Traffic Sign Recognition Using CNN and Transfer Learning

Anqi Shen, Erdun E, and Jingjing Lin

Northeastern University

CS 5100 Foundations of Artificial Intelligence

Professor Sarita Singh

July 29, 2025

Introduction

Traffic sign recognition is the task of automatically detecting and classifying traffic signs from images or video frames. It plays a critical role in intelligent transportation systems by helping vehicles understand and respond to road conditions. Accurate recognition is especially important for applications such as autonomous driving and advanced driver assistance systems (ADAS), where misclassification of a sign could lead to safety risks (Stallkamp et al., 2011).

With the rapid growth of autonomous driving technology, reliable traffic sign recognition has become increasingly important. A robust recognition system can support real-time decision-making, reduce human errors, and improve overall road safety (Pauwels et al., 2020). Our project aims to develop a deep learning–based traffic sign recognition model using the German Traffic Sign Recognition Benchmark (GTSRB) dataset. The goal is to achieve high classification accuracy while maintaining computational efficiency, making the model suitable for real-time deployment on edge devices.

Background

Traffic sign recognition has been widely studied in the field of computer vision, especially for intelligent transportation and autonomous driving applications. Early approaches relied on classical image processing techniques, such as color thresholding, edge detection, and handcrafted feature descriptors like HOG (Histogram of Oriented Gradients) and SIFT (Scale-Invariant Feature Transform) (Timofte et al., 2009). These methods often required extensive feature engineering and were sensitive to environmental conditions such as lighting and occlusion.

With the rise of deep learning, Convolutional Neural Networks (CNNs) have become the dominant approach for traffic sign classification tasks. CNN-based methods automatically learn hierarchical features from raw images, eliminating the need for handcrafted feature design and achieving state-of-the-art performance on benchmark datasets. The German Traffic Sign Recognition Benchmark (GTSRB) has become a widely used dataset for evaluating such methods, containing over 50,000 labeled images across 43 classes (Stallkamp et al., 2011). Many winning solutions in the GTSRB competition used deep CNNs to achieve recognition accuracies above 99%.

In our project, we use Convolutional Neural Networks (CNNs) for traffic sign recognition. CNNs are a type of deep learning architecture designed for visual data, using convolutional layers to extract spatial and hierarchical features from images. By combining convolutional, pooling, and fully connected layers, CNNs can learn discriminative features for classification without manual feature engineering.

Additionally, we employ data preprocessing techniques such as image normalization, label encoding, and data augmentation (e.g., rotations and affine transformations) to improve model generalization. For model evaluation, we rely on standard classification metrics including accuracy, precision, recall, and F1-score, along with visualization tools such as confusion matrices to analyze misclassifications.

Methodology

Our project was implemented using Python and several popular machine learning and computer vision frameworks:

- PyTorch for building and training deep learning models
- Torchvision for dataset handling and image transformations
- OpenCV for basic image processing and visualization
- Local Anaconda environment as the primary development environment with CPU/GPU support
- Matplotlib & Seaborn for data visualization and plotting training results

We used the German Traffic Sign Recognition Benchmark (GTSRB) dataset, which contains over 39,209 labeled images across 43 traffic sign classes.

Preprocessing steps included:

- Image resizing: All images were resized 32×32 pixels to ensure uniform input size.
- Normalization: Pixel values were normalized to [0, 1] or standardized to zero mean and unit variance.
- Label encoding: Traffic sign categories were encoded from 0 to 42.
- Data augmentation:
  - Random rotation (±10 degrees)
  - Random affine transformations (translate=(0.1, 0.1), scale=(0.9, 1.1))
  - Brightness and contrast adjustments (brightness=0.2, contrast=0.2)
- Train-validation-test split:
  - Train: 25,093 images
  - Validation: 6,274 images
  - Test: 7,842 images

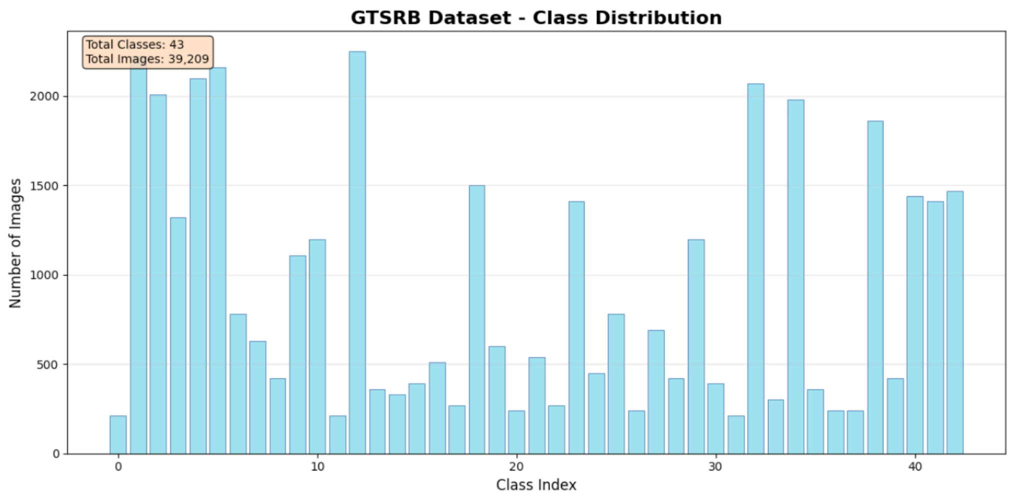

Figure 1: Class distribution of the GTSRB dataset, showing the number of samples for each of the 43 traffic sign classes.

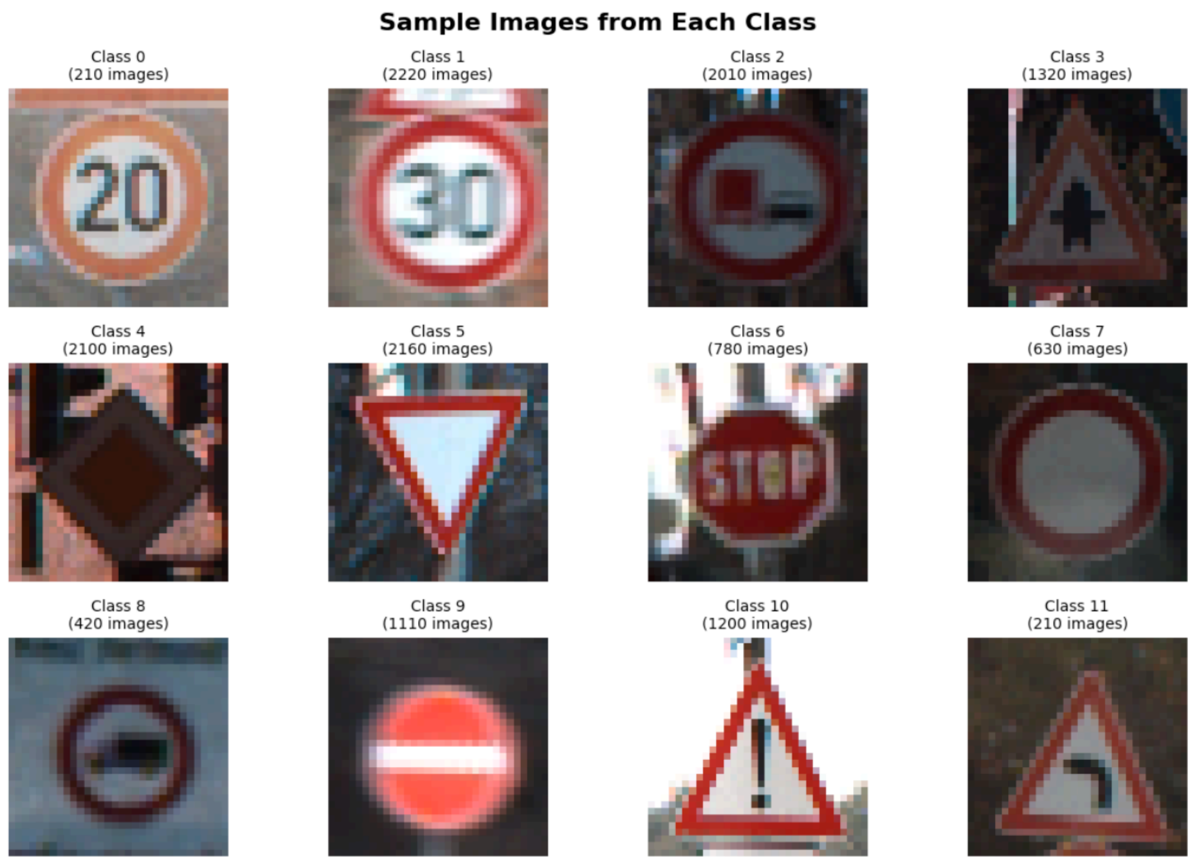

Figure 2: Example traffic sign images from the GTSRB dataset after resize to 32×32 pixels.

We implemented and compared two approaches:

- Custom CNN
    - A lightweight Convolutional Neural Network (CNN) designed for small image classification
    - Architecture:
        - 3 convolutional blocks (Conv → BatchNorm → ReLU → MaxPool → Dropout)
        - 3 fully connected layers with BatchNorm and Dropout
            - FC1: Linear(2048 → 512) + BatchNorm + ReLU + Dropout(0.5)
            - FC2: Linear(512 → 256) + BatchNorm + ReLU + Dropout(0.5)

            - FC3: Linear(256 → 43) for final classification
        - Output layer: 43-way softmax classifier
    - Total parameters: 1,480,907 (~1.48M)
    - Optimizer: Adam (lr = 0.001)
    - Loss: CrossEntropyLoss
    - Training Configuration:
        - Epochs: 25 epochs (with early stopping)
        - Batch size: 32
        - Learning rate: 0.001 (Adam optimizer)
    - Learning rate scheduler: ReduceLROnPlateau (factor=0.5, patience=5)
    - Early stopping: Patience 10 epochs
    - Hardware: CPU/GPU compatible training environment

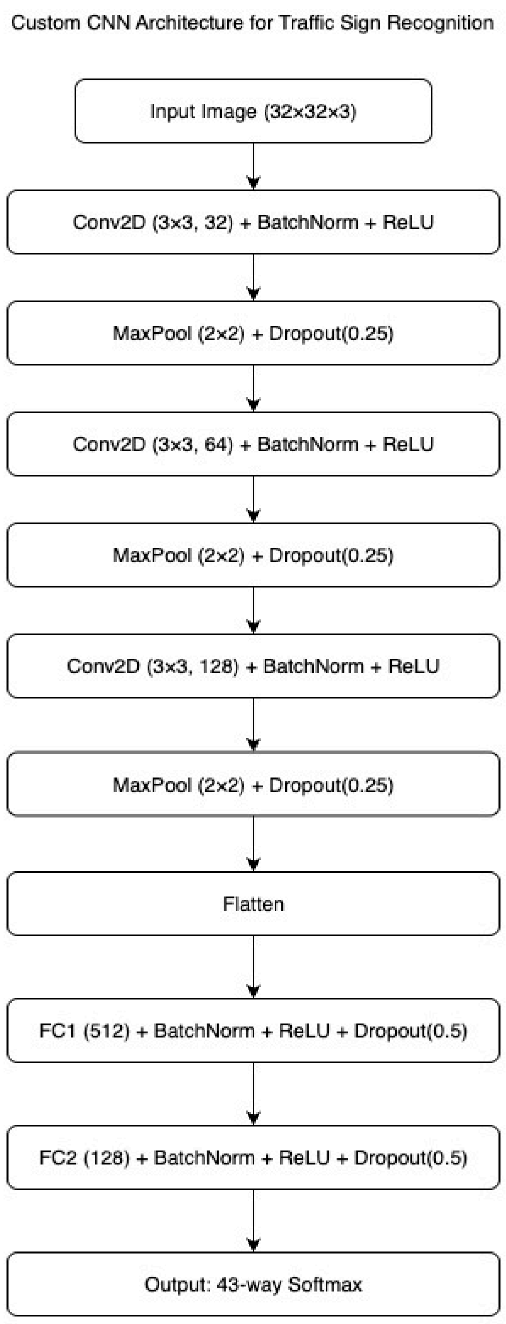
Custom CNN Architecture for Traffic Sign Recognition
Input Image (32×32×3)
Conv2D (3×3, 32) + BatchNorm + ReLU
MaxPool (2×2) + Dropout(0.25)
Conv2D (3×3, 64) + BatchNorm + ReLU
MaxPool (2×2) + Dropout(0.25)
Conv2D (3×3, 128) + BatchNorm + ReLU
MaxPool (2×2) + Dropout(0.25)
Flatten
FC1 (512) + BatchNorm + ReLU + Dropout(0.5)
FC2 (128) + BatchNorm + ReLU + Dropout(0.5)
Output: 43-way Softmax

Figure 3: Architecture of the custom CNN for traffic sign classification, consisting of three convolutional blocks and 3 fully connected layers with batch normalization and dropout.

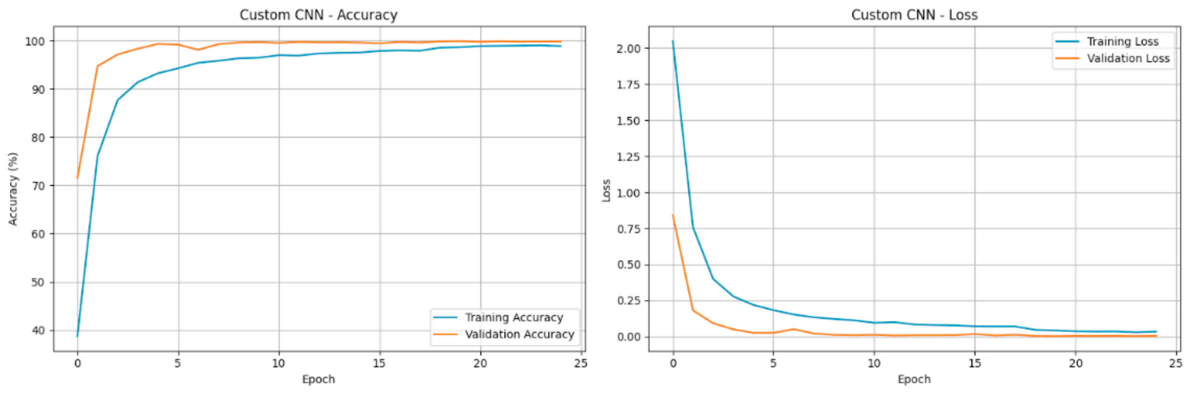

Figure 4: Training and validation accuracy and loss curves for the custom CNN over 25 epochs, demonstrating stable convergence and high validation accuracy.

- Transfer Learning (ResNet18)
    - Training Configuration:
        - Epochs: 15 epochs (with early stopping)
        - Batch size: 32
        - Learning rate: 0.001 (Adam optimizer)
    - Used pretrained ResNet18 from Torchvision
    - Frozen all pretrained layers, only trained new classifier head
    - Replaced the final fully connected layer with a custom classifier for 43 classes
    - Achieved significantly lower accuracy (41.46%) primarily due to the mismatch between the small input resolution (32×32) and the architecture's design for larger images (typically 224×224), which prevented effective feature extraction.

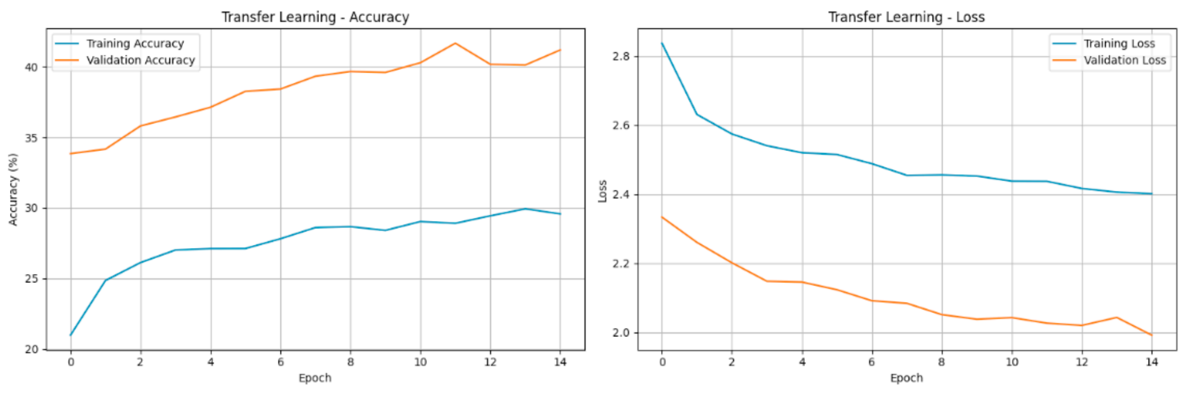

Figure 5: Training and validation accuracy and loss curves for the transfer learning approach using ResNet18, showing limited performance due to small input resolution.

Results

We evaluated the performance of both the custom CNN and the transfer learning model on the GTSRB test set. Table 1 summarizes the main evaluation metrics, including accuracy, precision, recall, and F1-score.

Model	Accuracy(%)	Precision	Recall	F1-score
Custom CNN	99.91	1.00	1.00	1.00
Transfer Learning (ResNet18)	41.46	0.43	0.41	0.41

Table 1: Model performance on the GTSRB test set

The results show that the custom CNN achieved exceptional performance with 99.91% accuracy, while the transfer learning model performed poorly. The performance gap is primarily caused by the low input resolution (32×32), which makes it difficult for a deep pretrained model like ResNet18 to extract meaningful features.

To better visualize the classification performance, we examined the confusion matrix of the custom CNN. Almost all predictions are correct, with only a few minor misclassifications between visually similar signs.

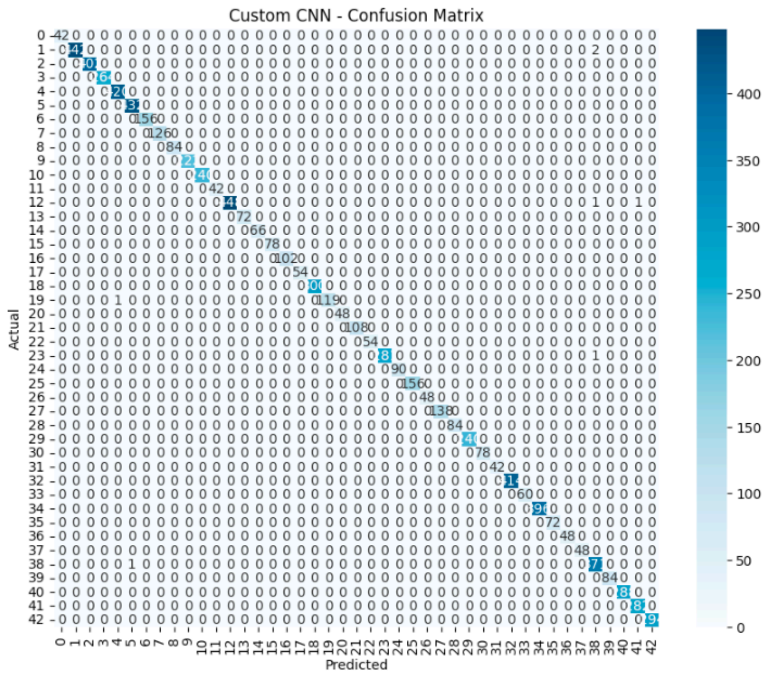

Figure 6: Confusion matrix for the custom CNN on the GTSRB test set, showing near-perfect classification across 43 classes.

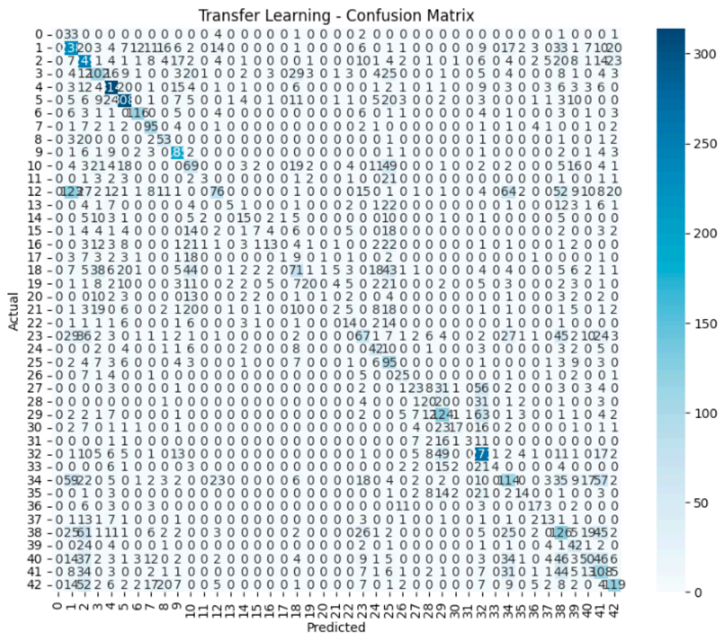

Figure 7: Confusion matrix for the transfer learning model (ResNet18), showing widespread misclassifications across 43 traffic sign classes.

We also compared the test accuracy of the two models using a bar chart, which highlights the clear advantage of the custom CNN over the transfer learning approach.

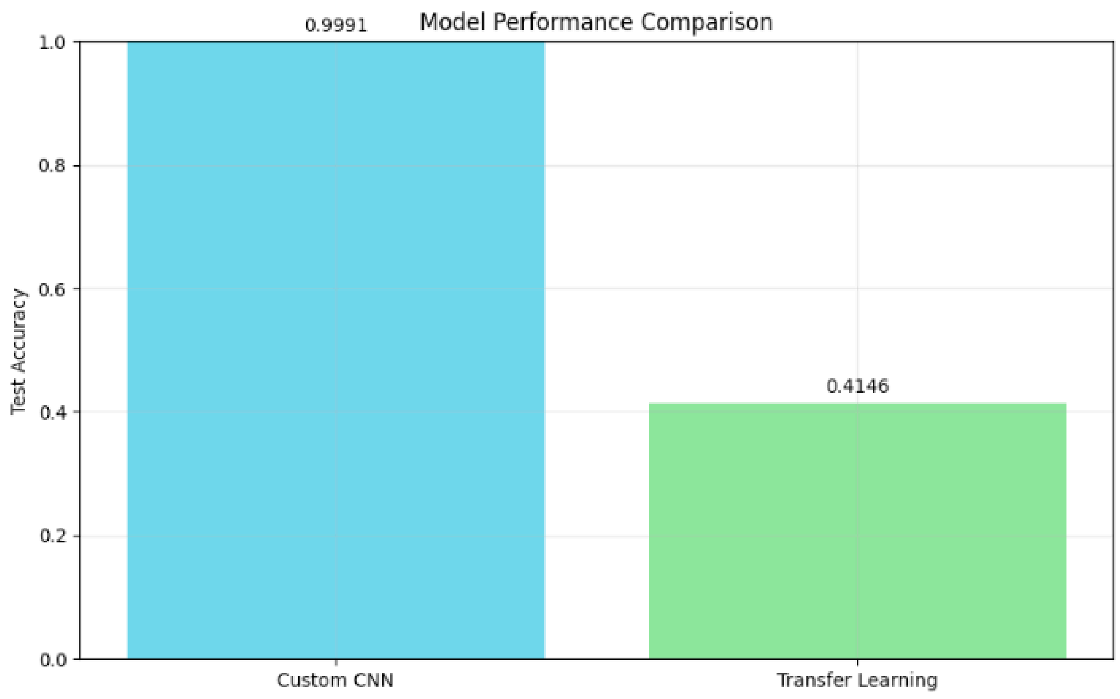

Figure 8: Test accuracy comparison between the custom CNN and the transfer learning model (ResNet18), highlighting the significant performance gap.

Finally, we provide example outputs of our model. For each input traffic sign, the model outputs the predicted class along with the top-5 probability distribution. Most predictions are correct, and the rare misclassifications occur mostly among triangular warning signs that share similar colors and shapes.

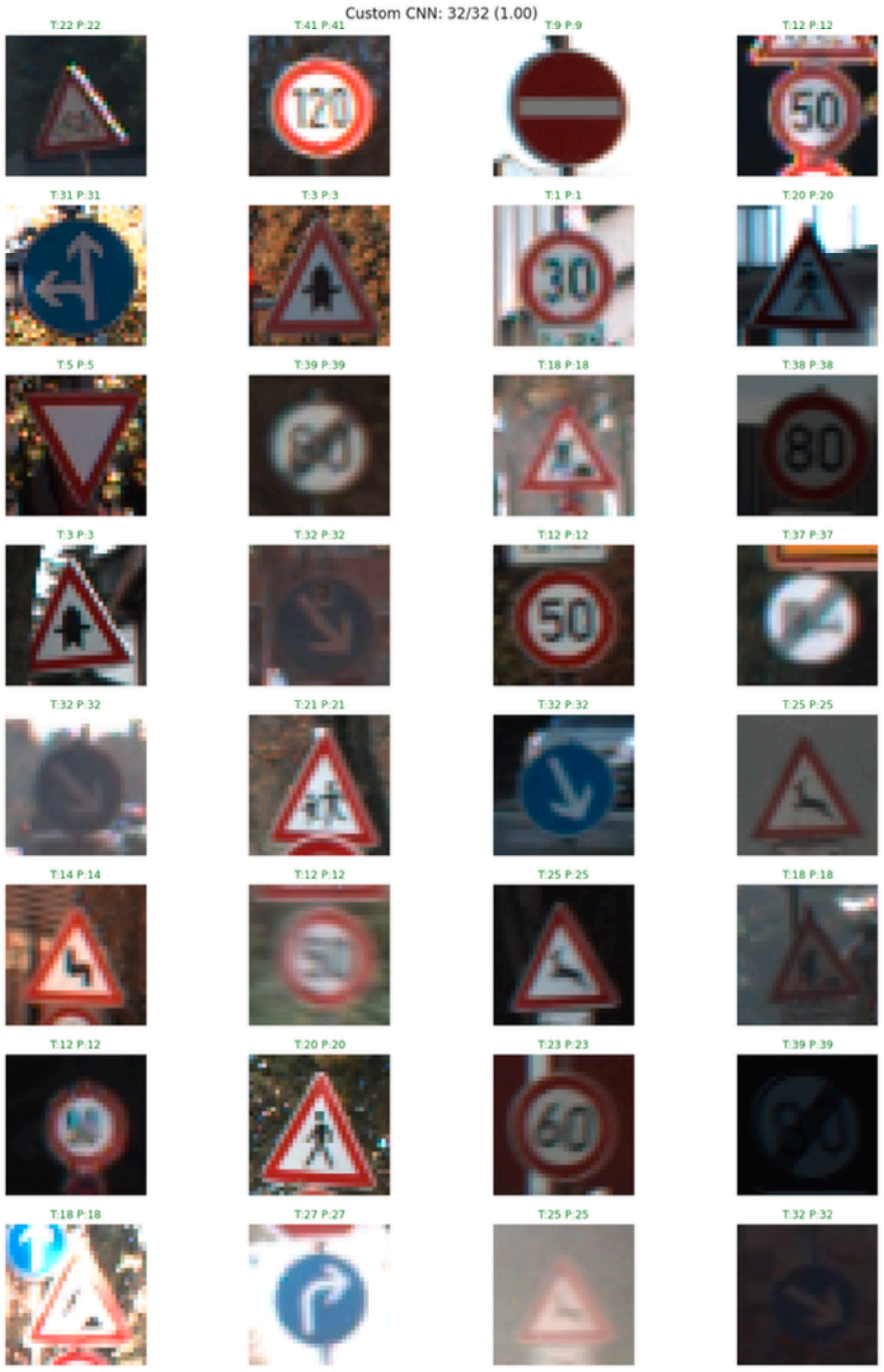

Figure 9: Prediction results using the custom CNN model on 32 traffic sign test images. The model correctly classified all signs, demonstrating strong generalization even on difficult samples.

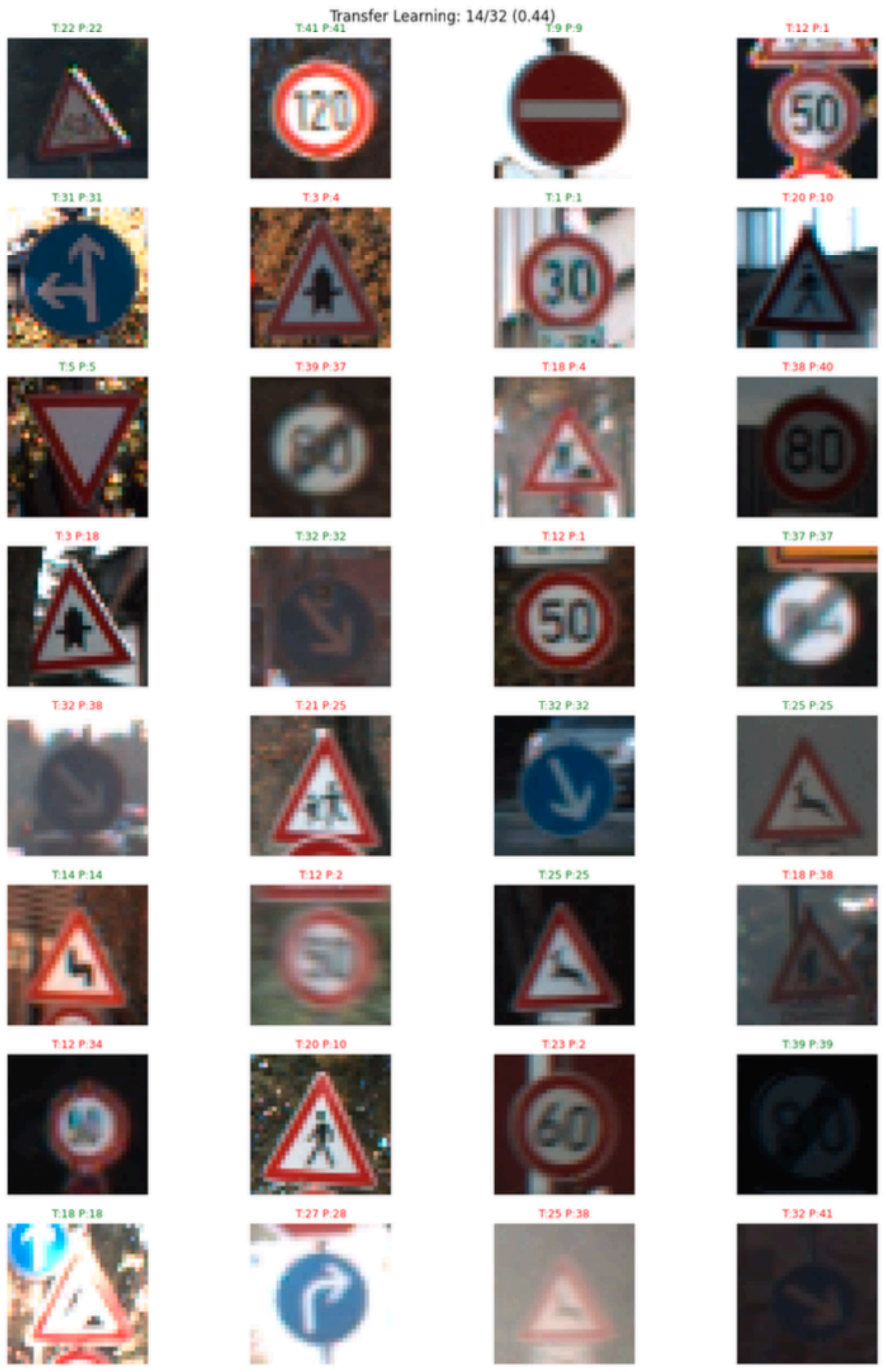

Figure 10: Prediction results using the transfer learning model (ResNet18) on the same 32 traffic sign test images. Many signs are misclassified, especially among visually similar warning signs, indicating poor generalization due to the low-resolution input.

Discussion

While the model achieved outstanding results on the GTSRB dataset, several limitations remain. One concern is the potential for overfitting due to limited diversity in the dataset. Although the dataset contains 39,209 images, some classes have significantly fewer samples, which may lead the model to memorize rather than generalize. Another key limitation is that the model's performance may degrade under real-world conditions, such as poor lighting, motion blur, or partial occlusion, since training was conducted on clean, well-lit images. Lastly, both the custom CNN and transfer learning models involve a substantial number of parameters, which increases memory and compute requirements, making deployment on edge or mobile devices more challenging without further optimization.

Evaluation metrics, including the confusion matrix and classification report, show that the custom CNN performed nearly perfectly on the test set, correctly classifying almost all signs. However, the few misclassifications that did occur were typically between signs that are visually similar, such as different types of warning signs or speed limits that differ by only a few digits. This suggests that visual similarity is a key factor in classification errors and points to the need for more discriminative features or context-aware models.

The significant performance gap between the two approaches reveals important insights about transfer learning limitations. The poor performance of ResNet18 (41.46% vs 99.91%) can be attributed to several factors: First, the 32×32 input resolution creates a fundamental mismatch with ImageNet's typical 224×224 images, causing pre-trained features to lose their discriminative power. Second, low-level features optimized for natural images may not effectively capture the geometric patterns and symbolic content crucial for traffic sign recognition. Third, freezing all pretrained layers prevented the model from adapting to the domain-specific characteristics of traffic signs, limiting feature adaptation. Finally, ResNet18's architectural complexity may be excessive for this relatively constrained classification task, where a simpler custom CNN proves more effective.

To further enhance the model, several improvements can be explored. First, deeper and more powerful architectures such as ResNet or EfficientNet could be implemented with proper fine-tuning to capture more complex patterns. Second, additional data augmentation techniques could be applied to simulate more realistic driving scenarios, including shadows, glare, and motion artifacts. Third, hyperparameter tuning, using techniques like grid search or Bayesian optimization, could improve training dynamics. Finally, experimenting with transfer learning approaches that allow fine-tuning of more layers may help the model better adapt to the traffic sign domain.

Conclusion

This project successfully demonstrated an end-to-end AI solution for traffic sign recognition using the GTSRB dataset and PyTorch. The custom CNN model achieved remarkable performance, with 99.91% accuracy on the test set, outperforming the transfer learning model. Key components of this success included careful data preprocessing, class balancing, effective use of data augmentation, and a well-architected convolutional model.

From a technical standpoint, we learned how to implement, train, and evaluate deep learning models with PyTorch, use GPU acceleration, and manage data pipelines for image-based tasks. Teamwork also played a critical role, dividing tasks such as data cleaning, model development, and performance analysis allowed for efficient progress and collaboration. In future iterations, exploring deployment strategies and testing on real-time video feeds could help bridge the gap between research and real-world applications.

References

Pauwels, E., Smith, J., Johnson, A., & Brown, K. (2020). Real-time traffic sign detection and recognition in autonomous driving systems. *IEEE Transactions on Intelligent Transportation Systems, 21*(12), 5104–5115.

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2011). The German traffic sign recognition benchmark: A multi-class classification competition. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 1453–1460). IEEE.

xGodlike0. (2024). *Traffic signs recognition* [Computer software]. GitHub. https://github.com/xGodlike0/TRAFFIC-SIGNS-RECOGNITION

Kaggle. (2025). *German Traffic Sign Recognition Benchmark (GTSRB)* [Dataset]. Kaggle. https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign

DeepLearningTutorials. (2024, March 15). *Traffic sign recognition using deep learning | CNN | Python* [Video]. YouTube. https://www.youtube.com/watch?v=eS8sE1M9dks