# Częstochowa University of Technology
# Computer Science

## Computer Vision, Pattern Recognition
## &
## Image Retrieval

**Project I**

**Furkan Şimşekli**

**15.01.2024**

# Objective

The project aims to explore the functioning of neural networks, delve into image data processing, and gain practical experience in convolutional neural networks (CNNs). The specific focus is on the implementation of a region-based convolutional neural network (RCNN) for the purpose of stop sign detection. The overarching objective is to foster a deep understanding of network architecture design, training methodologies, and the complete workflow essential for object detection through the application of advanced deep learning techniques.

# Implementation

## Dataset

The *CIFAR-10* dataset, a widely recognized benchmark in computer vision, comprises 60,000 32x32 color images distributed across 10 classes, with each class containing 6,000 images. Among these, 50,000 images are designated for training, and the remaining 10,000 for testing. The popularity of *CIFAR-10* arises from its well-structured image qualities and distributions, making it suitable for training neural networks. Furthermore, the absence of GPU availability on the utilized machine becomes a non-issue due to the manageable scale of *CIFAR-10*.

While *CIFAR-10* inherently lacks a validation set, a fair division is ensured by splitting the test batch in half, designating one portion as the validation set.

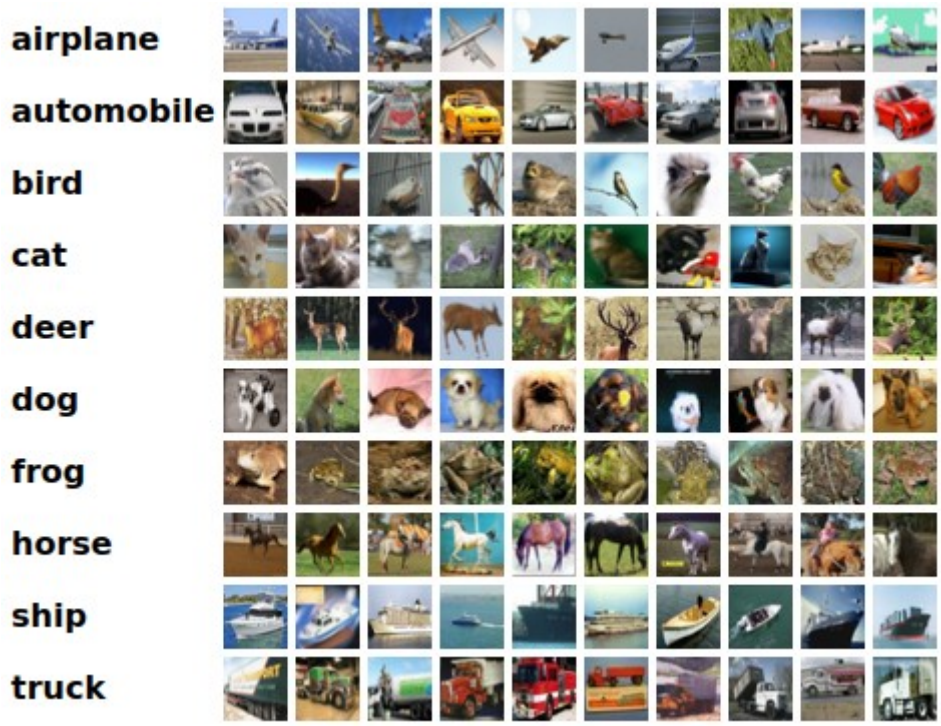



*Figure 1: CIFAR-10 sample images from 10 different categories*

For fine-tuning the network specifically for stop sign detection, the Matlab CV Toolbox's *stopSignImages* dataset has been employed. This dataset comprises a total of 41 RGB images. Although the size of the dataset may be considered limited, its adequacy is justified by the fine-tuning nature of the project. It's noteworthy that the images in this dataset may have different dimensions.

To further validate the trained model, a publicly available dataset featuring not only stop signs but also other street signs, traffic lights, and crosswalks was selected. For this project's purposes, manual extraction was performed to isolate only the stop signs along with their corresponding bounding box values. This validation dataset consists of 20 images, carefully curated to cover various scenarios and conditions. Additionally, 20 test images were selected from this dataset for evaluating the RCNN's performance on previously unseen examples.


*Figure 2: stopSignImages dataset example images with ground truth boxes annotated*

# Network

The chosen network architecture for this project is based on a Convolutional Neural Network (CNN), specifically designed for image classification tasks. In the context of stop sign detection, this CNN serves as a fundamental building block for the Region-Based Convolutional Neural Network (RCNN) employed later in the project.

The primary purpose of the CNN is to learn hierarchical features from input images, enabling it to recognize patterns and characteristics associated with stop signs. This pre-trained CNN, initially developed for image classification on the CIFAR-10 dataset, forms the feature extractor for the subsequent RCNN.

## Architecture

The CNN architecture, referred to as *cifar10Net*, consists of several layers, each contributing to the network's ability to understand and interpret visual information.

**1. Input Layer**

- Dimensions: [*height* x *width* x *numChannels*]
- Represents the input image with specified dimensions.

**2. Convolutional Layers**

- Convolutional layers with varying filter sizes (3x3) and increasing numbers of filters (16, 32, 64, 128, 256) extract features hierarchically.
- Padding is applied to maintain spatial dimensions.

**3. ReLU Activation Layers**

- Rectified Linear Unit (ReLU) activation layers follow each convolutional layer, introducing non-linearity to the network.

**4. Max Pooling Layers**

- Max pooling layers with a 2x2 pool size and specified stride reduce spatial dimensions, retaining important features.

**5. Fully Connected Layer**

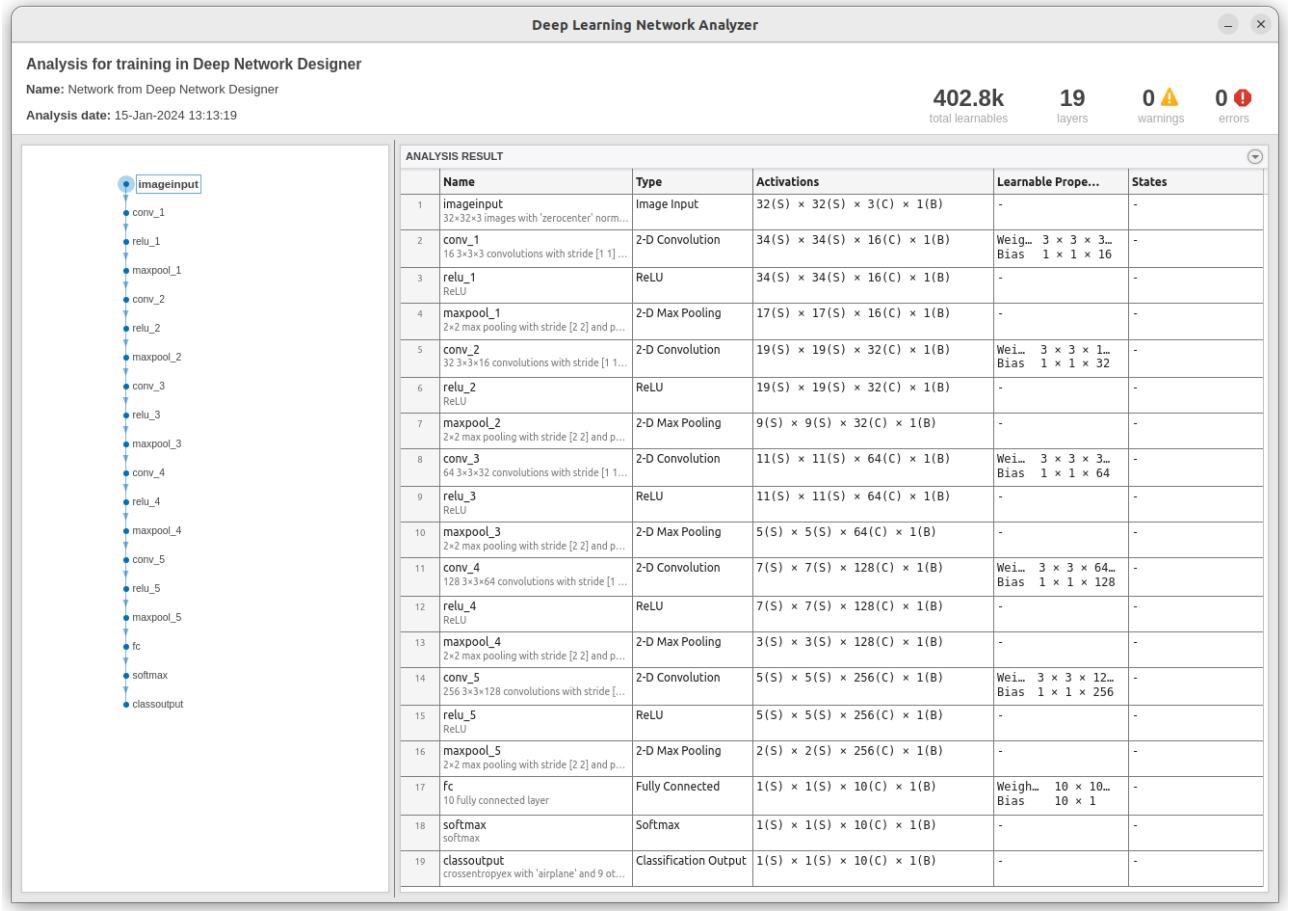
- A fully connected layer with 10 neurons represents the output classes for image classification on *CIFAR-10*.

## 6. Softmax Layer

- The softmax layer converts the network's output into probability distributions over the 10 classes.

## 7. Classification Layer

- The classification layer categorizes the input image into one of the predefined classes.




| | Name | Type | Activations | Learnable Prope... | States |
|---|---|---|---|---|---|
| 1 | imageinput<br>32×32×3 images with 'zerocenter' norm... | Image Input | 32(S) × 32(S) × 3(C) × 1(B) | - | - |
| 2 | conv_1<br>16 3×3×3 convolutions with stride [1 1] ... | 2-D Convolution | 34(S) × 34(S) × 16(C) × 1(B) | Weig… 3 × 3 × 3…<br>Bias 1 × 1 × 16 | - |
| 3 | relu_1<br>ReLU | ReLU | 34(S) × 34(S) × 16(C) × 1(B) | - | - |
| 4 | maxpool_1<br>2×2 max pooling with stride [2 2] and p... | 2-D Max Pooling | 17(S) × 17(S) × 16(C) × 1(B) | - | - |
| 5 | conv_2<br>32 3×3×16 convolutions with stride [1 1... | 2-D Convolution | 19(S) × 19(S) × 32(C) × 1(B) | Wei… 3 × 3 × 1…<br>Bias 1 × 1 × 32 | - |
| 6 | relu_2<br>ReLU | ReLU | 19(S) × 19(S) × 32(C) × 1(B) | - | - |
| 7 | maxpool_2<br>2×2 max pooling with stride [2 2] and p... | 2-D Max Pooling | 9(S) × 9(S) × 32(C) × 1(B) | - | - |
| 8 | conv_3<br>64 3×3×32 convolutions with stride [1 1... | 2-D Convolution | 11(S) × 11(S) × 64(C) × 1(B) | Wei… 3 × 3 × 3…<br>Bias 1 × 1 × 64 | - |
| 9 | relu_3<br>ReLU | ReLU | 11(S) × 11(S) × 64(C) × 1(B) | - | - |
| 10 | maxpool_3<br>2×2 max pooling with stride [2 2] and p... | 2-D Max Pooling | 5(S) × 5(S) × 64(C) × 1(B) | - | - |
| 11 | conv_4<br>128 3×3×64 convolutions with stride [1 ... | 2-D Convolution | 7(S) × 7(S) × 128(C) × 1(B) | Wei… 3 × 3 × 64…<br>Bias 1 × 1 × 128 | - |
| 12 | relu_4<br>ReLU | ReLU | 7(S) × 7(S) × 128(C) × 1(B) | - | - |
| 13 | maxpool_4<br>2×2 max pooling with stride [2 2] and p... | 2-D Max Pooling | 3(S) × 3(S) × 128(C) × 1(B) | - | - |
| 14 | conv_5<br>256 3×3×128 convolutions with stride [... | 2-D Convolution | 5(S) × 5(S) × 256(C) × 1(B) | Wei… 3 × 3 × 12…<br>Bias 1 × 1 × 256 | - |
| 15 | relu_5<br>ReLU | ReLU | 5(S) × 5(S) × 256(C) × 1(B) | - | - |
| 16 | maxpool_5<br>2×2 max pooling with stride [2 2] and p... | 2-D Max Pooling | 2(S) × 2(S) × 256(C) × 1(B) | - | - |
| 17 | fc<br>10 fully connected layer | Fully Connected | 1(S) × 1(S) × 10(C) × 1(B) | Weigh… 10 × 10…<br>Bias 10 × 1 | - |
| 18 | softmax<br>softmax | Softmax | 1(S) × 1(S) × 10(C) × 1(B) | - | - |
| 19 | classoutput<br>crossentropyex with 'airplane' and 9 ot... | Classification Output | 1(S) × 1(S) × 10(C) × 1(B) | - | - |

*Figure 3: cifar10Net architecture*

## RCNN Integration

Following the training of *cifar10Net*, it is integrated into the RCNN for stop sign detection. The RCNN, initiated by the *trainRCNNObjectDetector()* function, fine-tunes the pre-trained CNN on the stop sign dataset (*stopSignImages*). The RCNN's configuration includes options for positive and negative overlap ranges.

This architecture, by leveraging a pre-trained CNN for feature extraction, forms a robust foundation for the subsequent stop sign detection using the RCNN methodology.

## Preprocess

In this project, no explicit preprocessing or data augmentation was applied to either the *CIFAR-10* dataset or the stopSigns dataset. The decision not to perform preprocessing on the *CIFAR-10* dataset is grounded in its inherent characteristics: being a well-structured and compact dataset with uniform images. The absence of preprocessing operations on the stopSigns dataset is primarily due to the pre-existence of labeled data with ground truth coordinates. Any preprocessing could potentially alter these coordinates, leading to a deviation from the true object positions. Additionally, no data augmentation techniques were employed to artificially increase the dataset size. This deliberate choice was made to maintain the integrity of the original datasets and avoid introducing any unintended biases or distortions during training.

## Training, Validation and Testing

For the training phase, I meticulously selected parameter values based on the guidelines provided by MathWorks for RCNN and drew upon my past experiences. These guidelines can be referred to in the Resources section for more detailed insights.

Validation was seamlessly integrated into the training process to monitor the model's performance. Throughout validation, the parameters of the network remained unchanged, ensuring a stable evaluation without the risk of overfitting to the training examples. The training options include a crucial parameter called *ValidationPatience*, set to 10 in this project. This parameter dictates the number of epochs with no improvement in validation performance before terminating the training process. Choosing an appropriate *ValidationPatience* is essential for preventing overfitting and determining the optimal stopping point.

The testing process for *cifar10Net* involved straightforward predictions on a set of images, followed by a comparison with the true labels. While the accuracy of *cifar10Net* was measured at **68%** in the latest test, it's essential to note that the primary objective of this network is feature extraction through convolutional layers. This skill is subsequently transferred to the RCNN for stop sign detection. The accuracy of *cifar10Net* is less critical in the context of this project compared to its feature extraction capabilities.

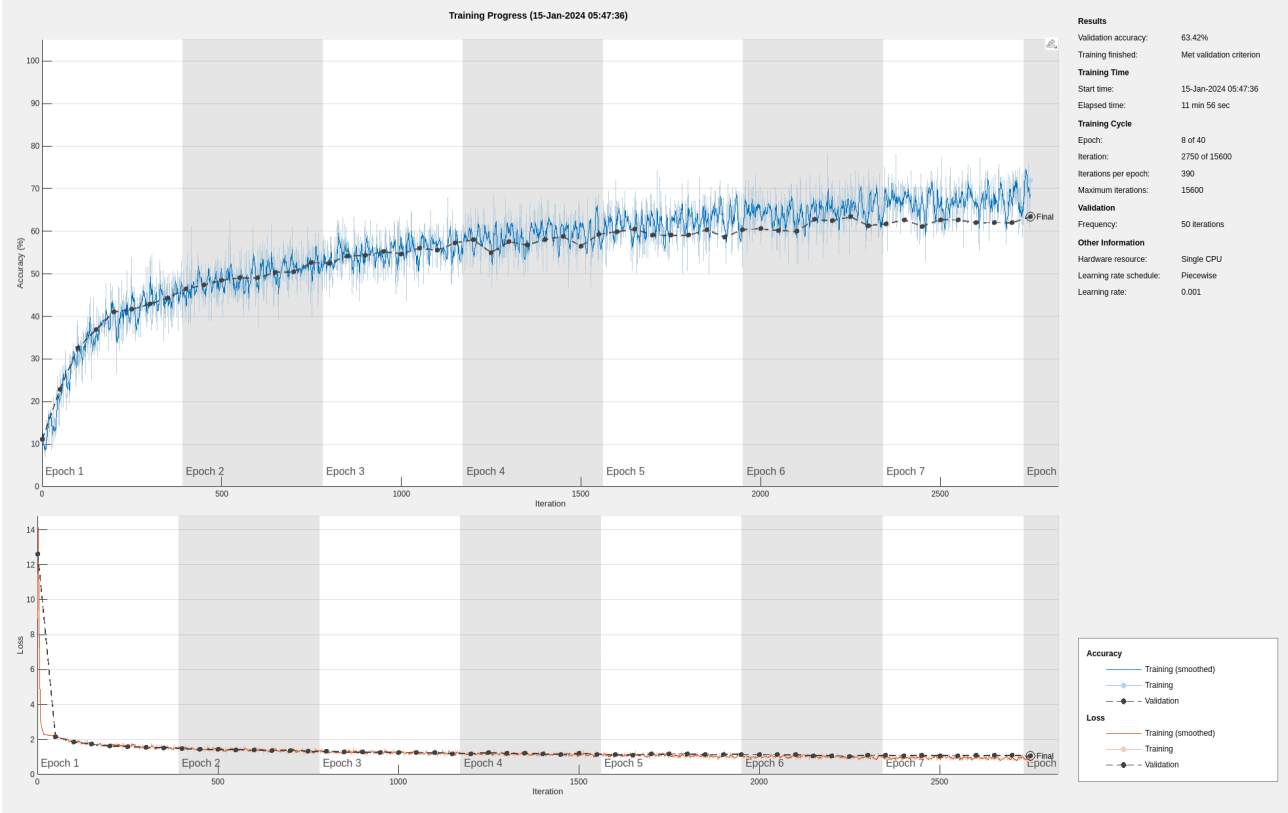



*Figure 4: Training progress of cifar10Net alongside with validation*

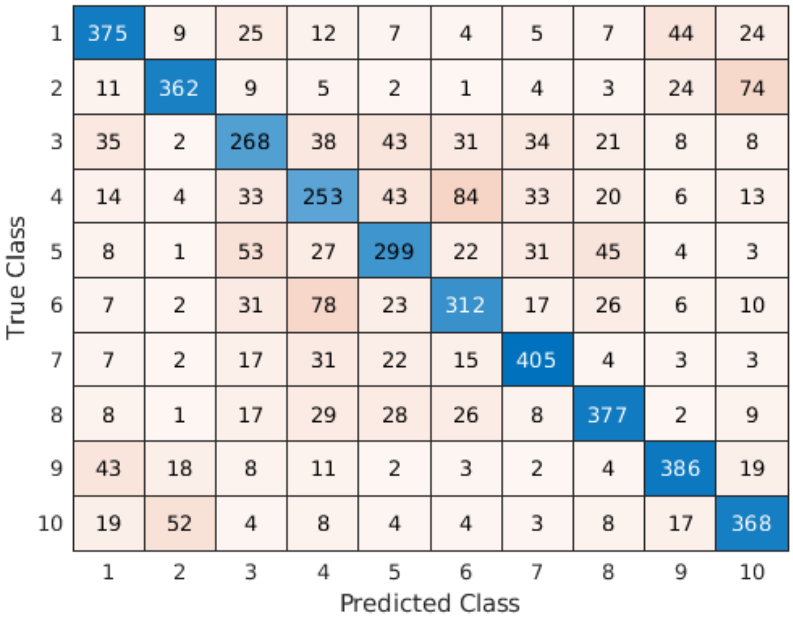



*Figure 5: Confusion matrix after testing over 5,000 images with cifar10Net*

This project's ulterior motive is detecting stop signs, so accuracy in classifying is less significant compared to normal. Our aim in the *cifar10Net* is feature extraction by using filters in convolutional layers, and transfering this skill to RCNN. In my latest test, accuracy in *cifar10Net* was **68%**. This can be improved, as we will see in the RCNN testing, it won't be that significant.


*Figure 6: cifar10Net example predictions*

It's worth highlighting that the use of multiple convolutional layers in *cifar10Net* may not significantly improve the overall accuracy of the network. However, project guidelines necessitate the inclusion of at least 5 different convolutional layers to fulfill the specified requirements.

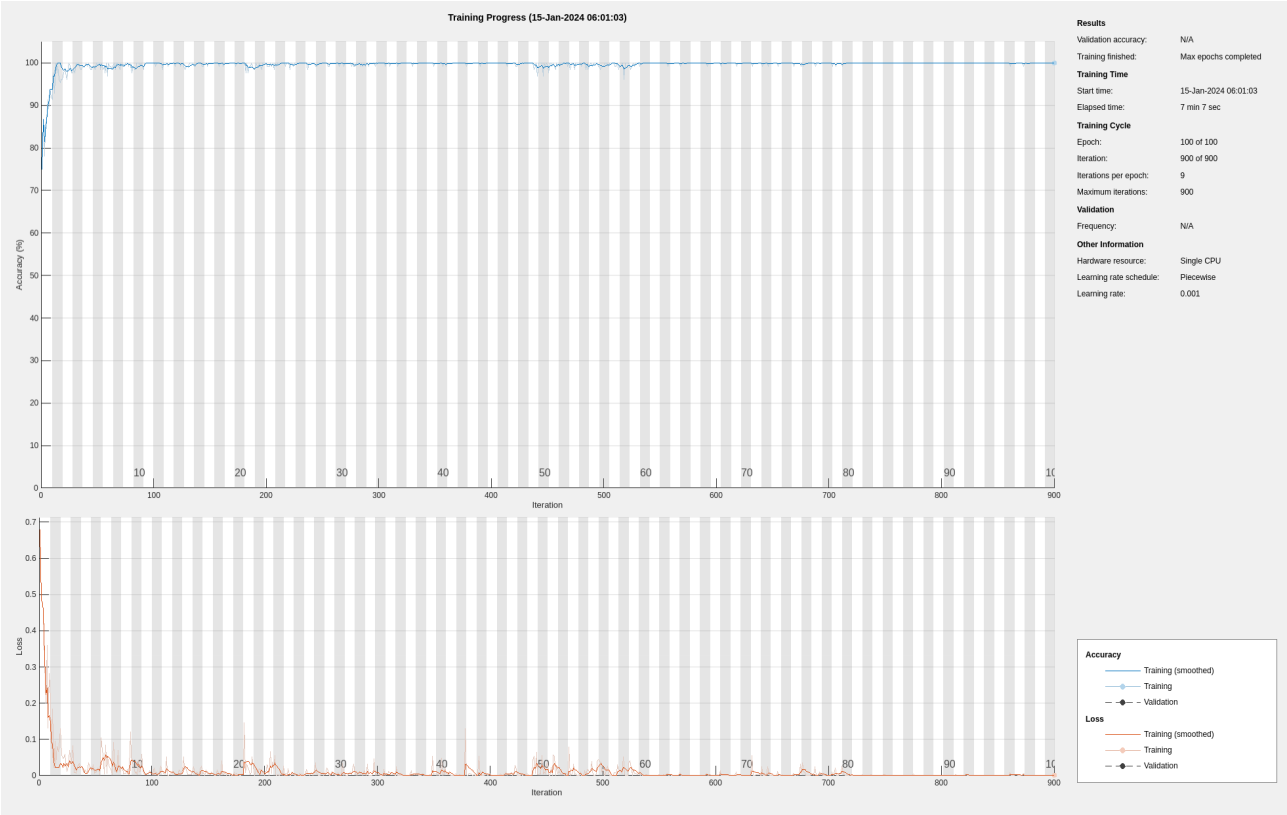



*Figure 7: Training progress of rcnn without validation*

*trainRCNNObjectDetector()* lacks built-in support for validation during training. Given the relatively small size of our dataset, the absence of this feature does not hinder the project's objectives. In scenarios with larger datasets encompassing diverse categories, validation loss and accuracy would play a more pivotal role in guiding the training process. In the tests, rcnn has given near **66%** accuracy for bounding boxes.

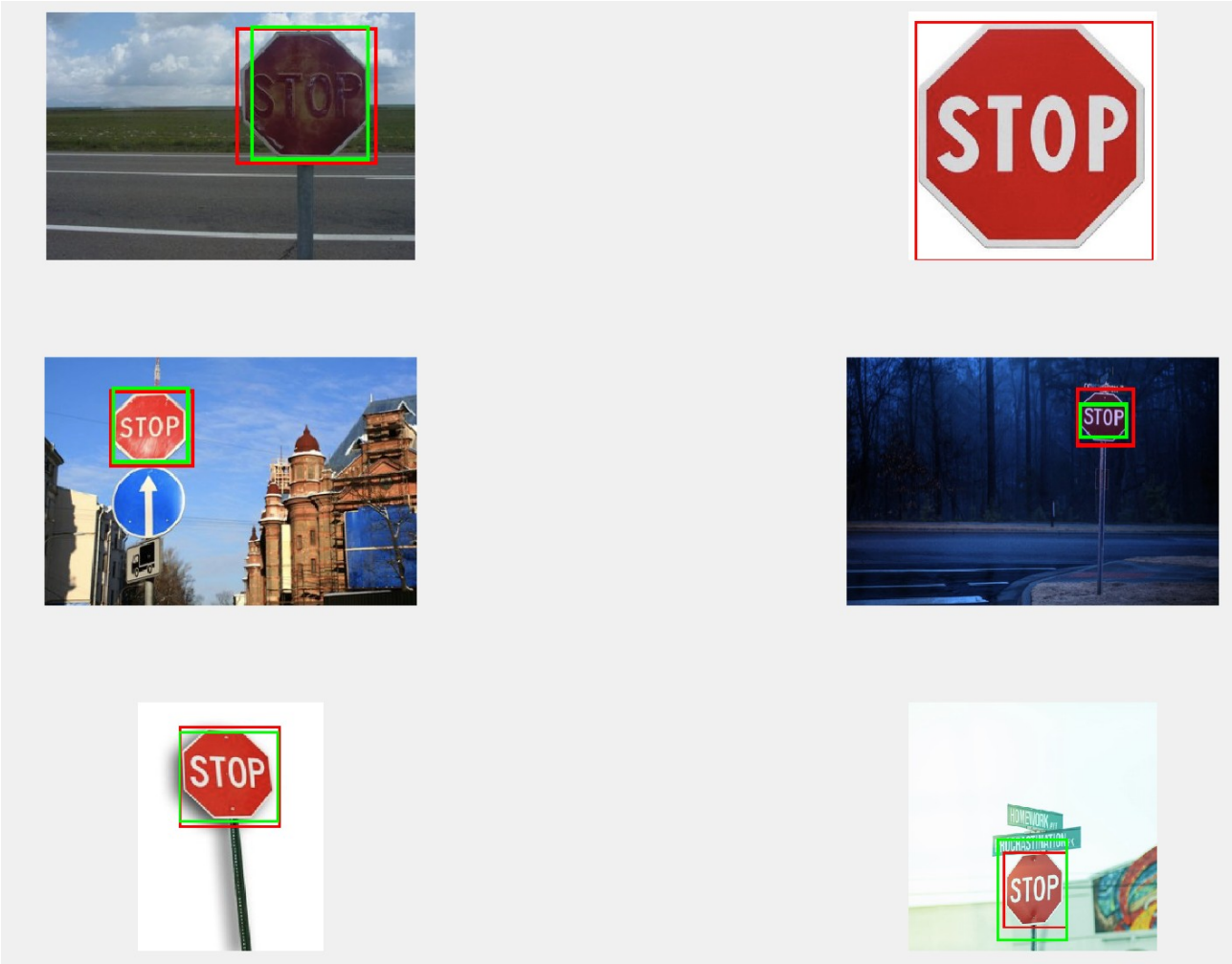



*Figure 8: rcnn predictions, red - ground truth, green predictions*

## Results

In evaluating the performance of *cifar10Net*, a comprehensive set of metrics has been employed beyond the conventional accuracy measure. The assessment includes key metrics such as *recall*, *specificity*, *precision*, *F1-score*, and *Matthew's correlation coefficient*. The utilization of these diverse metrics allows for a more nuanced understanding of the model's capabilities, capturing aspects beyond mere classification accuracy. Each metric serves a unique purpose in revealing specific strengths and weaknesses, contributing to a thorough and insightful analysis of the network's effectiveness in stop sign detection.

| | Accuracy | Recall | Specificity | Precision | F1-score | MCC |
|---|---|---|---|---|---|---|
| Class 1 | 0.9422 | 0.7324 | 0.9661 | 0.7116 | 0.7218 | 0.6897 |
| Class 2 | 0.9552 | 0.7313 | 0.9798 | 0.7991 | 0.7637 | 0.7399 |
| Class 3 | 0.9166 | 0.5492 | 0.9563 | 0.5763 | 0.5624 | 0.5166 |
| Class 4 | 0.9022 | 0.5030 | 0.9469 | 0.5142 | 0.5085 | 0.4543 |
| Class 5 | 0.9264 | 0.6065 | 0.9614 | 0.6321 | 0.6190 | 0.5785 |
| Class 6 | 0.9220 | 0.6094 | 0.9577 | 0.6215 | 0.6154 | 0.5720 |
| Class 7 | 0.9518 | 0.7957 | 0.9695 | 0.7472 | 0.7707 | 0.7443 |
| Class 8 | 0.9468 | 0.7465 | 0.9693 | 0.7320 | 0.7392 | 0.7096 |
| Class 9 | 0.9552 | 0.7782 | 0.9747 | 0.7720 | 0.7751 | 0.7502 |
| Class 10 | 0.9436 | 0.7556 | 0.9639 | 0.6930 | 0.7230 | 0.6925 |

*Figure 9: Evaluating test results with different metrics such as recall, specificity, precision, F1-score, and Matthew's correlation coefficient*

# Conclusion

The completion of this project has been an enjoyable and intellectually stimulating endeavor, albeit not without its challenges. Working within the Matlab environment presented certain difficulties, primarily attributed to the platform's less-than-optimal documentation and limited support for external datasets. Loading external datasets, especially for stop signs, proved challenging due to the absence of parsers for various formats like *Pascal VOC*, *Yolo Darknet*, *Yolo v4*, and *COCO*. Time constraints prevented the development of a custom parser from scratch, leading to a manual and hardcoded solution for the stop signs table in both validation and test datasets.

Matlab's closed-source nature posed an additional hurdle, restricting access to the source code for built-in methods. This limitation hindered the ability to delve into the implementation details, hindering the opportunity for in-depth understanding and optimized problem-solving. While the documentation provides guidance, the absence of direct access to source code impacted the learning curve.

Furthermore, the limited online support and discussions on Matlab exacerbated the problem-solving process. Unlike other widely used programming languages such as Python, Java, and C++, Matlab's online community and resources are notably sparse. Finding solutions to challenges faced during the project proved more cumbersome, emphasizing the need for a more robust and accessible support system.

In conclusion, while the project brought about excitement and fulfillment, the encountered challenges underscore the importance of a more open and collaborative ecosystem for streamlined development and problem-solving. Despite these hurdles, the project has contributed to skill development and provided valuable insights into the nuances of working within the Matlab environment.

# Resources

- CIFAR-10 dataset
https://www.cs.toronto.edu/~kriz/cifar.html
- Road sign detection dataset on Kaggle
https://www.kaggle.com/datasets/andrewmvd/road-sign-detection
- Training an object detector by using RCNN,
https://www.mathworks.com/help/vision/ug/object-detection-using-deep-learning.html
- Transfer learning
https://en.wikipedia.org/wiki/Transfer_learning
- Region-based Convolutional Neural Network
https://en.wikipedia.org/wiki/Region_Based_Convolutional_Neural_Networks