# Report: Traffic Sign Classification using Convolutional Neural Networks (CNN)

## Objective

The objective of this assignment is to develop a Convolutional Neural Network (CNN) for classifying traffic signs using the **German Traffic Sign Recognition Benchmark (GTSRB)** dataset. The dataset contains images of 43 different traffic sign classes, captured under various lighting, weather, and angle conditions. This diversity allows for the training of a robust model capable of recognizing traffic signs in different real-world scenarios. The goal is to evaluate the performance of a CNN in this multi-class classification problem and compare the accuracy of the model with and without data augmentation.

## Data Preprocessing

- **Importing the GTSRB dataset**: The GTSRB dataset is widely used for traffic sign recognition tasks. It contains over 50,000 images of 43 different traffic sign classes.
- **Resizing images to 32×32×3 (RGB)**: Resizing the images ensures that all input data has a consistent shape, which reduces computational complexity and helps the model focus on relevant features.
- **Normalizing pixel values to the range [0, 1]**: Normalizing the images ensures that pixel values are within a consistent range, improving the stability of the model during training.
- **Splitting the dataset**: The data is split into training (80%), validation (10%), and testing (10%) sets to evaluate the model's performance.
- **Data Augmentation**: To improve the generalization of the model and reduce overfitting, various augmentation techniques are applied during training,

such as:

- Rotation (up to 10 degrees)
- Zooming (up to 15%)
- Shifting (up to 10% in width and height)
- Shear transformations (up to 15%)
- No horizontal or vertical flipping (since traffic signs are directional)
- nearest fill mode to handle the empty pixels created by transformations.

## Model Architecture

The model is a deep Convolutional Neural Network (CNN) designed to classify images of traffic signs into one of 43 classes. The architecture includes several convolutional layers, followed by pooling, normalization, and fully connected layers:

1. **Convolutional Layer 1**: Extracts low-level features like edges and textures.
   - 16 filters with a kernel size of (3×3)
   - ReLU activation function
   - MaxPooling (2×2)
2. **Convolutional Layer 2**: Extracts mid-level features like shapes and finer textures.
   - 32 filters with a kernel size of (3×3)
   - ReLU activation function
   - MaxPooling (2×2)
3. **Convolutional Layer 3**: Extracts high-level features such as objects and specific sign patterns.
   - 64 filters with a kernel size of (3×3)
   - ReLU activation function
   - MaxPooling (2×2)
4. **Convolutional Layer 4**: Further feature extraction.

- 128 filters with a kernel size of (3×3)
- ReLU activation function
- MaxPooling (2×2)

5. **Flatten Layer**: Converts the 2D feature maps into 1D vectors to be passed to fully connected layers.
6. **Dense Layer 1**: A fully connected layer that learns relationships between the extracted features.
   - 512 neurons with ReLU activation function
   - Batch Normalization and Dropout (0.5) for regularization
7. **Output Layer**: A fully connected layer with 43 neurons, corresponding to the number of classes, and uses the **softmax** activation function to output probabilities for each class.

## Training Details

1. **Epochs**: The model is trained for **30 epochs** to allow sufficient learning.
2. **Batch Size**: A batch size of **32** is chosen to balance between speed and model performance.
3. **Optimizer**: The **Adam optimizer** is used with a learning rate of **0.001** to adjust the learning rate dynamically during training.
4. **Loss Function**: **Categorical Cross-Entropy** is used as the loss function, which is suitable for multi-class classification problems.
5. **Evaluation Metrics**: We track **accuracy**, **precision**, **recall**, and **F1-score** to evaluate the model's performance.

## Results

The model was evaluated on the training, validation, and testing datasets. The results indicate strong performance with high accuracy across all datasets:

- **Training Accuracy**: 98.1%
- **Validation Accuracy**: 97.2%
- **Test Accuracy**: 97.97%

Additionally, precision, recall, and F1-score metrics were calculated for each class. Overall, the model achieved:

- **Macro Average Precision**: 96%
- **Macro Average Recall**: 97%
- **Macro Average F1-score**: 96%

The model demonstrates consistent performance with strong accuracy across different datasets, indicating that the model is generalizing well.

## Evaluation Metrics

- **Precision**: Measures the proportion of correctly predicted positive instances out of all predicted positives.
- **Recall**: Measures the proportion of correctly predicted positive instances out of all actual positives.
- **F1-score**: Harmonic mean of precision and recall, providing a balance between them.

## Conclusion

The CNN model, trained with augmented data, successfully classifies traffic signs with high accuracy. The inclusion of data augmentation has enhanced the model's robustness to variations in the input images, such as rotation, zoom, and shifting. The model generalizes well, as seen in the high performance across the training, validation, and test sets. This approach is effective for real-world applications in autonomous driving systems and traffic sign recognition technologies.