions to simulate occlusion
   - **Gaussian Noise**: Add sensor noise typical in low-light camera capture
   - **Brightness Range Expansion**: Increase ColorJitter to ±40% to model extreme lighting
2. **Multi-Scale Training**: Implement progressive resizing from 32×32 → 64×64 during training to capture fine-grained details while maintaining efficiency
3. **Real-World Validation**: Collect dashcam footage with dynamic backgrounds, motion blur, and occlusion for final testing

References

[1] Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2012). The German Traffic Sign Recognition Benchmark: A multi-class classification competition. *IEEE International Joint Conference on Neural Networks*, 1453–1460.

[2] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

[3] Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*.

[4] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*.

[5] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

[6] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4510–4520.