AiLab Report: Traffic Sign Recognition With Pre-Processing and Spatial Transformers

Andrea Pizzi, Simone Turco, Simone Roma
AiLab Course at Sapienza's Computer Science Bachelor Degree

Emails for Contact
pizzi.1995517@studenti.uniroma1.it
turco.1987712@studenti.uniroma1.it
roma.1999214@studenti.uniroma1.it

***Abstract*—Traffic Signal Recognition and Detection (TSRD) is a crucial technology in the field of autonomous driving and advanced driver assistance systems (ADAS). It involves two main tasks: detecting traffic signals in a natural environment (the road) and classifying them. The primary objective of TSRD is to improve road safety by providing real-time information to drivers or autonomous vehicles about the traffic signals encountered along the road. These signals can include speed limits, stop signs, warnings, and other regulatory signals. To achieve these objectives, various artificial intelligence approaches, such as computer vision, have been employed over the years, yielding concrete results in performing the proposed tasks. In this regard, our work has focused on studying the issues related to the application of machine learning to the main tasks of recognition and detection, and on developing artificial intelligence models capable of addressing the identified issues. Finally, we implemented the proposed networks to carry out the complete TSRD activity. The developed networks were trained and tested using the GTSRB (German Traffic Signal Recognition) and GTSDB (German Traffic Signal Detection) datasets.**

I. Introduction

Traffic Sign Recognition and Detection, or TSRD, is an advanced technology that enables vehicles to detect, identify, and interpret traffic signs on the road. This technology is a key component in modern driver-assistance systems and autonomous driving, with the goal of improving road safety, optimizing traffic management, and reducing driver workload. TSRD systems typically use a combination of cameras, sensors, and sophisticated image processing algorithms to recognize various traffic signs and convey this information to the driver or the vehicle's control system.

During TSRD activities, several issues may arise, some of which are common to both main tasks. One issue is the variability in the appearance of traffic signs due to factors such as weather conditions that can affect lighting and scene contrast. Other issues involve the perspective of observation, such as the angle and distance of observation, and the occlusion of signs by objects like trees or other vehicles. Additionally, the diversity of traffic sign designs and standards between different regions and countries complicates the recognition process (especially across continents). Furthermore, the presence of visually similar non-traffic objects, such as advertising signs and billboards, can lead to false positives, causing the system to mistakenly identify irrelevant objects. Generally, these problems are very common in this environment, and although they may seem easy to address, ensuring the robustness and accuracy of TSRD in various driving environments, including urban and highway settings, is a complex but necessary task if we want to move from experimental to practical applications in autonomous driving.

Figure 1: Examples of occluded, blurred, overexposed, and underexposed images.

In our work on the introduced activity, we decided to focus on the classification and detection of traffic signs using ML models. During the realization of this project, we adopted several ML-based approaches for the recognition task, such as using Convolutional Neural Networks [1] (CNNs) for image analysis and Spatial Transformer Networks[2] (STNs) to improve image visibility by eliminating negative variants like distortion, rotation, and limited image size. For the detection task, we analyzed and compared various specialized networks for recognizing objects of different classes in the context of traffic sign recognition.

In Section 2 of this document, the main approaches adopted for detection and recognition are detailed. Section 3 covers the experimentation with various models, their evaluation, and comparison. Section 4 presents the results of the experiments conducted in the previous section. Finally, the concluding Section 5 contains bibliographic references to other papers consulted during the project.

II. Proposed approach

A. Dataset

For training the models, we used two datasets, GTSRB (German Traffic Signal Recognition) and GTSDB (German Traffic Signal Detection), respectively for recognition and detection tasks. GTSRB consists of 43 classes of traffic signals divided into 39,209 images for training and 12,630 images for testing, varying in lighting conditions and backgrounds. GTSDB, on the other hand, comprises 600 images for training and 300 for testing.

B. Model analysis method

For both detection and classification, the networks will extract characteristic information from the images, such as shape, color, and content. To achieve this purpose, all networks employ various convolutional filters capable of identifying and highlighting significant information within an image, which will then be used by the networks' predictive layers to perform detection and recognition tasks.

C. Detection activity

For the detection task, we analyzed two state-of-the-art models specialized in detecting multiple objects, FastR-CNN[3] and SSD300_VGG16[4], and adapted them for the specific task of recognizing various types of traffic signs. The first model analyzed was FastR-CNN, known for its excellent object distinction capabilities despite its lower computational performance due to high internal complexity. This network includes a ResNet50 adapted for detection, along with additional modules such as FPN, RPN, RoI, and two MLP layers. The second model analyzed was SSD300_VGG16, characterized by a good recognition rate and a relatively simple internal structure, making it particularly fast but with lower accuracy compared to the previous network. The computational efficiency of the SSD300_VGG16 network lies in using a VGG16 module for image analysis, supplemented with additional convolutional layers to improve overall accuracy without adding excessive computational overhead.

D. Recognition Activity

For the recognition task, we initially analyzed some of the main ML models based on CNN layers: ResNet18, ResNet32, and tinyVGG, widely used in computer vision. ResNet networks generally boast a good degree of accuracy in various computer vision tasks; however, their internal structure is very complex, especially in the 32-layer versions, making them unsuitable for the specific recognition task due to low accuracy levels. Even the lighter ResNet18 version did not show significant improvements in signal recognition. After exploring various solutions to improve the accuracy of convolutional networks, we developed a more compact and efficient network. The adopted solutions to improve accuracy include preprocessing activities and using STN modules.

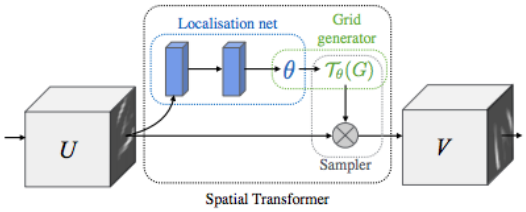


Figure 3: Spatial Transformer Network

The preprocessing module is designed to prepare and transform raw input data into a suitable format for subsequent analysis by the network. Generally, this module performs a series of operations to optimize and standardize the input. Specifically, it handles data normalization concerning chromatic distribution (global normalization) and contrast (local contrast normalization). The 'Spatial Transformer Network' (STN) module can spatially transform input images, producing a homography matrix that allows operations such as scaling, cropping, and rotation, useful for enhancing image quality and visibility. Therefore, our network's structure initially implements a preprocessing module for the images as described, followed by three convolutional blocks each succeeded by an STN module, and finally a linear block for classification.

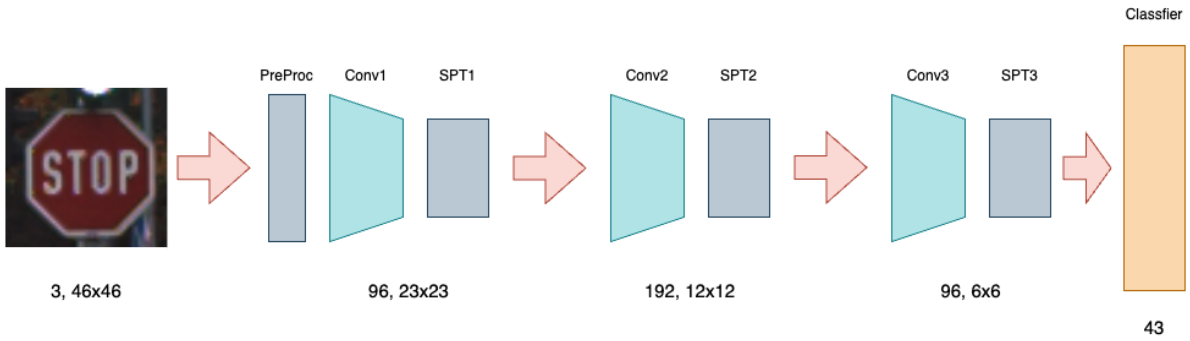


Figure 2: This is a graphical representation of the structure of our network. The image shows the placement of the convolutional blocks, Spatial Transformer Networks (STN), and Pre-Processing.

Table I: Block 1

Layer	Input Shape	Kernel,Stride and Padding
Conv2d	3, 46x46	3,1,1
BatchNorm2d	96	
ReLU	96	
Conv2d	96	3,1,1
BatchNorm2d	96	
ReLU	96	
MaxPool2d	96	2,2

Table II: Block 2

Layer	Input Shape	Kernel and Stride
Conv2d	96, 24x24	3,1,1
BatchNorm2d	192	
ReLU	192	
Conv2d	192	3,1
BatchNorm2d	96	
ReLU	96	
MaxPool2d	96	2

Table III: Block 3

Layer	Input Shape	Kernel and Stride
Conv2d	96, 12x12	3,1,1
BatchNorm2d	96	
ReLU	96	
Conv2d	96	3,1,1
BatchNorm2d	96	
ReLU	96	
Conv2d	96	3,1,1
BatchNorm2d	96	
ReLU	96	
MaxPool2d	96	2,2

Table IV: Classifier

Layer	Input Shape
Flatten	96, 6x6
Linear	3456
Linear	3456

III. Experiments

We will now describe the hardware configuration used to conduct the various experiments with the networks and the different selected hyperparameters. For each network, two experiments were conducted on a single machine with the following configuration: CPU Intel(R) Core(TM) i9-11900 @ 2.50GHz 11th, 16GB DDR4 RAM, GPU accelerator GeForce RTX 3060 with 12GB RAM and 3584 CUDA Cores. The networks were implemented using the PyTorch framework (preview-CUDA version 12.4).

A. Detection

The networks specialized in detection implement specific loss functions capable of evaluating, for each identified object, its belonging to the classes learned during the training phases. The optimization function is the same for both models: AdamW. The reference dataset for the training phase is the GTSDB, divided into two sets: one for training with 592 images and one for testing with 300 images.

Table V: HyperParameters For Detection Models

Model	Epoch	Learning Rate	Weight Decay	Batch (train/test)
FastRnet	100	0.001	0.005	3/1
ssd300_vgg16	1200	0.0001	0	64/1

B. Recognition

For the networks specialized in recognition, we used CEL (Cross Entropy Loss) as the loss function, which is typical for classification problems. The optimization function for both networks was AdamW. The networks were trained to recognize various types of traffic signs contained in the GTSRB dataset, divided into two sets: one for training, consisting of 94,342 images further divided into 43 classes (one for each type of sign), and a test set consisting of 12,631 images of signs belonging to the 43 classes defined for training, mixed together.

Table VI: HyperParameters For Recognition Models

Name	Epochs	Learning Rate	Weight Decay	Batch Size
ModelCNN	50	1e-4	0	64
Model2	60	0.001	1e-5	64
ResNet18	60	1e-7	0	64
ResNet32	50	1e-4	0	64

IV. Results

A. Results For Recognition

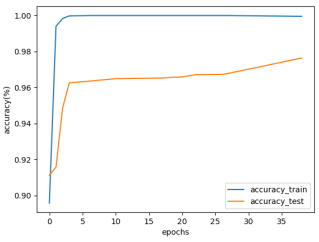
In this section, we present the experimental results obtained for the networks under analysis. As we can see from Table VII, our network shows the highest level of accuracy in performing the recognition task, with a score of 97.6%, achieved thanks to the use of STN modules and image pre-processing. Despite having a structure very similar to our network, Model2 achieves an accuracy of only 94.5%. This result can be explained by the number of convolutional blocks used by the networks. Our network, compared to Model2, implements more and deeper convolutional blocks, demonstrating that increasing the number and depth of CNN blocks enhances the network's ability to extract features from input images.

Table VII: Recognition Accuracy

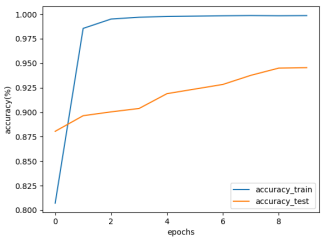
Model	Accuracy
ModelCNN (ours)	**97,6**
Model2	94,5
ResNet18	94,6
ResNet32	95,4

The ResNet networks, on the other hand, show good recognition capabilities with an accuracy of 94.6% (ResNet18) and 95.4% (ResNet32), but they do not surpass our network. This confirms that ResNet networks generally perform well in many computer vision tasks but do not achieve the level of accuracy of networks specialized in a single task. In fact, ResNet networks are typically adopted for input data of substantial size to properly feed the networks. In our case, the input images are small, at 48x48 pixels, which is insufficient for training ResNet networks. Our network, however, has been specifically adapted to process small-sized images, thus achieving not only a higher level of accuracy but also better performance compared to the ResNet networks.

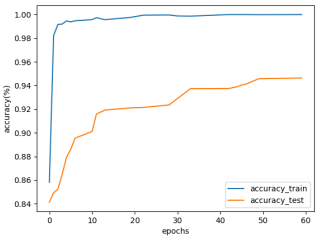
We can therefore conclude that experimentally, our network is overall the best for recognition compared to the other networks under analysis.



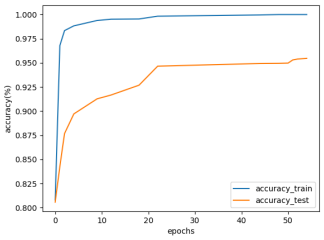
Accuracy ModelCNN(ours)



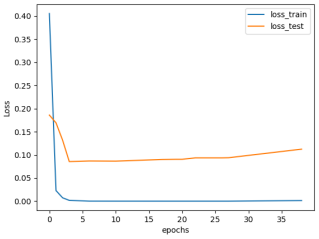
Accuracy Model2



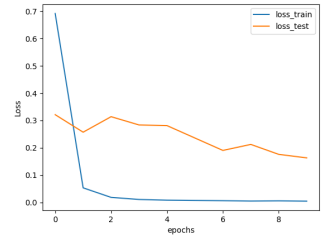
Accuracy ResNet18



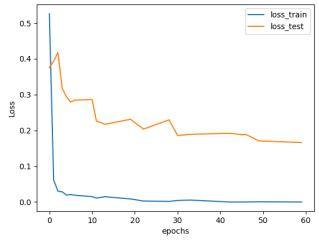
Accuracy ResNet32



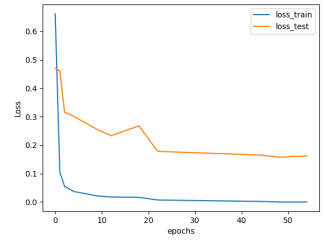
Loss ModelCNN(ours)



Loss Model2



Loss ResNet18



Loss ResNet32

Figure 4: Example of recognition (Fast-RCNN) + detection (ModelCNN): Identify the road signs and specify their type.

B. Results For Detection

In this section, we present the experimental results obtained in the analysis of the Fast-RCNN and SSD300-VGG16 networks during the road sign detection task. The networks in the testing phase demonstrated different recognition capabilities based on the nature of the test data.

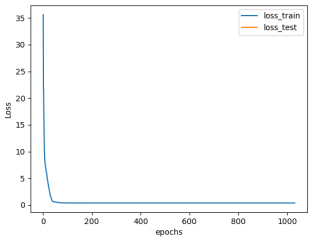


Figure 5: Loss ssd300

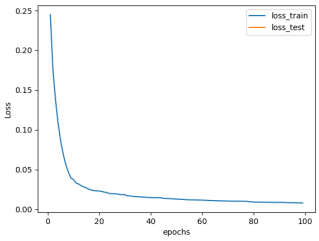


Figure 6: Loss Fast-RCNN

During the testing phase, we experimentally administered two types of data: a video containing road signs and the GTSDB test set containing images of natural environments where one or more road signs appear. We observed the results obtained in order to estimate the accuracy of the networks. From the direct observation of the produced images, it was undoubtedly clear that the Fast-RCNN network had greater accuracy in photo detection compared to the SSD300-VGG16 network, as shown in image X. In the video tests, however, the SSD300-VGG16 network produced better results with excellent performance, given the simple structure of the network, and decent identification capabilities.

The Fast-RCNN also showed a good degree of accuracy, at the expense of efficiency, as it is very slow in processing video frames due to its high complexity, thus generating high latency in video playback. In both cases, the networks demonstrated their ability to adapt to the more specific task of object recognition without the need for architectural modifications, while still achieving satisfactory levels of accuracy.

V. Conclusions and Future Work

In this section, we present the conclusions we have reached following the experimentation. Having implemented and evaluated the best models for detection and recognition and established the feasibility of using ML in TSRD activities, we were able to create a complete TSRD system using a Jupyter notebook (Python) and an experimental video demo of the system capable of recognizing and classifying encountered road signs. This demonstrates the feasibility of developing ML-based systems in the future that will integrate into more complex systems such as autonomous driving and smart-city systems.

A future improvement for the task is to enhance the efficiency and accuracy of the presented models so that they can be adopted in the real world for real-time road sign recognition, without issues of latency or false positives.

VI. Project Link

Click on one of the entries of the list to access the wanted thing

A. Drive

B. GitHub

C. Weights

Acknowledgement

References

[1] Alvario Arcos-Garcia, Juan A. Alvarez-Garcia, and Luis M. Soria-Morillo. Deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimization methods. 2018.

[2] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. 2016.

[3] Yanghao Li, Saining Xie, Xinlei Chen, Piotr Dollar, Kaiming He, and Ross Girshick. Benchmarking detection transfer learning with vision transformers. 2021.

[4] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. 2016.