ADVANCE DATA MINING PROJECT

TRAFFIC SIGN RECOGNITION

Monika Mundała

Mehmet Akif Arıcan

Krakow - 2023

1. Introduction

Traffic Sign Recognition (TSR) is a critical component of modern transportation systems, especially in the context of emerging technologies such as autonomous vehicles and advanced driver assistance systems (ADAS). The ability to accurately and swiftly identify and interpret traffic signs is paramount for ensuring safe and efficient navigation on roadways.

This project centers on employing deep learning techniques to tackle the TSR problem, aiming to develop a model that robustly classifies traffic signs into distinct categories. The project leverages a dataset curated from real-world traffic scenarios, emphasizing the practicality and real-world applicability of the proposed solution.

Figure - 1 (Examples from the dataset)

2. Dataset

The dataset used for this project is sourced from:
https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign

It consists of images of traffic signs, categorized into 43 classes.

3. Method

The flow chart of the project is as follows.

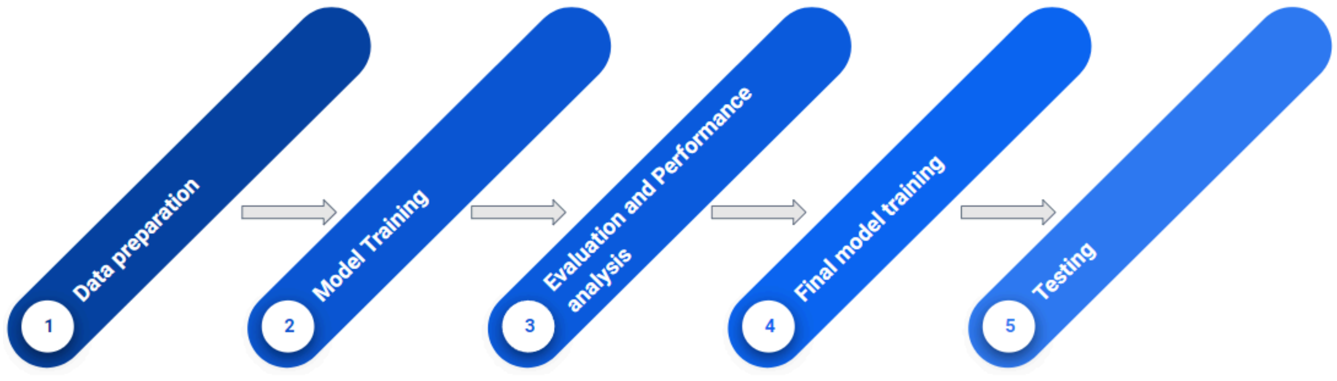
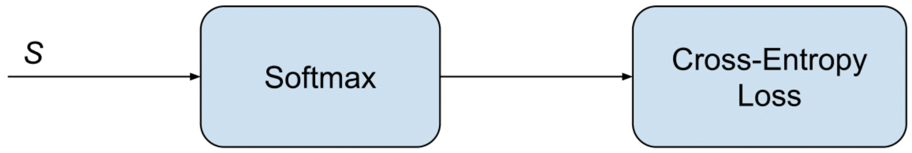


Figure - 2 (Project's flow chart)

The neural network model employed in this project is a convolutional neural network (CNN), a class of deep learning models well-suited for image classification tasks. The architecture comprises multiple convolutional layers followed by max-pooling layers to extract hierarchical features from input images. Batch normalization and dropout layers are incorporated to enhance model generalization and prevent overfitting.

The final layers consist of densely connected layers, culminating in a softmax activation layer for multi-class classification. The model's architecture is carefully crafted to balance complexity and efficiency, ensuring a robust representation of traffic sign features.

The chosen loss function for training the model is categorical cross-entropy. This loss function is appropriate for multi-class classification tasks, penalizing the model based on the divergence between predicted and true class probabilities. It serves as a critical component in guiding the model towards accurate classification.



$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$

Figure - 3 (Categorical Cross-Entropy Loss)

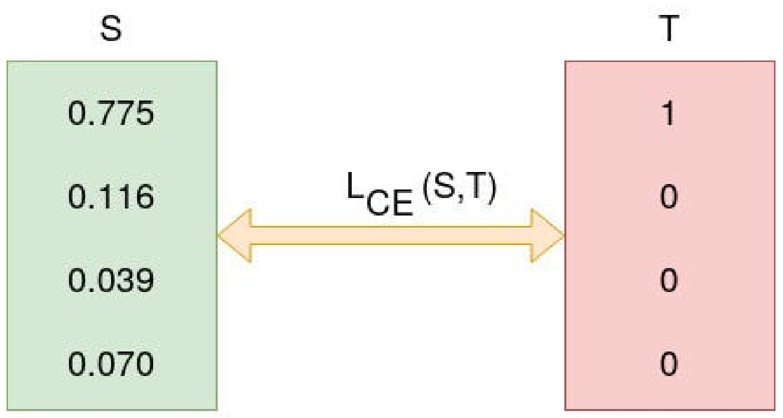


Figure - 4 (Categorical Cross-Entropy Loss)

To optimize model performance, hyperparameter tuning is employed using the Hyperband algorithm through the Keras Tuner framework. The hyperparameters under consideration include learning rate, batch size, and the number of training epochs.

The hyperparameter tuning process enhances the model's adaptability to the intricacies of the traffic sign dataset, contributing to improved generalization and overall accuracy.

4. Implementation of the Method

The model training phase is a pivotal aspect of the project, aiming to optimize the neural network's parameters to achieve superior performance in classifying traffic signs. This section provides a comprehensive overview of the training pipeline, including data preprocessing, augmentation, and the fine-tuning of hyperparameters.

Before delving into the model training itself, it's crucial to highlight the significance of data preprocessing and augmentation. Our dataset has 43 different sign classes with different amounts of images.

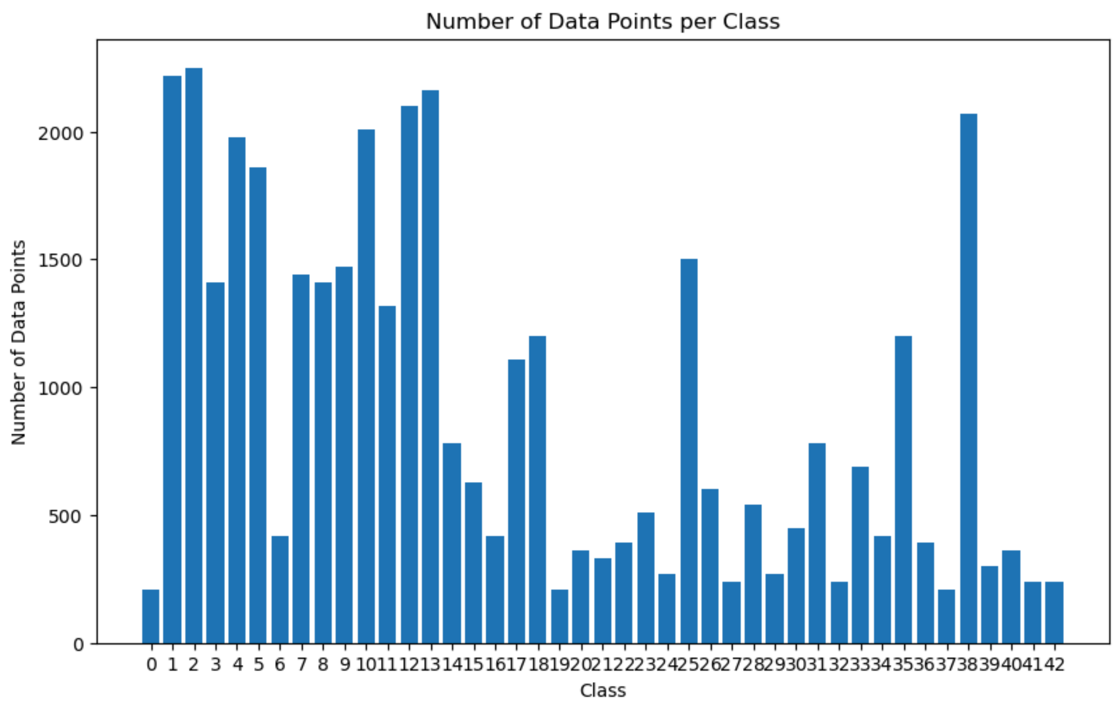


Figure - 5 (Distribution of images according to the classes)

The raw dataset, consisting of traffic sign images, undergoes preprocessing steps, including resizing to a uniform dimension of 30x30 pixels. Additionally, data augmentation is applied to the training set using the Keras `ImageDataGenerator`. This augmentation introduces variations in the training images, such as rotation, width and height shifting, rescaling and zooming. These techniques artificially expand the dataset, enhancing the model's ability to generalize to diverse scenarios and improve its overall robustness.

The model training process involves feeding the augmented training dataset into the neural network.

The model is constructed based on the best hyperparameters determined through the hyperparameter tuning process, which involves a grid search using the Keras Tuner framework. The best hyperparameters, such as the learning rate, batch size, and number of epochs, are crucial in fine-tuning the model's performance.

```
param_grid = {
    'learning_rate': [0.01, 0.001, 0.0001],
    'dropout_rate': [0.1, 0.2, 0.3],
    'num_filters': [(64, 128), (128, 256), (256, 512)],
    'dense_units': [256, 512, 1024],
    'batch_size': [128, 256, 512, 1024],
    'epochs': [15, 20, 30],
}
grid_search = GridSearchCV(estimator=keras_model, param_grid=param_grid,
cv=3)
grid_result = grid_search.fit(X_train, y_train)
```

The model is a Sequential neural network, indicating a linear stack of layers.
It starts with a 2D convolutional layer with a specified number of filters and a 3x3 kernel. The activation function used in the convolutional layers is Rectified Linear Unit (ReLU), known for introducing non-linearity. The input shape of the first layer is set to (30, 30, 3), representing a 3-channel image with dimensions 30x30. Two convolutional layers with the same number of filters are followed by a MaxPooling layer, which reduces spatial dimensions by selecting the maximum value in each region. Dropout layers are added after each MaxPooling layer to prevent overfitting by randomly deactivating a fraction of neurons during training. The model then repeats a similar pattern of two convolutional layers, MaxPooling, and Dropout. After the convolutional layers, a Flatten layer is used to convert the 2D output to a 1D vector for feeding into a dense layer. A dense layer with a specified number of units and ReLU activation follows, introducing further non-linearity. The final layer is a dense layer with 43 units (representing the number of output classes) and a softmax activation function, suitable for multiclass classification. The model is compiled using the Adam optimizer and categorical cross-entropy loss, with accuracy as the evaluation metric.

```
 model = Sequential()

    model.add(Conv2D(num_filters[0], (3, 3), activation='relu', input_shape=(30, 
30, 3)))
    model.add(Conv2D(num_filters[0], (3, 3), activation='relu'))
    model.add(MaxPool2D(pool_size=(2, 2)))
    model.add(Dropout(dropout_rate))

    model.add(Conv2D(num_filters[1], (3, 3), activation='relu'))
    model.add(Conv2D(num_filters[1], (3, 3), activation='relu'))
    model.add(MaxPool2D(pool_size=(2, 2)))
    model.add(Dropout(dropout_rate))

    model.add(Flatten())
    model.add(Dense(dense_units, activation='relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(43, activation='softmax'))

    optimizer = Adam(learning_rate=learning_rate)
    model.compile(loss='categorical_crossentropy', optimizer=optimizer, 
metrics=['accuracy'])
```

The training process is visualized using performance graphs, showcasing the evolution of accuracy and loss over epochs for both the training and validation sets.These graphs provide insights into the model's convergence and its ability to generalize to unseen data.

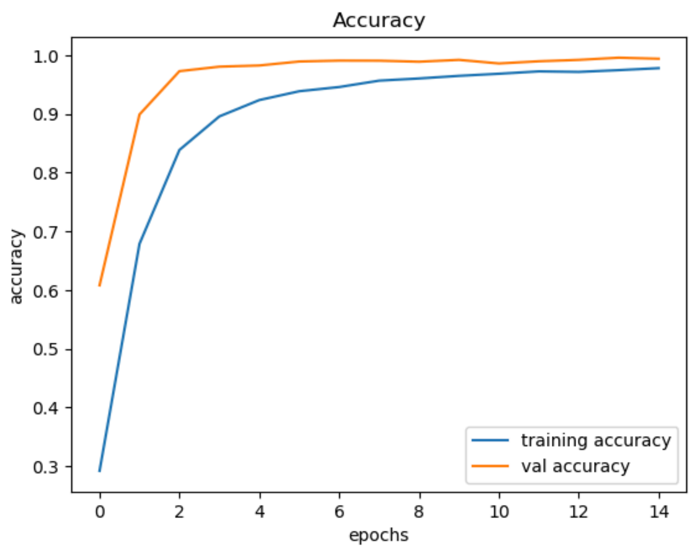


Figure - 6 (Accuracy plot)

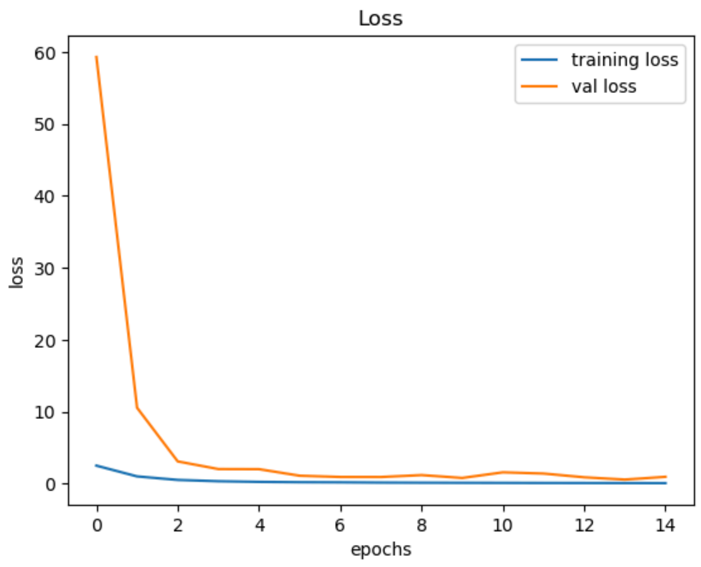


Figure - 7 (Loss plot)

The trained model is evaluated on the test dataset, which comprises real-world traffic sign images not seen during training. This evaluation involves calculating various metrics to assess the model's overall performance. These metrics provide a comprehensive assessment of the model's ability to generalize to new, unseen data.

`Test Accuracy: 0.9600950118764846` #The overall accuracy of the model on the test dataset

`Test F1 Score: 0.9603818645521773` #The harmonic mean of precision and recall, providing a balanced measure of the model's accuracy.

`Test Precision: 0.9635205457731415` #The ratio of correctly predicted positive observations to the total predicted positives.

`Test Recall: 0.9600950118764846` #The ratio of correctly predicted positive observations to all actual positives.

The confusion matrix is a valuable visualization tool that provides a detailed breakdown of the model's predictions. It displays the number of true positive, true negative, false positive, and false negative predictions for each class.

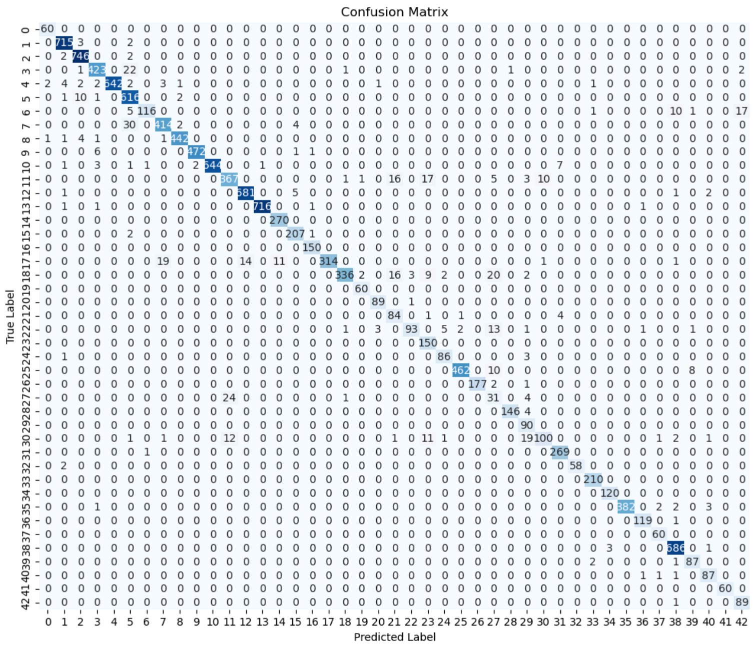


Figure - 7 (Confusion matrix)

The confusion matrix is particularly useful for identifying classes where the model may struggle, allowing for targeted improvements in future iterations.

To further gauge the model's generalization, it undergoes evaluation on a separate test dataset, ensuring its readiness for real-world deployment. This evaluation on an additional test dataset validates the model's generalization and real-world performance.

5. Results

By using hyperparameter tuning we detected the best parameters for our model and these parameters are:

- Learning Rate: 0.001
- Batch Size: 512
- Epochs: 30
- Num Filters: 32, 64
- Dense Units: 512
- Dropout Rate: 0.25

By using these parameters we runned our model and get the accuracy and loss curves for our model’s training and validation.

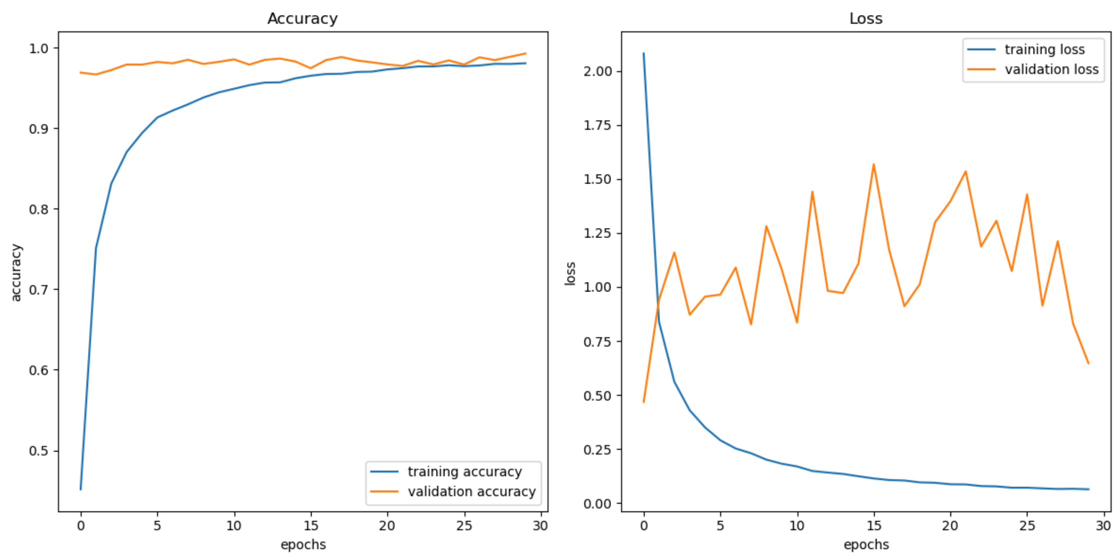


Figure - 8 (Accuracy and Loss plots)

We can observe an absolute increase in the training accuracy till the 20th epoch, then it starts to stay the same. After the 30th epoch model shows overfitting and accuracy starts to decrease.

The model's performance is quantified using a range of accuracy metrics, providing a nuanced understanding of its effectiveness in classifying traffic signs.

The overall accuracy of the model on the test dataset is a key metric, indicating the percentage of correctly classified instances. In our evaluation, the model demonstrates an impressive Test Accuracy of 96.25%, attesting to its ability to discern and categorize diverse traffic signs.

The F1 Score, the harmonic mean of precision and recall, offers a balanced assessment of the model's predictive capabilities. Our model achieves a noteworthy F1 Score of 96.27%, reinforcing its precision and recall across multiple classes.

Precision and recall metrics provide insights into the model's ability to minimize false positives and false negatives, respectively. The model exhibits a Precision of 96.72% and a Recall of 96.25%, reflecting its high precision in predicting positive instances and its comprehensive coverage of actual positive instances.

The confusion matrix is a valuable tool for understanding the model's performance at a granular level. It presents a detailed breakdown of true positive, true negative, false positive, and false negative predictions for each class.

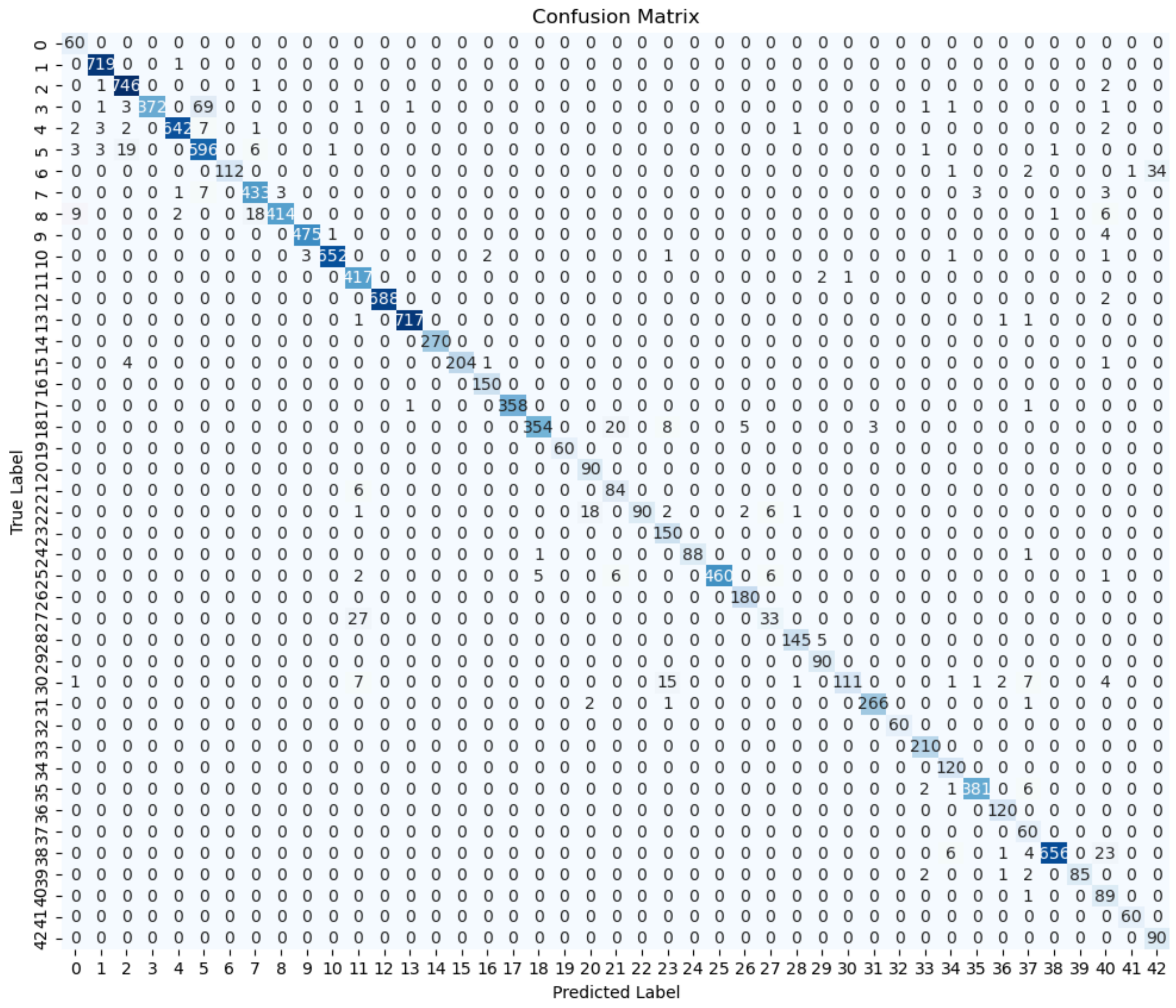


Figure - 8 (Confusion matrix)

The confusion matrix allows us to identify specific classes where the model excels and areas that might benefit from further optimization. For instance, instances of misclassification or lower precision and recall can be pinpointed, guiding future model refinements.

The model demonstrates notable strengths in accurately classifying a diverse array of traffic signs. Classes with high precision and recall contribute to the model's overall impressive performance. Notably, Class - 12 with almost no misclassification shows 99.7 percent accuracy.

Figure - 9 (Class - 12)

Although the model works well in many areas, there are opportunities for improvement. The confusion matrix highlights specific classes where misclassifications occur more frequently. Investigating and addressing these examples may lead to further improvements in overall accuracy.

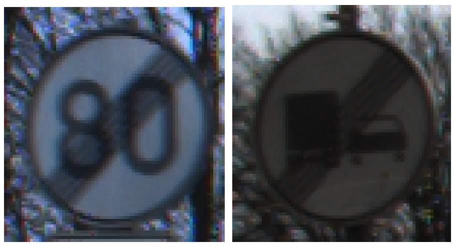


Figure - 10 (Most confused classes (class 6 - 42))

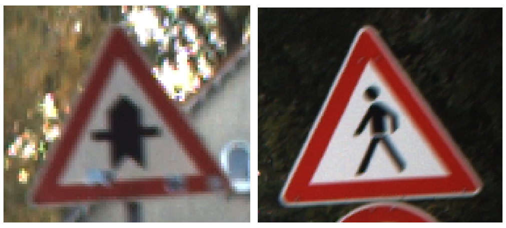


Figure - 11 (Most confused classes (class 27 - 11))

However, the similarity between confused classes shows that the problem can be fixed by little adjustments.

In the end, we tested our model on a separate test dataset to be sure of its performance. We got a close result for accuracy metric to training and validation accuracies.

```
395/395 [==============================] - 12s 31ms/step
Accuracy on test dataset: 0.981631037212985
```

6. Discussions

As part of future work, exploring additional data augmentation techniques, fine-tuning the model architecture, or incorporating advanced optimization strategies could be considered. Additionally, an in-depth analysis of misclassified instances can provide valuable insights for targeted improvements.

The robust performance of the model on the test dataset instills confidence in its real-world applicability. Further validation on diverse datasets and in varied environmental conditions would bolster its suitability for deployment in practical traffic scenarios.

The most challenging part of the project is that it is so time-consuming. Especially, running a model by using different parameters on hyper-parameter tuning took us a couple of days.

We have builded more complicated and simpler model architectures by adding and removing layers and we can infer that neither more complicated nor simpler models are useful for us. We need to detect the most suitable one to get the best results.