Documentation: Traffic Object Classification

1. Introduction

This project aims to develop an effective pipeline for object classification in traffic scenes, using deep learning techniques. The work covers cleaning, augmentation, dataset preparation, model training (from scratch and pretrained), and evaluation on real-world images.

2. Dataset & Problem Definition

Dataset: The dataset used is *mikoajkoek/traffic-road-object-detection-polish-12k* from Kaggle, containing annotated traffic images of Polish roads. The data was collected using a car camera on roads mainly in Krakow. The images capture a diverse range of scenarios including different road types and various lighting conditions. The dataset had 11 classes, of which 3 classes were selected that were most represented. The rest of the classes had fewer labeled items and were not considered. The goal was to accurately classify three key traffic objects: **Cars**, **Different Traffic Signs** and **Pedestrians.**

3. Preprocessing & Data Preparation

3.1 Data Cleaning & Deduplication

The dataset was annotated, so the duplicate images were removed by retaining only the first image of each set (original) with the same prefix, ensuring unique samples for model evaluation. At this step corresponding label files were also filtered and matched with the original photo.

3.2 Dataset Splitting

The dataset was split into training (60%), validation (20%) and test (20%) sets, ensuring reproducibility by saving split indices. The splits were made before any preprocessing step to ensure that no cropped images from the same image will end up in train and test set.

The indices were used to create "Golden" versions of the splits to keep track of data integrity. The golden test set was used for the evaluation of both the models from scratch and the pretrained model.

3.3 Copying for Preprocessing

For augmentation and cropping, all necessary directories were created and populated with images and labels, ready for further processing.

4. Preprocessing Techniques

4.1 Cropping and Augmentation

For the specific classes, YOLO-format bounding boxes were used to crop object regions from images. To insert noise into the training dataset, three jittered crops were generated with bounding box jittering for each object. The box center moves randomly but by no more than ±15% of the original box's width and height.

The images in validation and test set were only cropped without augmentation, ensuring clean and unbiased evaluation.

For each cropped image, the corresponding label file was updated and saved to dedicated output directories for later use in training and evaluation.

4.2 Image- Label Pairing

For the pairing of images and labels a custom CroppedImageDataset class was implemented, which pairs each cropped image with its corresponding label and makes it easy to load them as image - class samples for training or evaluating models in Pytorch. This class ensures that the data is aligned and ready for use in data loaders.

All images were resized to a consistent size of 128×128 pixels for training compatibility.

5. Model Architecture & Training

5.1 SimpleCNN Classifier

A simple convolutional network (CNN) was implemented at first. The model consists of three convolutional layers with increasing channel sizes, each followed by batch normalization, ReLU activation, and max pooling to progressively extract and condense image features. The extracted features are then flattened and passed through two fully connected layers, with dropout applied to reduce overfitting, before outputting class predictions. The training pipeline includes separate functions for training and validation, ensuring proper handling of data, loss computation, and accuracy tracking.

5.2 Deep CNN Classifier

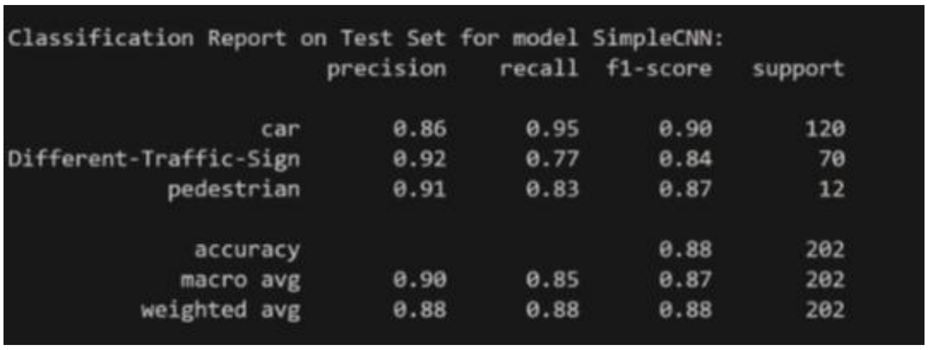
A deep convolutional neural network (CNN) architecture was developed to enhance image classification performance. The model is structured into three main convolutional blocks, each containing two convolutional layers followed by batch normalization and ReLU activations to enable deeper feature extraction. Max pooling is applied after the first two blocks to reduce spatial dimensions while the final block uses adaptive average pooling to generate compact, fixed-size feature maps regardless of input size. The resulting features are then flattened and passed through a fully connected classifier with a hidden layer, ReLU activation, and dropout for regularization before the final output layer predicts class probabilities.

Both models are trained using the Adam optimizer and cross-entropy loss and the network with the highest validation accuracy is automatically saved for later evaluation. The weights of the models are saved to drive for later use.

6. Evaluation & Results

6.1 Classification metrics

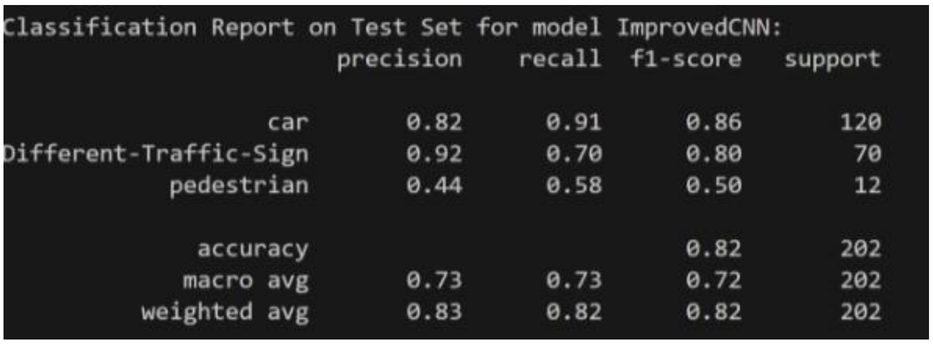
Classification reports for the two models:



```
Classification Report on Test Set for model SimpleCNN:
                       precision    recall  f1-score   support

                 car       0.86      0.95      0.90       120
Different-Traffic-Sign       0.92      0.77      0.84        70
          pedestrian       0.91      0.83      0.87        12

            accuracy                           0.88       202
           macro avg       0.90      0.85      0.87       202
        weighted avg       0.88      0.88      0.88       202
```



```
Classification Report on Test Set for model ImprovedCNN:
                       precision    recall  f1-score   support

                 car       0.82      0.91      0.86       120
Different-Traffic-Sign       0.92      0.70      0.80        70
          pedestrian       0.44      0.58      0.50        12

            accuracy                           0.82       202
           macro avg       0.73      0.73      0.72       202
        weighted avg       0.83      0.82      0.82       202
```

The confusion matrices for each model for comparison.

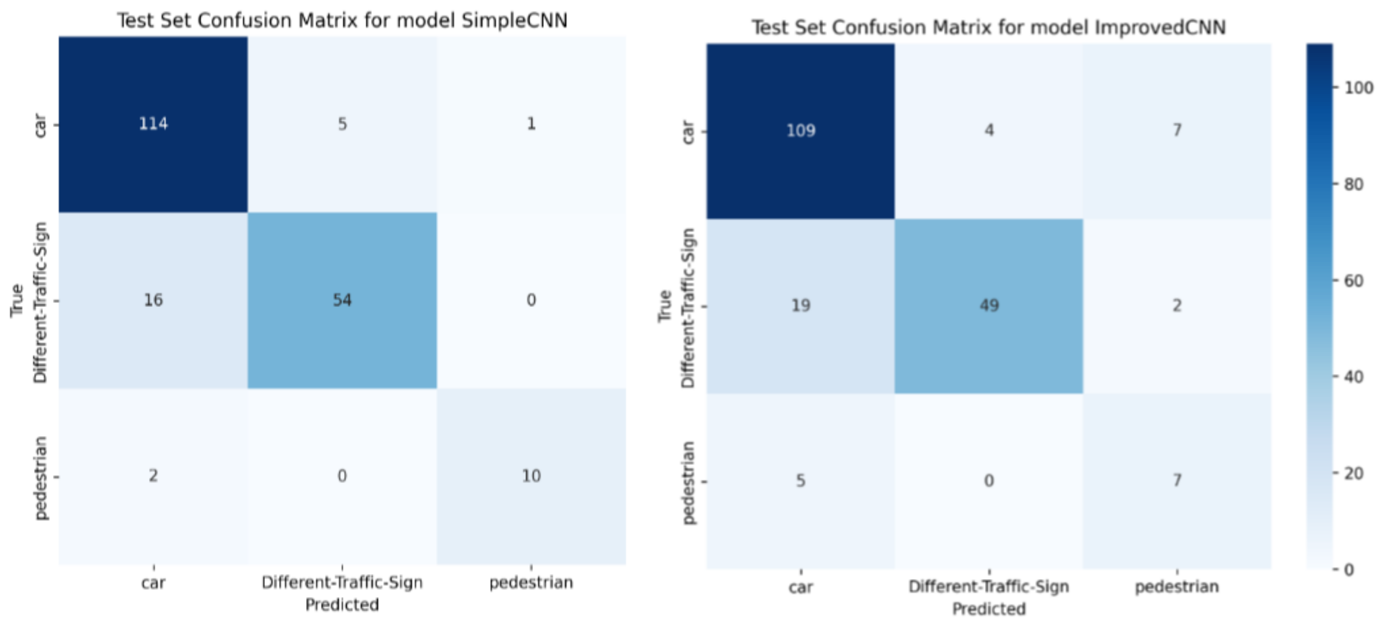


Figure 3: The confusion matrix for the SimpleCNN (on the left) and for the DeepCNN (on the right)

The comparison between SimpleCNN and DeepCNN reveals that SimpleCNN consistently outperforms ImprovedCNN across all evaluation metrics. SimpleCNN achieves an overall accuracy of 0.88, compared to 0.82 for ImprovedCNN. For the car" and "Different-Traffic-Sign" classes, SimpleCNN records higher recall and f1-scores, despite both models having the same precision for the latter. Also, it has an advantage in classifying the “pedestrian” class. The average macro and weighted f1-scores also favor SimpleCNN (0.87 and 0.88) over ImprovedCNN (0.72 and 0.82). These results indicate that the modifications in the ImprovedCNN model did not lead to better performance and, in fact, significantly reduced its effectiveness, particularly for minority classes such as "pedestrian".

6.2 Generalization to New Datasets

A new dataset was used that contained among others 2 classes, car and pedestrian, for testing the trained model. The images were cropped with the bounding boxes for the specified classes and were saved with the corresponding labels. The cropped images were filtered and were kept only the ones that were above a minimum dimension (min_size_pixels = 50) for a better resolution. The images were used as a new test set for the already trained SimpleCNN model, which achieved an accuracy of 69.2%.

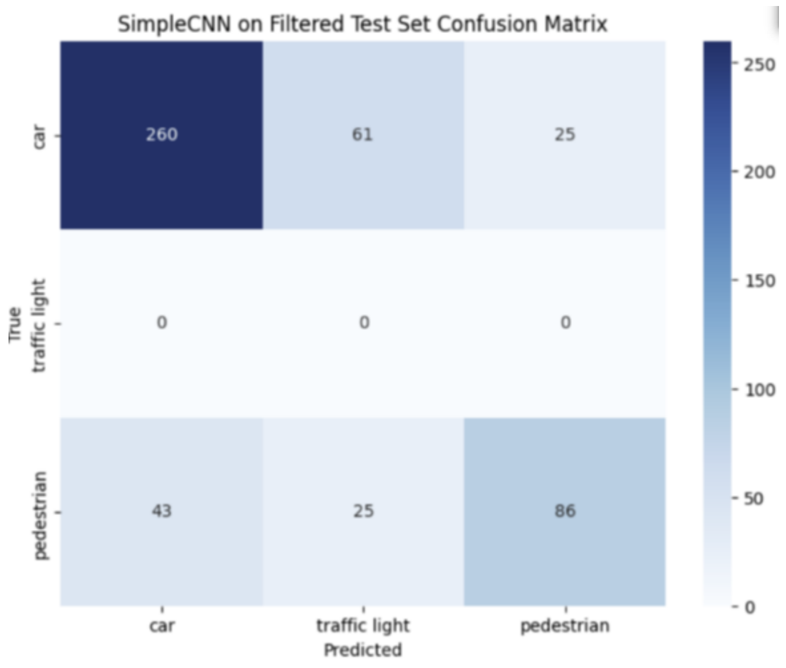


Figure 4

7. Pretrained VGG16 Classifier (Transfer Learning)

The VGG-16 model is a CNN architecture that consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. These layers are organized into blocks, with each block containing multiple convolutional layers followed by a max-pooling layer for down sampling. The pretrained version of this network is trained on over one million images from the ImageNet visual database. It can achieve strong performance on various computer vision tasks, including image classification and object recognition.

7.1 Transformation

The training images were resized to 256×256, and then cropped randomly to 224×224 pixels, because this is the default input size for this model. The images were horizontally flipped, random rotated, colored jittered, transformed to grayscale to augment the dataset and improve model generalization. All convolutional feature extraction layers were frozen so only the classifier was updated during training. The final classification layer of VGG16 was replaced with a new fully connected layer that matches the number of the 3 target classes.

7.2 Training

In the training process, Adam optimizer and cross-entropy loss were used. During training all layers participated in the forward pass to generate predictions and compute the loss. But only the classifier layers had their parameters updated during backpropagation, while the pretrained feature extraction layers remained unchanged. At the end of the training, the best model was reloaded and evaluated on the test set.

7.3 Evaluation & Results

Test performance was reported using accuracy and confusion matrix visualization. The VGG16 achieved 91% accuracy on the same test set, the two previous models were evaluated.

The confusion matrix for the pretrained model.

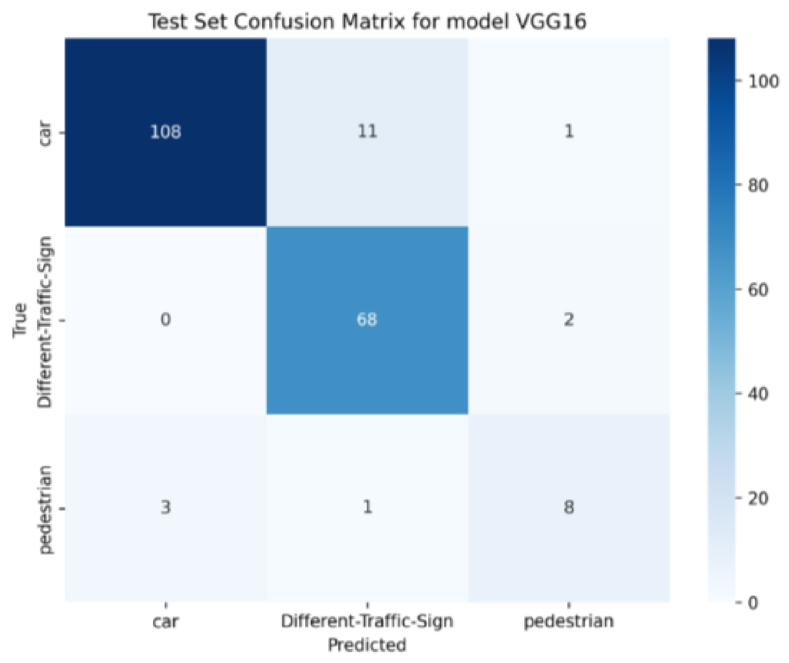


Figure 5

When comparing the performance of SimpleCNN, DeepCNN and the pretrained VGG16 model, it is evident that VGG16 achieves the highest overall accuracy (0.91). The VGG16 finds less cars and pedestrians compared to the simpleCNN but classifies correctly almost all the traffic signs, something the two previous models did not achieve.

8. Conclusion

The project demonstrates a full deep learning pipeline for traffic object classification, combining data preparation and augmentation, diverse modeling strategies, and thorough evaluation. The pipeline is robust, generalizes on new datasets, and is modular for future improvements (e.g., more classes, different datasets).