International Journal of Transport Development and Integration
https://www.acadlore.com/journals/IJTDI

# Traffic Sign Classification using Spatial Transformer Networks (STN) based Convolutional Neural Networks (CNN)

Burcu Tiryaki*, Emin Argun Oral

Department of Electrical and Electronics Engineering, Faculty of Engineering, Ataturk University, 25240 Erzurum, Turkey

* Correspondence: Burcu Tiryaki (burcutiryaki@atauni.edu.tr)

**Received:** 09-22-2025 **Revised:** 11-17-2025 **Accepted:** 11-21-2025

**Citation:** B. Tiryaki and E. A. Oral, "Traffic sign classification using Spatial Transformer Networks (STN) based Convolutional Neural Networks (CNN)," *Int. J. Transp. Dev. Integr.*, vol. 9, no. 4, pp. 840–850, 2025. https://doi.org/10.56578/ijtdi090411.

**Abstract:** Recognition of traffic signs by drivers is essential for ensuring road safety. Recently, with the growing demand for driver assistance systems and autonomous vehicles, traffic sign recognition has become increasingly important. In this study, Spatial Transformer Networks (STN) integrated with Convolutional Neural Networks (CNN) were used to classify traffic signs. STNs minimize the effects of geometric distortions by applying affine transformations to images, thereby improving classification performance. This study focuses on adapting and optimizing an STN-based CNN model specifically for the Russian Traffic Signs Dataset (RTSD) to achieve higher classification accuracy. The proposed model was trained and tested on the RTSD. First, the proposed CNN model was trained on the RTSD-R1 and RTSD-R3 datasets, achieving accuracy rates of 89.15% and 94.3%, respectively. Then, by integrating STN into the CNN model, the proposed model was trained on the RTSD-R1 and RTSD-R3 datasets, achieving accuracy rates of 93% and 95%, respectively. These results demonstrate that incorporating STNs into the CNNs is effective in improving traffic sign classification performance.

**Keywords:** STN; CNN; Traffic sign; Classification; RTSD

## 1 Introduction

Nowadays, identification and classification of traffic signs has become essential components of driver assistance systems and autonomous vehicles. Traffic sign recognition and detection are essential for road safety and advanced driver-assistance systems [1, 2]. Traffic signs provide essential information about road conditions and help maintain traffic order for both drivers and pedestrians. It plays a very important role in regulating traffic and facilitating safe driving. Therefore, traffic signs are vital for drivers and pedestrians. Traffic signs are difficult to recognize under poor illumination, partial occlusion, outdated signage, or adverse weather conditions, outdated traffic signs, adverse weather conditions such as rain, snow, fog, etc. and safety risks of following traffic signs while driving. These situations may also lead to accidents. In order to minimize the number of accidents, automatic systems have been developed that automatically detect traffic signs and warn drivers during driving. In this sense, driver assistance systems are utilized to detect and recognize traffic signs and make real-time decisions [3, 4]. With the increasing interest in autonomous vehicles in recent years, on the other hand, detection of traffic signs has become very important for environmental perception. In this context, traffic sign detection is required to locate the signs in the scene before classifying them into specific categories.

Various environmental factors and the visual similarity of traffic signs can adversely affect real-time performance of such autonomous systems. To resolve this, machine learning methods are used both to recognize and classify traffic signs processing image related features. In order to obtain such features from images, deep learning methods, a novel machine learning approach, can be used to extract general and hidden features. For this purpose, Convolutional Neural Networks (CNN) are employed. CNNs have been used in many studies such as visual recognition, speech recognition and natural language processing in recent years. A traditional convolutional neural network consists of one or more blocks of convolution and pooling layers, followed by one or more fully connected (FC) layers and an output layer. Convolutional layers consist of filters to calculate different feature maps. The pooling layer, on the other hand, takes a small region as input and produces a single output for downsampling. The fully connected layer, produces output from the inputs it receives from the final pooling or convolutional layer [5].

In the literature, there are some studies on recognizing and classifying traffic signs. Luo et al proposed a new data-driven system to recognize all traffic sign categories, including both symbol-based and text-based signs, in video, captured by a car-mounted camera. Li and Wang [6] designed the sensor using faster R-CNN neural network and MobileNet model. In the study, shape and color information was used to improve recognition of small traffic signs. An efficient CNN with an asymmetric core is used as the classifier of traffic signs. Song et al. proposed a convolutional neural network that reduces parameters and speeds up networks. Network performance is evaluated on the Tsinghua–Tencent 100K dataset. Experimental results showed that the model outperformed Fast R-CNN and Faster R-CNN [7]. Zhu and Yan [8] evaluated YOLOv5 performance for traffic sign recognition in their study. They showed that YOLOv5 has better performance when compared with SSD detector.

In the study conducted by Alawaji et al. [9], the aim was to enhance the safety of driverless transportation systems operating on fixed routes by detecting and recognizing traffic signs using computer vision and deep learning techniques. The research primarily employed CNN-based approaches. In this framework, a structure was developed in which convolutional layer parameters were shared across different tasks, utilizing pre-trained models such as InceptionResNetV2 and DenseNet201. Initially, single-task models were compared on symbol-based traffic signs, and subsequently, a multi-task learning approach was applied to further improve accuracy. In the study conducted by Kandasamy et al., the focus was on recent advancements in machine perception, particularly in the context of autonomous vehicles. The accurate detection and interpretation of road signs by autonomous systems are critically important for ensuring safety and efficiency on the roads. In this research, a novel algorithm called TrafficSignNet was developed using a custom dataset that reflects the diversity of traffic signs in India, consisting of 12 main categories and 7 subcategories. The algorithm was specifically designed to recognize traffic signs indicating speed limits, turns, zones, and road bumps. The model was trained on 4,962 images and evaluated on 705 real traffic images, demonstrating high accuracy under varying lighting conditions [10]. In the study conducted by Enan and Chowdhury [11], a defense method was developed against adversarial patch attacks (APA), a type of physical attack targeting the computer vision module of autonomous vehicles. The researchers proposed a GAN-based single-stage defense strategy to enhance the robustness of traffic sign classifiers against such attacks. The proposed method can protect different classes of traffic signs without requiring prior knowledge and increases classification accuracy by up to 80.8% under APA conditions. Moreover, since the approach is model-agnostic, it can be applied to any traffic sign classification model.

Although existing deep learning models achieve high accuracy, their performance may vary under geometric distortions, non-uniform lighting, and viewpoint changes. Spatial Transformer Networks (STN) have been proposed to address these issues by learning spatial transformations that normalize input images before classification. In this study, a CNN architecture integrated with STN modules is developed and optimized for the Russian Traffic Signs Dataset (RTSD) dataset. Unlike previous works, this study does not introduce a new algorithmic framework but rather designs a more suitable network architecture specifically adapted to the RTSD dataset. The proposed model aims to enhance classification accuracy by mitigating geometric variations while maintaining computational efficiency.

The rest of this article is organized as follows: Section 2 describes our proposed method for traffic sign classification. Section 3 shows the experimental results. Section 4 shows the discussion and finally Section 5 shows the conclusion and future work.

## 2 Materials and Methods

In this study, a traffic sign recognition network integrating a convolutional neural network and a spatial transformer module is proposed. Figure 1 illustrates the main block diagram of the proposed traffic sign recognition system. It is composed of two main blocks: a STN and classifier network. In the traffic sign recognition framework, traffic sign images are first resized to different resolutions of $20 \times 20$, $60 \times 60$, and $90 \times 90$ before being fed into the spatial transformer network. The STN performs an affine transformation on the input image to improve classification performance by removing unnecessary background elements and enhancing focus [12]. The STN consists of a proposed localization network to detect the necessary affine transformation parameters, including cropping, translation, rotation, scale, and skew, a grid generator and a sampler, as shown in Figure 1.

STN identify the appropriate region in the input image and generate a set of affine parameters ($\theta$) for the transform. This information is then forwarded to the grid generator. Using the supplied affine transformation parameters, the system performs operations such as normalization, cropping, and scaling on the relevant region of the input image. During the sampling phase, the output image is reconstructed using the parameters obtained from the grid generator. As a result, input traffic sign images captured at non-ideal angles or containing geometric distortions are corrected.

The proposed architecture integrates the STN directly with the CNN classifier in an end-to-end manner. The STN module first receives the input image and performs affine transformations, including translation, scaling, rotation, and cropping, to correct geometric distortions and emphasize the region of interest. The transformed output feature maps generated by the STN are then passed as tensors to the initial convolutional layer of the CNN classifier. This sequential tensor flow allows the network to automatically learn spatially normalized representations before feature extraction and classification.

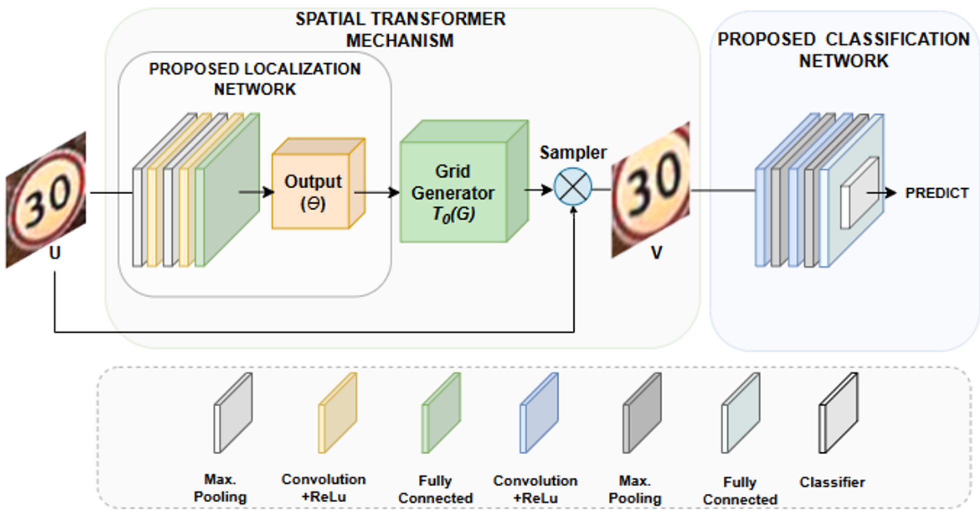

**Figure 1.** Block diagram of the proposed traffic sign recognition system

In the final classification step, the output image of the STN is processed by proposed classification network of traffic signs. This proposed classification network is based on CNN and described in the following sub-sections.

## 2.1 Dataset

The RSTD consists of frames captured at a rate of 5 frames per second from a video recorder provided by the company Geocenter Consulting [13]. The video was captured at different times of a day, in different seasons and in different weather conditions. For the evaluation of the detectors, the dataset was divided into three groups as RTSD-D1, RTSD-D2 and RTSD-D3. For the evaluation of classifiers, the dataset is divided into two groups: RTSD-R1 and RTSD-R3. Here, RTSD-R1 consists of traffic signs cropped from the images in the RTSD-D1 dataset while RTSD-R3 consists of traffic signs cropped from the images in the RTSD-D3 dataset. Among these, RTSD-R1 consists of 66 classes with 25.432 training and 7.551 test images, and RTSD-R3 consists of 106 classes that include 70.687 training and 22.967 test images [12]. Figure 2 shows examples of images in the RTSD-R1 and RTSD-R3 datasets. The RTSD-R1 and RTSD-R3 datasets were used with their predefined training and test partitions as provided by the original RTSD benchmark. Both subsets are class-balanced, ensuring proportional representation of each traffic-sign category.

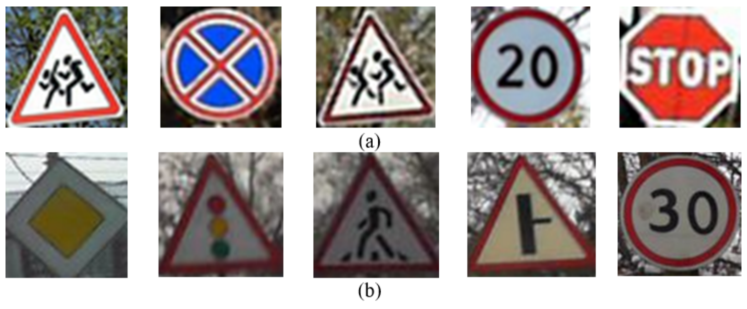

**Figure 2.** Example images of (a) RTSD-R1 and (b) RTSD-R3 dataset

## 2.2 MatConvNet

MatConvNet is an open source MATLAB toolbox developed by the authors of the VLfeat library [14]. It uses CNN in image processing applications. Since it is designed as simple and flexible, it enables rapid prototyping of CNN architectures. Convolutional neural network applications handle large amounts of data. MatConvNet supports working with large datasets and training of complex models by enabling calculations on both CPU and GPU. Therefore, the main reason for the development of the MatConvNet toolbox is to ensure that the work is done in a more efficient environment. Integration into the MATLAB working environment is possible [15]. Layers are easily combined and extended when building the CNN architecture. The layers in the neural network are arranged in arrays. Interconnections can also be made. Basically, the network structure created in MatConvNet is shown in Figure 3.

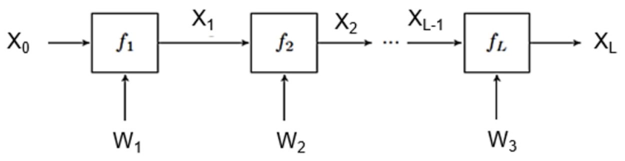

**Figure 3.** MatConvNet chain block structure [15]

In the f block, $X_0$ shows the input data and W shows the parameter values. The function obtained when the network is evaluated from left to right;

$$X_L = f\left(X_0; W_l; \ldots; W_L\right) \tag{1}$$

MatConvNet includes two wrappers, SimpleNN and DagNN, designed to accommodate different types of network architectures. While SimpleNN is suitable for networks with linear structure, DagNN structure is used for more complex networks.

MatConvNet performs a MATLAB function as $y$ = vl_nn $<$ block $>$ $(x, w).x$ represents the input, w represents the parameter values, and y returns the output string. Functions also work backwards to calculate derivatives.

Various blocks such as convolution, pooling and activation (ReLU) are used in MatConvNet.

Convolution:

The convolution block is implemented by the vl_nnconv function. In function $y$ = vl_nnconv $(x, f, b), x$ shows the input map, f the filter bank and b the slope. The filter is circulated over the image according to the specified number of steps.

Activation function: Two activation functions are supported:

ReLU: vl_nnrelu calculates the rectified linear unit layer.

$$y_{ijd} = \max\left\{0, x_{ijd}\right\} \tag{2}$$

Sigmoid: vl_nnsigmoid calculates sigmoid.

$$y_{ijd} = \sigma\left(x_{ijd}\right) = \frac{1}{1 + e^{-x_{ijd}}} \tag{3}$$

### 2.3 STN

The spatial transformation mechanism consists of a localization network, grid generator, and sampler. While CNNs have limited ability to perform spatial transformations on images, STNs overcome this limitation by applying operations such as scaling, rotation, and shearing to each input data. The transformations are applied across the entire feature map, eliminating the need for manual scaling of the data [16]. STNs identify important regions in images and adjust their poses to be processed effectively in subsequent layers, thereby reducing geometric variability in the data. Transformation networks placed in CNNs are trained by back propagation, allowing the CNN architecture to be trained end-to-end.

STNs consist of three parts: Localization Network, Grid Generator and Sampling.

#### 2.3.1 Localization network

The purpose of the network is to determine the affine transformation parameters to be applied on the input map. The network can have any number of layers, but the last layer must be a regression layer with six output neurons to generate the transformation parameters ($\theta$). Affine transform parameters are used to correct images with geometric distortions. Spatial transformations perform transformation operations in the relevant regions, and the transformation information is compressed in the weights of the localization network.

The localization network receives the input feature map $U \epsilon R^{H\times W\times C}$. Where U is the input map, H is the height, W is the width and C is the number of channels [16]. The output of the localization network is represented by Eq. (4).

$$\Theta = f_{loc}(\mathrm{U}) \tag{4}$$

where, $f_{loc}$ is the localization network function and $\theta$ denotes the 6 -dimensional affine transformation parameters.

#### 2.3.2 Grid generator

Creates a coordinate grid corresponding to each pixel in the output image using the transformation predicted by the localization network in the input image. Each output pixel is computed by applying a sampling kernel focused on a specific location in the input feature map. In general, the output pixels are defined to be located on a regular grid as in Eq. (5).

$$G = \{Gi\}, Gi = \left(x_i^t, \mathcal{Y}_i^t\right) \tag{5}$$

where, G grid generator, $x_i^t$, $y_i^t$ denotes the positions of i pixels. In the generated output feature map $V \epsilon R^{H' \times W' \times C}$, $H'$ and $W'$ indicate the height and width of the grid. The number of channels (C) is the same for input and output [17].

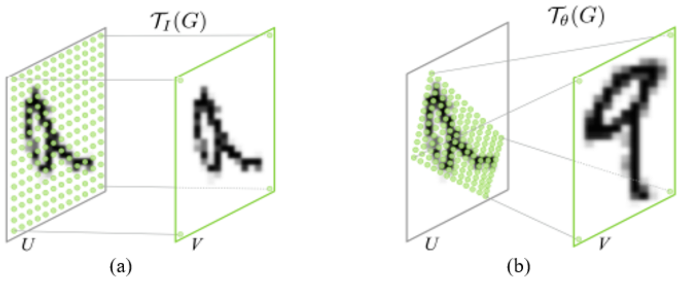

**Figure 4.** Grid Generator [17]

Figure 4 shows how to obtain the V output from the U input. In subgraph (a) of Figure 4, the grid generator is defined as $G = \tau_\imath(G)$. In subgraph (b) of Figure 4, the grid generator is defined as $\tau_\theta(\mathrm{G})$. Here I is the transformation parameter and $\tau_\theta$ is the affine transformation parameter. Eq. (6) illustrates the affine transformation.

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \tau_{\theta(G_i) = \mathrm{A}_\theta} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} \tag{6}$$

where, $\mathrm{A}_\theta$ is the affine transformation. $(x_i^s, y_i^s)$, defines the coordinates in the input feature map, $(x_i^t, y_i^t)$, defines the positions in the output feature map. Since the affine transformation in Eq. (6) allows clipping, translation, scaling, rotation and warping operations, 6 parameters must be generated by the localization network [16].

#### 2.3.3 Sampling

The feature map and sampling grid are used as inputs to the sampler, and the sampler produces the output map by sampling from the input at the grid points. It produces the output V based on the new coordinate sets $(x_i^t, y_i^t)$, coming from the grid generator. Sampling is defined by Eq. (7).

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c k\left(x_i^s - m; \Phi_x\right) k\left(\mathcal{Y}_i^s - n; \Phi_y\right) \tag{7}$$

where, $(x_i^s, y_i^s)$ defines the positions of the V output in pixel i, $\Phi_x$ ve $\Phi_y$ defines the sampling kernel parameters. $U_{nm}^c$ defines the value of the input at position n, m in channel c , and $V_i^c$ defines the output value of pixel i at position $(x_i^t, y_i^t)$ in channel c . Sampling is done the identical way for each channel of the input, so each channel is converted the identical way [17].

### 2.4 Proposed Localization Network

In the proposed localization network, a small convolutional neural network model is used. This model consists of a convolutional structure with layers such as Maxpool, Conv, ReLU, Maxpool, Conv, ReLU, followed by a Fully Connected Layer, ReLU, and an output layer. To ensure that the localization network remains compatible with all three input resolutions (20 × 20, 60 × 60, and 90 × 90), the spatial dimensions of the intermediate feature maps were recalculated for each input size. The kernel sizes, strides, and pooling operations were checked against each resolution to confirm that the resulting feature maps retain valid spatial dimensions before entering the fully connected layer. The architecture itself—number of layers and channel depths—remains unchanged across resolutions; only the spatial dimensions vary according to the input size. This adjustment prevents the feature maps from collapsing to overly small or non-integer sizes and allows the same localization network structure to operate consistently at different resolutions.The parameters of the proposed localization network are detailed in Table 1. In Table 1, the parameters calculated for the 90 × 90 size are given as an example. The same parameters were used for the RTSD-R1 and RTSD-R2 datasets of the same size.

**Table 1.** Parameters of the proposed STN architecture

| Layers | Kernel Size | Filters | Units | Stride | Padding | Θ |
|---|---|---|---|---|---|---|
| Max Pooling | 2 × 2 | | | 2 | 0 | |
| Convolution | 12 × 12 | 70 | | 1 | 0 | |
| ReLU | | | | | | |
| Max Pooling | 2 × 2 | | | 2 | 0 | |
| Convolution | 11 × 11 | 110 | | | | |
| ReLU | | | | | | |
| Fully Connected | | | 180 | 1 | 0 | |
| ReLU | | | | | | |
| Output | 1 × 1 | | | 1 | 0 | 6 |

## 2.5 Proposed Classification Network

We constructed a CNN model without spatial transformer layers to evaluate their impact on the recognition of traffic signs. The model consists of a convolutional structure with Conv, ReLU, MaxPool, Conv, ReLU, and MaxPool layers. Then, following these layers, a fully connected layer and a ReLU layer are added, respectively. With the convolutional layers used in the network, features in the input image are identified, a feature map is created, and the data is processed. With the MaxPooling layer, the computational load is reduced by decreasing the size of the feature maps. A fully connected layer was used to make the features from these layers available for classification. A classifier layer has been added to classify traffic signs.

The layers used in the proposed model and the parameter values for each layer are shown in Table 2. In Table 2, the parameters calculated for the 90 × 90 size are given as an example. Layer size, stride, and padding values are calculated based on the specified image size. The stride value of each convolutional layer is set to 1, while the stride value of each max pooling layer is set to 2, allowing for spatial subsampling calculations in this layer. Additionally, the kernel size values for each layer have been adjusted according to these calculations.

**Table 2.** Parameters of the proposed CNN architecture

| Layers | Kernel Size | Filters | Units | Stride | Padding | Number of Class |
|---|---|---|---|---|---|---|
| Convolution | 10 × 10 | 32 | | 1 | 0 | |
| ReLU | | | | | | |
| Max Pooling | 2 × 2 | | | 2 | 0 | |
| Convolution | 7 × 7 | 32 | | 1 | 0 | |
| ReLU | | | | | | |
| Max Pooling | 2 × 2 | | | 2 | 0 | |
| Convolution | 5 × 5 | 64 | | 1 | 0 | |
| Fully Connected | | | 128 | 1 | 0 | |
| ReLU | | | | | | |
| Classifiers | 1 × 1 | | | 1 | 0 | 66 / 106 |

## 3 Experimental Results

The experimental results are presented for two cases: one involving the application of the CNN-based classification network and the other involving the application of the STN architecture to improve performance. All experiments were run on a computer equipped with an Intel Core i7-7700HQ CPU, 16 GB of RAM, and a Nvidia GeForce GTX 1050 discrete GPU with 8 GB of internal memory. In the optimization parameters used during all training stages, Stochastic Gradient Descent with Momentum (SGDM) was employed as the optimization algorithm, the mini-batch size was set to 80, and the number of epochs was set to 50. The learning rate was set to $1 \times 10^{-3}$, a momentum of 0.9 was used, and an L2 regularization term with a weight decay of $1 \times 10^{-4}$ was applied to prevent overfitting.

The model contains approximately 3 million parameters (12.2 MB). The average inference time per image was measured to be 11.0 ms per image, corresponding to roughly 90 frames per second (FPS). These results indicate that the proposed network achieves real-time performance while maintaining a moderate computational load.

All experiments were conducted using the predefined RTSD-R1 and RTSD-R3 training–test splits provided by the official benchmark. Since no additional random sampling or reshuffling was applied, the results are deterministic. Each experiment was run once.

The confidence intervals (CI) of the accuracy rates were calculated according to Eq. (8), where z denotes the constant value corresponding to a 95% confidence level (1.96) and n represents the size of the test dataset [18–20].

$$CI = Acc \pm z\sqrt{\frac{Acc(1-Acc)}{n}} \tag{8}$$

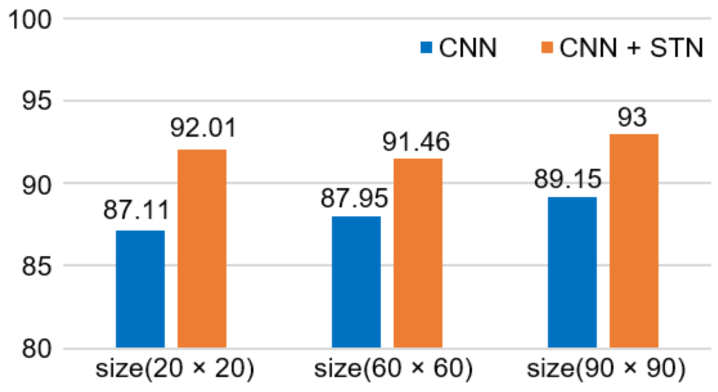

**Figure 5.** Classification results of RTSD-R1 dataset

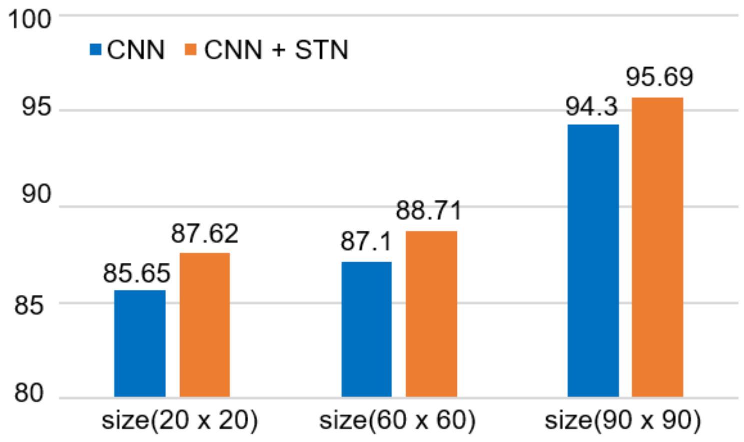

**Figure 6.** Classification results of RTSD-R3 dataset

To observe the effect of the STN architecture on classification performance, each dataset was first classified using the proposed Traffic Sign Classification Network (TSCN), and then the performances were compared by adding the STN mechanism to this network. When the RTSD-R1 dataset was resized to $20 \times 20$ and classified using the TSCN network, an accuracy of $87.11 \pm 0.0076\%$ was achieved, while an accuracy of $92.01 \pm 0.006\%$ was obtained by adding the STN mechanism to the network. Similarly, when the RTSD-R1 dataset was resized to $60 \times 60$, an accuracy of $87.95 \pm 0.0074\%$ was achieved using the TSCN network, while an accuracy of $91.46 \pm 0.006\%$ was obtained by adding the STN mechanism to the network. When the RTSD-R1 dataset was resized to $90 \times 90$, an accuracy of $89.15 \pm 0.007\%$ was achieved using the TSCN network, while an accuracy of $93 \pm 0.0057\%$ was obtained by adding the STN mechanism to the network. These results are shown in Figure 5.

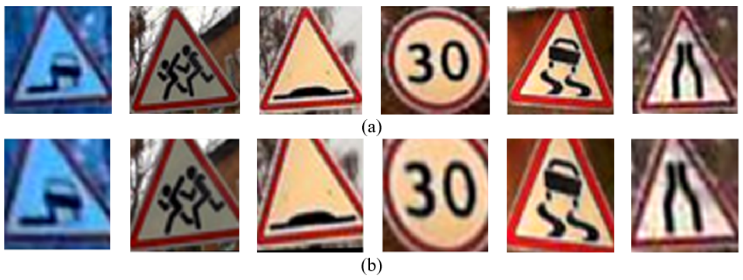

**Figure 7.** (a) Original images and (b) STN resulting images

Similarly, the RTSD-R3 dataset was resized and classified into three different dimensions. When the RTSD-R3 dataset was resized to 20 × 20 and classified using the TSCN network, an accuracy of 85.65 ± 0.0045% was achieved, while an accuracy of 87.62 ± 0.0043% was obtained by adding the STN mechanism to the network. When the RTSD-R3 dataset was resized to 60 × 60, an accuracy of 87.1 ± 0.0043% was achieved using the TSCN network, while an accuracy of 88.71 ± 0.0041% was obtained by adding the STN mechanism to the network. When the RTSD-R3 dataset was resized to 90 × 90, an accuracy of 94.3 ± 0.003% was achieved using the TSCN network, while an accuracy of 95.69 ± 0.0026% was obtained by adding the STN mechanism to the network. These results are shown in Figure 6. Image examples obtained from the output of the STN mechanism are also shown in Figure 7.

To provide a clearer understanding of failure cases, several misclassified examples obtained from the STN + TSCN model were examined. A large proportion of the errors occurred in traffic sign images that were recorded at smaller scales and lower resolutions. In the analysis conducted on the RTSD-R1 subset, accuracy exceeded 95% for traffic signs in the prohibitory and warning categories, whereas accuracy rates remained within the 85–90% range for classes that included information signs with lower resolution. Similarly, in the RTSD-R3 dataset, misclassifications were predominantly observed in small-scale signs captured under low-light conditions or partially occluded. In the information sign category, accuracy rates were observed to remain between 85% and 90%. These findings indicate that although the STN module enhances geometric robustness, challenges persist in fine-grained classes characterized by high intra-class similarity or low-quality visual features.

Overall, the results indicate that the Spatial Transformer Network positively impacts classification performance.

## 4 Discussion

Traffic signs were classified by adding a classification network to the STN mechanism, which was created using the MatConvNet toolbox. The results indicate that the STN mechanism positively impacted classification performance.

In previous studies, the STN mechanism was employed to classify traffic signs, resulting in strong classification performance on benchmark datasets. In the study by Haloi [21], a network was designed to classify traffic signs using a spatial transformer layer and a custom initialization module. In the study conducted on the German Traffic Signs Recognition Benchmark (GTSRB) dataset, an accuracy of 99.81% was achieved. In the study conducted by Zhang et al. [22], a multi-column STN consisting of CNNs and STNs was proposed to address the issue of CNNs not adapting well to the spatial diversity of images. In their research on the GTSRB dataset, an impressive classification accuracy of 99.75% was achieved. In the study by Arcos-García et al. [16], a Deep Neural Network with convolutional layers and STNs was used to perform various classification experiments on public traffic sign datasets from Germany and Belgium. In the classification conducted on the GTSRB dataset using a CNN structure with the SGD optimization algorithm and without STNs, an accuracy of 98.31% was achieved. When three STN blocks were added to this CNN structure, an accuracy of 99.49% was achieved, demonstrating the positive impact of the STN mechanism on classification. Lim et al. [23] propose a new convolutional neural network consisting of a spatial transformer network and a multi-structure convolutional neural network. The classification performances of the basic CNN without the STN mechanism and the multi-column convolutional neural network with STNs (SPMCNN) on the GTSRB dataset were achieved as 91% and 97.45%, respectively.

In our study, the RTSD-R1 and RTSD-R3 traffic sign recognition datasets were used to evaluate the performance of the STN mechanism on classification. In the study conducted by Shakhuro and Konushin [12], a dataset consisting of Russian traffic sign images was developed. RTSD-R1 and RTSD-R3 datasets created for classification of this dataset were classified in a proposed convolutional neural network and an accuracy of 90.78% and 92.9% was obtained, respectively. In our study, both datasets were resized to 20 × 20, 60 × 60, and 90 × 90, respectively, and classified using the TSCN and STN + TSCN models. The localization network of the STN mechanism and the proposed TSCN parameters were recalculated for each dimension. In the RTSD-R1 dataset, the addition of the STN mechanism increased the classification accuracy by 5.6% for the 20 × 20 size, 3.9% for the 60 × 60 size, and 4.3%

for the 90 × 90 size. In the RTSD-R3 dataset, the addition of STN mechanism increased the classification accuracy by 2.3% for the 20 × 20 size, 1.9% for the 60 × 60 size and 1.5% for the 90 × 90 size. The highest accuracy rates were achieved as 93% for the RTSD-R1 dataset and 95.69% for the RTSD-R3 dataset in the STN + TSCN model with 90 × 90 dimensions.

The results obtained at different image sizes (20 × 20, 60 × 60, and 90 × 90) show that image resolution has a noticeable effect on the contribution of the STN mechanism. At smaller image sizes, such as 20×20, the STN provided a greater improvement in accuracy, especially for the RTSD-R1 dataset. This is because low-resolution images lose spatial details, and the STN helps to correct geometric distortions and focus on the most relevant regions. As the image size increased, the improvement provided by the STN became smaller, since higher-resolution images already contained enough spatial information for the base network (TSCN) to perform well. These results indicate that the STN mechanism is particularly useful when images are low in resolution or not well aligned, while its effect decreases when the input data already provide clear spatial information. The improvement in performance can be attributed to the ability of the STN to learn spatial transformations such as rotation, scaling, and translation automatically, allowing the network to correct geometric distortions and focus on the most informative regions of the image. This effectively reduces intra-class variability caused by different viewing angles, lighting conditions, or partial occlusions, leading to more stable feature extraction by the convolutional layers.

However, it was also observed that when the dataset images were already well-aligned and contained limited spatial variations (as in the RTSD-R3 subset), the incremental gain from adding STN was relatively smaller. This suggests that while the STN mechanism enhances robustness under non-ideal conditions, its contribution becomes less significant when the training data are already normalized or when excessive transformations cause the loss of fine-grained details. These findings highlight that the effectiveness of STN is closely tied to the geometric diversity of the dataset and the complexity of the visual distortions present in the input images.

Compared with recent lightweight detection models such as YOLOv8 or transformer-based traffic-sign recognizers, the proposed STN-enhanced CNN model serves a different purpose and operates with a substantially smaller computational footprint. YOLOv8-nano, for example, contains approximately 3–4 million parameters and is designed for real-time object detection, whereas transformer-based architectures typically exceed tens of millions of parameters and require higher computational resources for multi-head attention operations. In contrast, our model contains only 3 million parameters with a 12.2 MB model size and achieves an average inference time of 11 ms, making it suitable for resource-constrained or embedded environments where detection is not required and classification alone is sufficient. While YOLO-based or transformer-based systems offer end-to-end detection-plus-classification pipelines, the goal of this study is to enhance classification accuracy under geometric distortions. Thus, the proposed STN-integrated CNN can be considered a lightweight and distortion-robust alternative when a standalone classification module is required or when computational resources are limited.

The results indicate that while the TSCN model exhibited high classification performance on its own, the addition of the STN mechanism further improved the classification performance.

## 5 Conclusion and Future Work

CNNs provide powerful architectures for image processing applications; however, their ability to perform spatial transformations on data is limited. Correcting geometrically corrupted or distorted images is crucial for improving classification performance. Although fully connected layers in CNNs partially handle this task, their effectiveness remains limited on large and complex datasets due to the small receptive fields of convolutional filters. In this study, STNs were employed to identify the region of interest by applying transformations such as rotation, scaling, and cropping, while simultaneously removing irrelevant areas from the image. These transformations reduce the need for extensive manual data augmentation. Spatial transformations apply geometric transformations to the input feature map, enabling CNNs to achieve computationally efficient spatial invariance [15]. Thus, the conventional need for data augmentation typically used to improve performance when training data is limited is effectively eliminated.

In the study, STNs were integrated into the TSCN architectures. The TSCN and STN+TSCN structures were tested on the RTSD-R1 and RTSD-R3 datasets, and it was observed that STNs had a positive effect on classification performance. This work contributes to the field of AI by demonstrating the practical integration of STN with CNNs. In our future work, we aim to achieve improved classification accuracy and robustness by enhancing the proposed localization network and traffic sign classification network for other AI applications.

Although MatConvNet provides a simple and MATLAB-integrated environment for rapid prototyping, it lacks some of the advanced features, flexibility, and computational efficiency available in modern deep learning frameworks such as PyTorch and TensorFlow. In future work, we plan to reimplement the proposed STN-based CNN architecture using these frameworks to enable faster training, more efficient GPU utilization, and easier integration with state-of-the-art modules.

Additionally, the model will be benchmarked against state-of-the-art architectures such as YOLOv8 and Transformer-based detectors to assess its relative performance and identify potential limitations. Future studies will further focus on incorporating attention mechanisms to enhance spatial feature discrimination and robustness against occlusions. Moreover, lightweight and real-time variants of the proposed architecture will be developed to optimize its applicability for embedded systems and autonomous driving platforms.

Through these improvements, the current framework is expected to evolve into a more adaptive, efficient, and deployable solution for intelligent transportation and other real-world computer vision applications.

## Author Contributions

Software, B.T.; Data curation, B.T.; Formal analysis, B.T. and E.A.O.; Writing—review and editing, B.T. and E.A.O. All authors have read and agreed to the published version of the manuscript.

## Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## References

[1] S. Ahmed, U. Kamal, and M. K. Hasan, "DFR-TSD: A deep learning based framework for robust traffic sign detection under challenging weather conditions," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 5150–5162, 2022. https://doi.org/10.13140/RG.2.2.18341.86249

[2] B. Preeti and A. Kunjal, "Traffic sign classification using CNN," *Int. J. Appl. Sci. Eng. Technol.*, vol. 10, no. 2, 2022. https://doi.org/10.22214/ijraset.2022.40224

[3] Z. Ou, F. Xiao, B. Xiong, S. Shi, and M. Song, "FAMN: Feature aggregation multipath network for small traffic sign detection," *IEEE Access*, vol. 7, pp. 178 798–178 810, 2019. https://doi.org/10.1109/ACCESS.2019.2959015

[4] Y. Cinar, S. Taspinar, M. M. Saritas, and M. Koklu, "Feature extraction and recognition on traffic sign ımages," *Selcuk Univ. J. Eng. Sci.*, vol. 19, no. 4, pp. 282–292, 2020.

[5] A. Ghosh, A. Sufian, F. Sultana, A. Chakrabarti, and D. De, "Fundamental concepts of convolutional neural network," in *Recent Trends and Advances in Artificial Intelligence and Internet of Things, Intelligent Systems Reference Library (ISRL, volume 172)*, 2019, pp. 519–567. https://doi.org/10.1007/978-3-030-32644-9_36

[6] J. Li and Z. Wang, "Real-time traffic sign recognition based on efficient CNNs in the wild," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 3, pp. 975–984, 2018. https://doi.org/10.1109/TITS.2018.2843815

[7] S. Song, Z. Que, J. Hou, S. Du, and Y. Song, "An efficient convolutional neural network for small traffic sign detection," *J. Syst. Archit.*, vol. 97, pp. 269–277, 2019. https://doi.org/10.1016/j.sysarc.2019.01.012

[8] Y. Zhu and W. Q. Yan, "Traffic sign recognition based on deep learning," *Multimed. Tools Appl.*, vol. 81, no. 13, pp. 17 779–17 791, 2022. https://doi.org/10.1007/s11042-022-12163-0

[9] K. Alawaji, R. Hedjar, and M. Zuair, "Traffic sign recognition using multi-task deep learning for self-driving vehicles," *Sensors*, vol. 24, no. 11, p. 3282, 2024. https://doi.org/10.3390/s24113282

[10] K. Kandasamy, Y. Natarajan, K. R. S. Preethaa, and A. A. Y. Ali, "A robust TrafficSignNet algorithm for enhanced traffic sign recognition in autonomous vehicles under varying light conditions," *Neural Process. Lett.*, vol. 56, no. 5, p. 241, 2024. https://doi.org/10.1007/s11063-024-11693-y

[11] A. Enan and M. Chowdhury, "GAN-based single-stage defense for traffic sign classification under adversarial patch attack," *arXiv preprint*, 2025, Art. no. arXiv:2503.12567. https://doi.org/10.48550/arXiv.2503.12567

[12] V. I. Shakhuro and A. S. Konushin, "Russian traffic sign images dataset," *Comput. Opt.*, vol. 40, no. 2, pp. 294–300, 2016. https://doi.org/10.18287/2412-6179-2016-40-2-294-300

[13] Graphics and Media Lab, "Traffic sign recognition," 2025. https://graphics.cs.msu.ru/projects/traffic-sign-recognition.html

[14] MatConvNet, "MatConvNet: CNNs for MATLAB," 2025. https://www.vlfeat.org/matconvnet/

[15] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Proceedings of the 23rd ACM International Conference on Multimedia*, Brisbane, Australia, 2015, pp. 689–692. https://doi.org/10.1145/2733373.2807412

[16] A. Arcos-García, J. A. Alvarez-García, and L. M. Soria-Morillo, "Deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimisation methods," *Neural Netw.*, vol. 99, pp. 158–165, 2018. https://doi.org/10.1016/j.neunet.2018.01.005

[17] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, vol. 28, 2025.

[18] R. G. Babukarthik, V. A. K. Adiga, G. Sambasivam, D. Chandramohan, and J. Amudhavel, “Prediction of COVID-19 using genetic deep learning convolutional neural network (GDCNN),” *IEEE Access*, vol. 8, pp. 177 647–177 666, 2020. https://doi.org/10.1109/ACCESS.2020.3025164

[19] E. G. Moung, C. J. Hou, M. M. Sufian, M. H. A. Hijazi, J. A. Dargham, and S. Omatu, “Fusion of moment invariant method and deep learning algorithm for COVID-19 classification,” *Big Data Cogn. Comput.*, vol. 5, no. 4, p. 74, 2021. https://doi.org/10.3390/bdcc5040074

[20] A. N. Omeroglu, H. M. Mohammed, E. A. Oral, and S. Aydin, “A novel soft attention-based multi-modal deep learning framework for multi-label skin lesion classification,” *Eng. Appl. Artif. Intell.*, vol. 120, p. 105897, 2023. https://doi.org/10.1016/j.engappai.2023.105897

[21] M. Haloi, “Traffic sign classification using deep inception based convolutional networks,” *arXiv preprint*, 2015, Art. no. arXiv:1511.02992. https://doi.org/10.48550/arXiv.1511.02992

[22] J. Zhang, S. Duan, L. Wang, and X. Zou, “Multi-column spatial transformer convolution neural network for traffic sign recognition,” in *Advances in Neural Networks—ISNN 2018*, Minsk, Belarus, 2018, pp. 593–600. https://doi.org/10.1007/978-3-319-92537-0_68

[23] M. T. Lim, D. W. Kim, J. H. Chung, W. J. Ahn, S. K. Park, T. K. Kang, and C. S. Dongnam-gu, “Traffic sign recognition using spatial transformer network with multi-structure convolutional neural network,” *Int. J. Eng. Sci.*, vol. 12, no. 4, pp. 118–122, 2020. https://doi.org/10.36224/ijes.120401