SN Computer Science (2024) 5:682
https://doi.org/10.1007/s42979-024-03034-6

ORIGINAL RESEARCH

Unveiling Interpretability: Analyzing Transfer Learning in Deep Learning Models for Traffic Sign Recognition

Sadaf Waziry[1] · Jawad Rasheed[2] · Fahad Mahmoud Ghabban[3] · Shtwai Alsubai[4] · Harun Elkiran[2] · Abdullah Alqahtani[4]

Received: 1 March 2024 / Accepted: 2 June 2024

Abstract

Since the advent of automobiles and driver assistance technologies, traffic sign recognition has been of the utmost importance for Industry 4.0. In the driving system, good data pre-processing is critical. For such objectives, sophisticated transformations or fundamentally computational image processing approaches are out of the question. Convolutional Neural Networks (CNN) have been used to perform more object identification challenges, thus, improving most computer vision applications, both existing and new, because of their excellent recognition rate and rapid execution. This study introduces a method for recognizing traffic signs by utilizing a CNN-based model and the transfer learning concept. The TensorFlow library is used for training the underlying neural network model. The offered approach makes use of the German Traffic Sign Recognition Benchmark (GTSRB) and images from the Traffic Sign Images from Turkey (TSIT) databases. These datasets are dependable and vibrant, and they have been used to train many algorithms. Furthermore, after training the model, the proposed scheme acquired a testing accuracy is 99.44%.

Keywords CNN · Image processing · German traffic sign · Transfer learning

Introduction

Road accidents have become one of the world's major causes of death as the number of automobiles on the road has increased. In 2018, the World Health Organization (WHO) highlighted traffic accidents as a substantial public health issue, resulting in an annual toll of 1.35 million fatalities [1]. Responding to this crisis, the United Nations has set forth an ambitious objective aimed at halving road traffic casualties by 2030, as articulated in the Global Plan for the Decade of Action for Road Safety 2021–2030. To reduce traffic fatalities in autonomous systems, the transportation system is required to develop into a smart transportation system. Consequently, traffic sign recognition software has to be created to solve this problem. The traffic sign classification system is a module of the Advanced Driver Assistance System (ADAS), and it is one of the most important responsibilities in self-driving since it provides information about what sign is in the image for decision-making. In this work, traffic sign recognition is a technology that can detect traffic signs placed on the road, such as "no parking", "stop", "turn ahead", or "give way". The proposed technique uses a Deep Neural Network (DNN) class known as CNN to construct a model that may be utilized as a pre-trained model for fine-tuning and used for traffic sign detection and categorization.

DNN offers superior visual pattern recognition skills and is commonly employed in computer vision techniques [2]. A CNN or ConvNet is a specific form of DNN commonly employed for visual image analysis. The system is a Deep Learning model that processes an image by assigning learnable weights and biases to different items within the image to distinguish them from each other [3, 4]. A CNN requires much less pre-processing compared to other classification algorithms. Primitive filtering methods are manually

✉ Jawad Rasheed
jawad.rasheed@izu.edu.tr

1 Department of Software Engineering, Istanbul Aydin University, Isanbul 34295, Turkey

2 Department of Computer Engineering, Istanbul Sabahattin Zaim University, Istanbul 34303, Turkey

3 Faculty of Computer Science and Engineering, Taibah University, Medina, Saudi Arabia

4 College of Computer Engineering and Sciences, Prince Sattam bin Abdulaziz University, 11942 Al Kharj, Saudi Arabia

Published online: 28 June 2024

designed, but ConvNets can learn these filters with proper training.

A frequent task of machine learning techniques is to recognize things and categorize them. For instance, autonomous vehicles face a significant challenge in navigating busy roads safely, particularly in avoiding collisions with pedestrians, especially at night. Existing ADAS lack nighttime pedestrian detection capabilities. The researchers in [5] proposes a Deep Reinforcement Learning (DRL) framework utilizing Scale Invariant Faster Region-based Convolutional Neural Networks (SIFRCNN) to improve pedestrian detection, achieving a 2.3% reduction in miss rate compared to other CNN-based detectors.

Autonomous vehicle operations face challenges in detecting and recognizing traffic signs in diverse environments, especially with small signs. Global car manufacturers are racing to develop driverless vehicles, extending the concept to various moving platforms like wheelchairs and recreational carts. Achieving this entails safe street navigation, obstacle detection, and traffic sign interpretation, leveraging smart technologies. There have been numerous studies that have examined the impact of machine learning methods on traffic sign recognition using different techniques and approaches. For instance, the study by [6] contrasts and evaluates two fundamental approaches for detecting and recognizing traffic signs. The first technique utilizes color segmentation and convolutional neural networks (C-CNN), whereas the second method relies on rapid region-based convolutional neural networks (Fast R-CNN). A comparative analysis of Convolutional Neural Networks (CNN) and Radial Basis Function Neural Networks (RBFNN), which represent deep and shallow architectural neural networks, is discussed in [7]. The paper's simulation findings compared incremental training with batch training techniques. The trials revealed that the incremental training method is faster than the batch training option. The CNN's performance is evaluated using the Malaysian traffic sign database, achieving a recognition rate of 99%.

[8] presents an innovative data-driven approach for identifying various traffic signals in video sequences acquired by a car-mounted camera. This approach takes into account both text-based and symbol-based signs. Post-processing is the third stage of the system, which is structured as follows: the first stage is the extraction of traffic sign areas of interest (ROIs), followed by the refinement and categorization of ROIs. After receiving training with a substantial amount of data, the multi-task CNN can refine and classify data into specific categories. This training comprises synthetic traffic signs and photographs from street views that have been annotated. To arrive at a recognition conclusion, post-processing brings together the input from all of the frames.

A CNN methodology was presented as a potential method for a traffic sign identification system in the research conducted by [9]. The study compares and contrasts several different CNN designs throughout its duration. A graphics processing unit (GPU) integrated into a mobile device is used to carry out the procedure in real-time. The computer vision system that was built was shown to be exceptionally effective, as demonstrated by the results of the tests. The research carried out by [10] provides an overview of the development of a sophisticated traffic sign identification system that is intended to recognize and categorize traffic signs in high-quality photographs in an effective manner. Three components make up the framework. These are the region-of-interest (ROI) extraction method, the split-flow cascade tree detector, and a rapid occlusion-robust traffic sign categorization strategy. In contrast to earlier methods, the HCRE ROI extraction method focuses on extracting ROI with strong local contrast. This approach yields a balanced detection and extraction rate, which is a significant improvement over the prior methods.

An ELM classifier with deep convolutional features is used in the [11] study, which combines the excellent discriminative power of deep convolutional features learned by CNN with the ELM classifier's superior generalization performance. In this method the CNN first learns deep and robust features, then removes the fully-connected layers, transforming CNN into a feature extractor. The classifier is then fed with CNN features to provide an outstanding classification. This experiment shows that the suggested technique can compete with state-of-the-art algorithms while being far less complicated.

Recent advancements in deep learning have significantly impacted transportation, prompting a comprehensive examination of traffic sign classification methodologies, such as [12] introduces ConvNeSe, a lightweight convolutional neural network model for automated traffic sign recognition, utilizing Depthwise Separable Convolution and Inverted Residuals for feature extraction, along with Squeeze and Excitation Blocks for enhanced feature attention. Achieving 99.85% accuracy on the GTSRB, the model demonstrates robustness through ablation experiments. Likewise, [13] evaluates the effectiveness of various approaches, including traditional computer vision methods and deep learning techniques, for traffic sign recognition. Using models like EfficientNetB7, VGG19, ResNet50, and CNN, on the GTSRB dataset, accuracy rates of 98.24, 98.95, 99.28, and 99.25%, respectively are achieved, aiding in identifying the strengths and weaknesses of each method.

Researchers in [14] introduce a modified YOLOv4-based model with CSPDarknet53 backbone, incorporating data preprocessing and nighttime image enhancement techniques. The model, featuring improved PANet and K-Means clustering for anchor box calculation, achieves 94.80 and 80.71% accuracy on TT-100K and MTSD datasets, respectively, surpassing existing methods, with cross-data experiments

yielding 91.74 and 63.64% accuracy on GTSRB and ITSD datasets. Another study [15] presents a CNN for classifying 43 traffic signs from the GTSRB dataset, achieving 99.20% accuracy with 0.8 M parameters. Additionally, Faster R-CNN and YOLOv4 networks are implemented for traffic sign recognition, with YOLOv4 outperforming at 59.88% mean average precision (mAP) at 35 FPS, making it preferable for real-time applications. A comparative analysis of these models is provided, showcasing their respective strengths and limitations.

Authors in [16] presented a unified deep learning model for traffic sign detection and recognition, DLHR-TSDR, achieving 99.01% accuracy on the CURE-TSD dataset, surpassing traditional approaches. Researcher in [17] introduces two optimized ResNet models, ResNet V1 and ResNet V2, for automatic traffic sign recognition utilizing the ArTS dataset, alongside a novel dataset specifically curated for Arabic Traffic Sign recognition. The optimized ResNet V1 model attains superior training and validation accuracies of 99.18 and 96.14%, respectively, addressing overfitting and underfitting concerns. Notably, these results surpass existing methods in the literature, demonstrating the efficacy of the proposed models on similar-sized datasets. Leveraging CNN in image processing, the existing challenges in traffic sign recognition, achieving improved accuracy and reduced processing time under diverse environmental conditions are discussed in [18].

As the social economy progresses, the road traffic system, fundamental to the national economy, faces escalating safety concerns, notably driving violations like disobeying traffic signs, a major contributor to accidents. Rapid advancements in autonomous vehicle technology face challenges in automatic sign recognition, particularly in complex illumination environments like low light and haze. Conventional methods struggle to simultaneously perform haze removal and traffic sign detection and recognition, leading to reduced performance. Most of the work mentioned above exploited one dataset, models trained on specific datasets may not generalize well to new, unseen conditions or types of traffic signs. For example, a model trained on signs from one country may not perform as well when applied to signs from another country with different designs or standards. Moreover, these models may have difficulty detecting uncommon or rare traffic signs that are not well-represented in the training data. This can pose a safety risk if drivers encounter these signs on the road but the system fails to recognize them accurately. Most of these machine learning algorithms used for traffic sign recognition require significant computational resources for training and inference due to deep CNN-based networks, limiting their applicability in resource-constrained environments or real-time applications. In literature, extensive research exists in English traffic sign recognition, but limited work has been done for Turkish recognition, with potential applications in indoor signage and smart cities.

Contrarily, this study implemented the transfer learning concept by building a CNN model from scratch, training it with the GTSRB [19] dataset (Fig. 1 depicts a few images of the dataset), and then using it as a pre-trained model to fine-tune it with a different number of classes using two datasets, GTSRB and TSIT [20], thus fed our model with extra parameters to get a fast and more accurate result in less time. We built a Graphical User Interface (GUI) for this system

Fig. 1 The sample images of the GTSRB dataset

to classify the traffic signs graphically. Also, we compared our approach of building a traffic sign classifier with other papers, between the pre-trained model and fine-tuned model.

For this study, we exploited transfer learning [21] which is a research problem in machine learning and a common approach in computer vision since it allows us to create accurate models in a short amount of time. Transfer learning does not start from the beginning but rather utilizes patterns acquired from solving a new problem. By doing this, we can expand on current knowledge instead of starting from scratch. Transfer learning in computer vision is commonly demonstrated by utilizing pre-trained models.

This study exploits the deep neural network model as a pre-trained model for fine-tuning as a new model using the transfer learning approach, which can categorize signals in an image into distinct categories. With this model, we read and comprehend traffic signs, which is a critical duty for all autonomous cars. After model training and fine-tuning it, the algorithm showed the final results of more than 99% accuracy in traffic sign recognition. The dataset GTSRB was used for training and testing the base model, and also, we added 5 classes of the TSIT dataset for implementing the transfer learning with the new data. As the final step, we created a GUI for the system to show our classified signs' class names, and we compared the accuracy and processing time of the base model and fine-tuned model. The contributions of this study are summarized below:

- Presented an efficient, lightweight CNN-based transfer learning model.
- Exploited datasets of two different countries (Germany and Turkey) for better generalization of the model.
- Leveraged pre-existing knowledge to achieve valid results by training the base model with GTSRB and then fine-tuning it with a different number of classes using both GTSRB and TSIT datasets.
- Incorporated an additional 5 classes from the TSIT dataset for implementing transfer learning with new data.
- Exploited transfer learning approach to create an accurate model in a short time. Thus, by leveraging pre-trained models and fine-tuning them with new data, the study expanded on current knowledge rather than starting from scratch.
- Built a GUI for a traffic sign classification system, enabling graphical representation and classification of traffic signs.
- The presented approach demonstrated final results of more than 99% accuracy in traffic sign recognition

The rest of the paper is classified as Sect."Overview, Methods, and Tools" presents the method and tools overview, while Sect."Architecture, Data, and Model" outlines the data description and architecture of the proposed scheme, Sect."Experiments, Analysis, and Performance" illustrates the results, performance, and comparison with prior studies based on accuracy, performance, and methods. Lastly, Sect."Conclusions" concludes the study.

Overview, Methods, and Tools

This research aims to create a system that receives a traffic sign image as input, processes it, trains a CNN-based model, identifies the class of the input image, and produces the expected class or label. The complete system is divided into three major sections: Creating and training of CNN model with training dataset, using this model as a pre-trained model for a new model, fine-tuning the model and testing the model with test data, and previewing this process in a GUI. The training part of this work involves: loading the dataset, pre-processing images, encoding the labels, training of CNN model, and saving it into a file for testing purposes. The testing part involves some extra minor pre-processing of the input image, but it does not include any training in CNN. We used the deep learning framework TensorFlow [22] to address the task of identifying traffic signs. The GTSRB dataset was utilized to train and test the first model, while 5 more classes from the TSIT dataset were incorporated for training the updated model. The approach described can classify the 48 most prevalent forms of traffic signs.

The base model that is suggested in this study is a sequential model that is made up of three convolutional layers with different filter values. Each of these levels is followed by a max-pooling layer, which is designed to reduce the spatial dimensions of the convolutional layers. After that, we constructed a Dropout layer for regularization and the prevention of overfitting, as well as a Flatten layer, which was designed to flatten or unroll the three-dimensional (3D) output into a one-dimensional (1D) format. To generate the output layer, we first created a dense layer with a softmax activation function, and then we connected it to 43 different classes. Both the Adam optimizer and the categorical cross-entropy loss function were applied in the process of compiling the model for the classification of signs into several different categories. Figure 2 depicts the whole procedure of training a CNN model.

For implementing the transfer learning and fine-tuning the new model we clone the first model layers except for the last (Dense) layer to a new Sequential model. The pre-trained model layers must be frozen before compiling and training the new model because freezing prevents the weights in a given layer from being updated during training. However, we created another dense layer having 48 classes and we compiled the model then we unfreeze all the layers to fine-tune our model and set a learning rate to a small value (0.0001) to avoid wrecking the fine-tuned weights. Figure 3

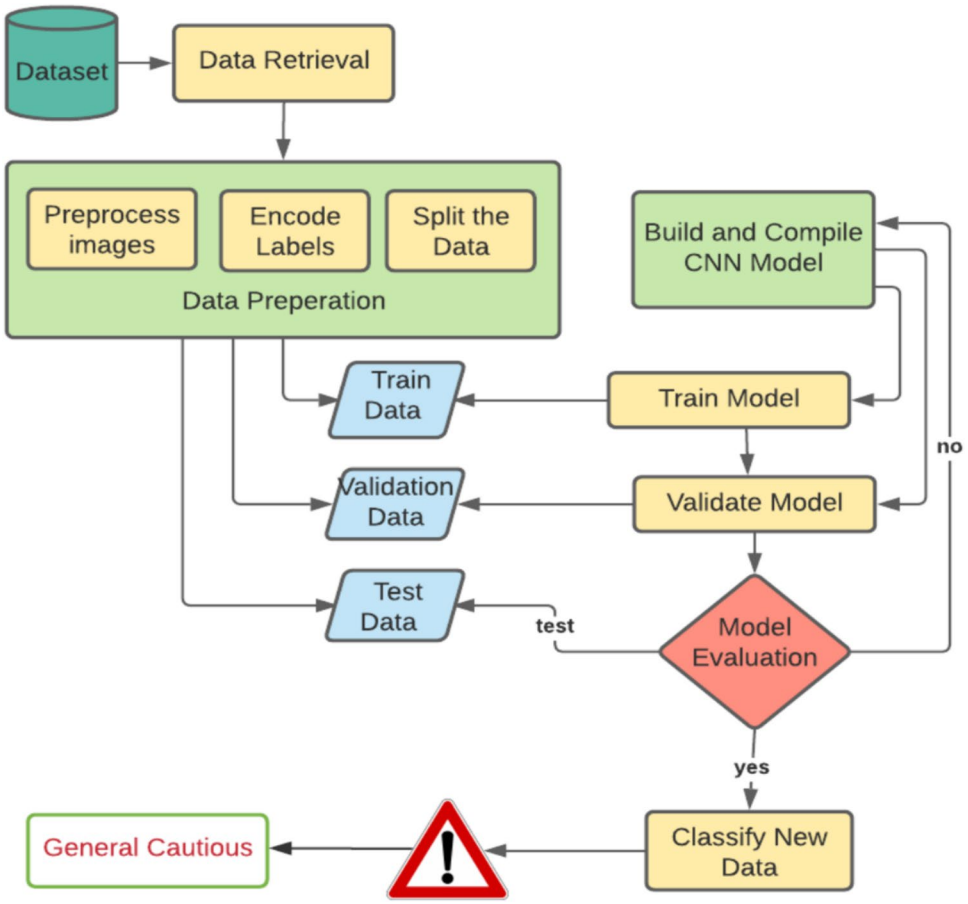


Fig. 2 The proposed workflow for traffic sign classification

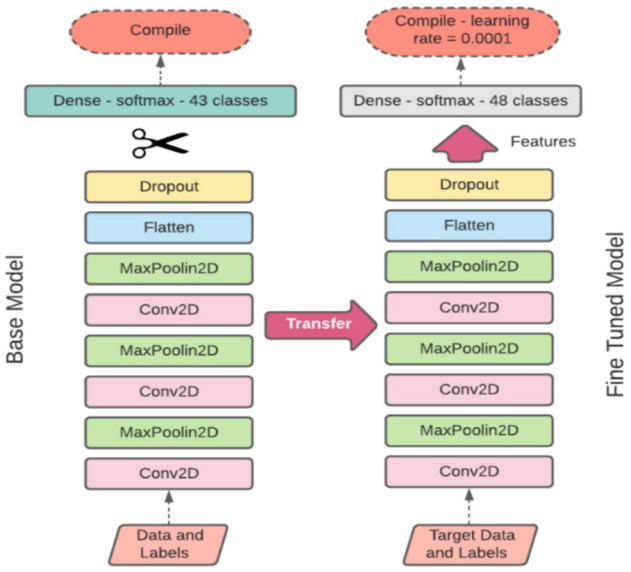


Fig. 3 Transfer-learning with pre-trained CNN model

shows the complete workflow of transfer learning in this work and how it is used from the pre-trained model.

The investigation is carried out using Python version 3, which is installed on a Windows 10 computer, and PyCharm is utilized for coding. JetBrains, a Czech business that was established in 2010 [23]. is responsible for the development of PyCharm, which is an Integrated Development Environment (IDE) created specifically for the Python programming language. A unit tester, code analysis tools, an interface with version control systems, a graphical debugger, and support for web development using Django and data research using Anaconda are all included in the software's comprehensive package, which is provided by the software [24].

We utilized the Tkinter Python Programming interface to create the system's graphical user interface. Tkinter is Python's standard GUI library. When Python is paired with Tkinter, it provides a quick and straightforward approach to constructing GUI applications. Tkinter extends the Tk GUI toolkit with a robust object-oriented interface. Figure 4 depicts the GUI created for this system.

Fig. 4 The user interface of the traffic sign recognition system

Architecture, Data, and Model

In this part, we will go into detail about the architecture of our used neural networks, a description of the data that is used for this project, and the training of our model.

Deep Neural Network Architecture

According to [25] there are three primary categories of DNN architectures: CNNs, unsupervised pre-trained networks (UPNs), and recurrent neural networks (RNNs). It is possible to gain an understanding of how these networks can be applied in practice by reading the architectural overview. In the field of image modeling and classification, CNNs are among the many different architectures of DNN.

CNNs are distinguished by the existence of convolution layers in addition to neurons. These layers considerably minimize the number of parameters that need to be learned in comparison to fully connected networks. CNNs are dependent on the process of convolution, which is carried out in such a way that the input that is provided to the CNN is represented as matrices or grids. CNNs are almost exclusively utilized for image identification, which has resulted in a significant advancement in the field of deep learning among its applications. Some examples of these technologies include but are not limited to, natural language processing, autonomous vehicles, assistance for people with vision impairments, and robotics [2].

Classifying with CNNs is the most advanced pattern recognition approach in computer vision today. CNN processes a two-dimensional image using convolutional layers, in contrast to normal neural networks, which deal with feature vectors that are only one dimension. To generate an activation map, each convolutional layer is composed of a collection of trainable filters. These filters calculate dot productions in the space between the filters and the layer input. These filters, which are often referred to as kernels, make it possible to identify the same properties in many different regions. As an illustration, Fig. 5 illustrates the results that were obtained by applying convolution to an image that contained four kernels [9].

Data Analysation

We utilize the GTSRB [19] and TSIT [20] datasets for this research, as described in the introduction. The GTSRB is a

Fig. 5 The Input image with 4 kernels of convolution

multi-class, single-image classification challenge that took place during the IJCNN (International Joint Conference on Neural Networks) in 2011. The benchmark dataset includes almost 50,000 images of various traffic signals. It is further subdivided into 43 distinct classes. The dataset is highly varied; some classes contain many images, while others have few images. The dataset is approximately 300 MB in size. While the TSIT dataset has total images of 21249 with 90 classes we will use some of its classes combined with the GTSRB dataset for fine-tuning the pre-trained model. Table 1 lists the details of the datasets and train-test split.

The images in the GTSRB dataset are in an old Portable Pixmap Format (PPM), and most of the tools don't support it, which meant that it was not possible to casually browse the folders to take a look at the images. We read the images using cv2 and Scikit Image library which recognize this format. Figure 6 shows the resized images and the labels of the dataset after reading and plotting them. Seems like it is great! The traffic signs occupy most of the area of each image, which is going to make our job easier. And we have a variety of angles and lighting conditions, which will help our model generalize. However, although the images were of different sizes and ratios. So as our neural network takes a fixed-size input, that's why we needed a little pre-processing to resize the images to a fixed size (30 x 30).

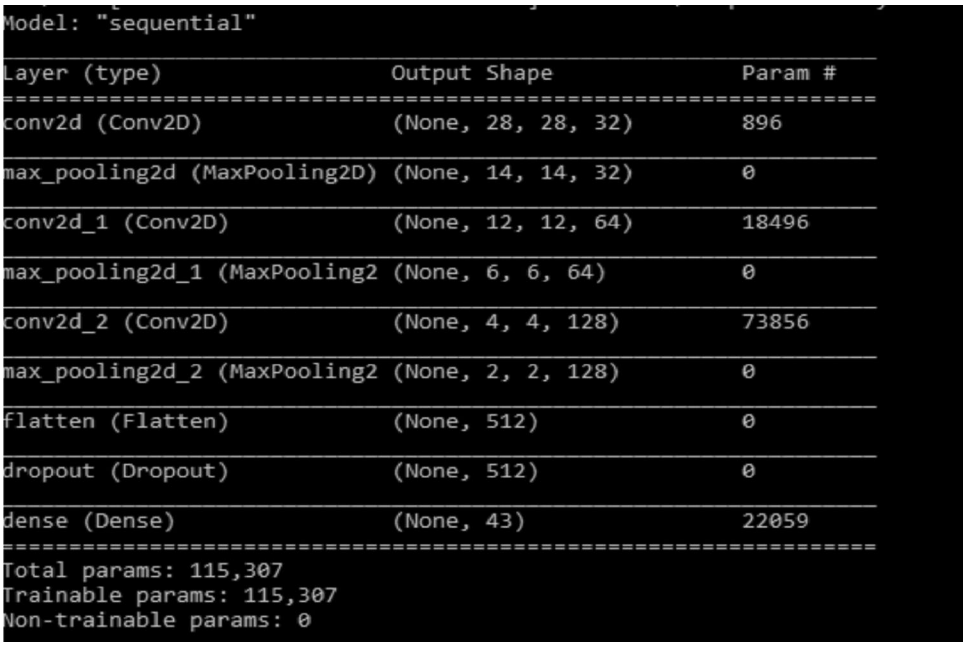
Table 1 The dataset description and train-test split ratio

GTSRB dataset	50,000 Images
TSIT dataset	21,249
Train set	60%
Test set	40%

Model Training

We developed a CNN model to classify the photographs into the categories that are most relevant to them. We exploited different machine learning and CNN-based models, however, Fig. 7 is a representation of the network design that gives the best result for traffic sign recognition. The structure

Fig. 6 Sample image from each category of the GTSRB dataset



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 12, 12, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 128)         73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 2, 2, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 43)                22059
=================================================================
Total params: 115,307
Trainable params: 115,307
Non-trainable params: 0
```

Fig. 7 The proposed convolutional neural network topology

is composed of several convolutional layers, pooling layers, fully linked layers, and one softmax layer. For each Conv2D and MaxPooling2D layer, the result is a three-dimensional form tensor that includes height, width, and channels. There is a gradual decrease in the width and height measures as we move deeper into the network. It is possible to change the number of output channels for each Conv2D layer by using the first option, which can be either 32 or 64 filters. After that, the max-pooling layer is utilized to reduce the spatial dimensions of the output volume as much as possible. In general, as the width and height decrease, we can afford to add more output channels in each Conv2D layer. The softmax layer is responsible for generalizing the output of the layer that came before it. This layer is responsible for containing the likelihood of belonging to recognized classes for the real input image.

To complete the model, we input the final output tensor from the convolutional base, which has the shape (2, 2, 128), into a single Dense layer for classification. The dense layer takes in vectors in one dimension as input and generates a tensor in three dimensions as output. Before adding one Dropout layer for regularization, we first flatten the three-dimensional output to one-dimensional space to prevent overfitting. Outputs of (2, 2, 128) were transformed into vectors of the form (512) before their passage through the Dense layer. In conclusion, since GTSRB possesses 43 output classes, we utilize a final thick layer that also possesses 43 outputs and softmax activation.

To train and validate the fundamental model, the initial dataset was divided into train and test sets with a ratio of sixty percent to forty percent. After the training was completed successfully, the accuracy was determined in each iteration by using all of the photos extracted from the test dataset. Figure 8 illustrates that the classification accuracy and loss both rise in proportion to the number of training iterations that are performed.

During the training of the model, the network processed all of the images from the training dataset in a single iteration. Additionally, we validated the model by using a validation set and included three callbacks, which are EarlyStopping, ModelCheckpoint, and TensorBoard callbacks. During the data training process, the EarlyStopping callback is utilized to prevent overfitting from occurring. If the monitored metric does not show any signs of improvement, the training will be terminated using this callback. Since this is the case, the loss value would be the metric that needs to be watched. Upon the conclusion of each epoch, the training loop will do a check to determine whether or not the loss is no longer decreasing. If it is, the stop training condition will be marked as true, and the training will be complete. We utilized the ModelCheckpoint function to save the best model or weights in a checkpoint file at regular intervals. This allowed the model or weights to be loaded at a later time, allowing us to resume training from the stored state while utilizing the best model. Logging TensorBoard events, which are necessary for TensorBoard visualizations, is accomplished through the use of the TensorBoard callback. TensorBoard is a visualization tool that is included as part of the TensorFlow package automatically.

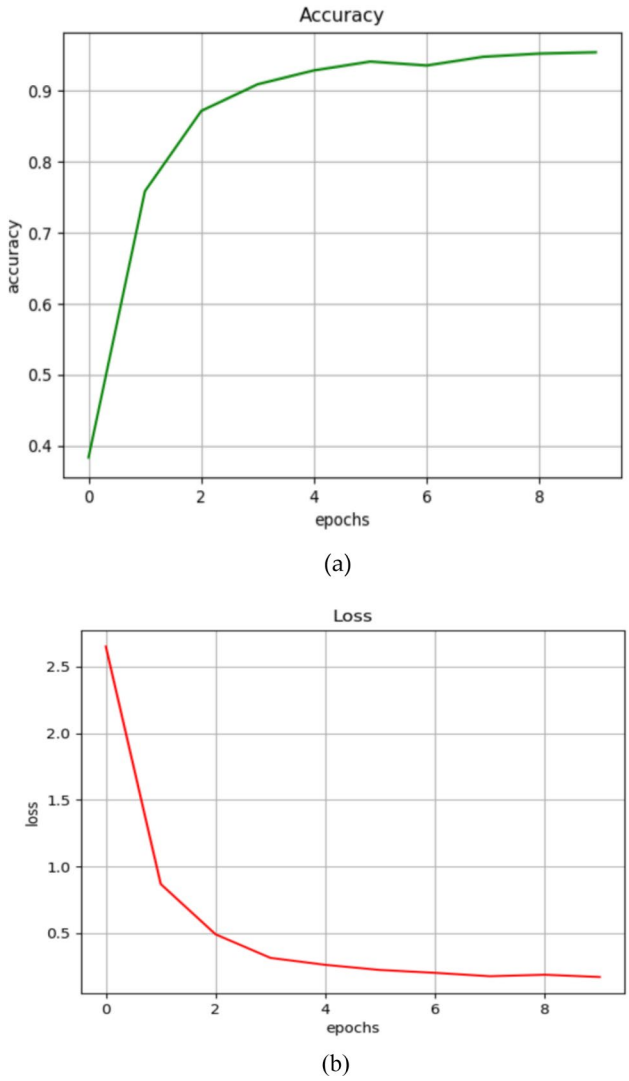


Fig. 8 Proposed model performance plot along each iteration; **a** accuracy curve, **b** loss curve

Experiments, Analysis, and Performance

Since this study is centered on a traffic recognition system, it is necessary to evaluate the performance of both processing and classification. The authors in [26] presents a demonstration of an effective implementation of sign recognition that takes 0.43 seconds and makes use of Color Segmentation, Shape Matching, and Support Vector Machines. Both the Intel(R) Core(TM) i7 CPU 2.70GHz and the Intel(R) Core(TM) i7 CPU 2.90GHz computers were utilized to determine the total amount of time required for the processing of this work. During the training phase of the CNN model, we implemented batch gradient descent by including a batch size. This was done to reduce the amount of time required for processing and to monitor the generalization error. The EarlyStopping callback was implemented so that it would immediately stop when there was no improvement in the monitored metric. We made use of transfer learning, which resulted in a significant reduction in the amount of time to the other two models and publications. Table 2 shows the overall processing time of CNN model training and classification before and after using EarlyStopping (ES) callback and gradient descent (GD).

When evaluating the accuracy of the base model, we made use of the dataset that was provided by the GTSRB. Over fifty thousand photographs of traffic signs taken in a variety of settings are included in this collection. Following the evaluation of the model, we utilized it as a pre-trained model and utilized transfer learning to fine-tune a new model that was based on the base model. Additionally, we utilized some techniques such as adding a dropout layer with an experimental value of 0.4, which had a significant impact in increasing the accuracy of the model and adding more than one convolutional layer with a Relu activation function, which resulted in a 99.44% increase in accuracy. Figure 9 shows the TensorBoard visualization for model accuracy and loss of both training and test stages in each iteration using transfer learning. As we can see using transfer learning during the training stage the accuracy is jumping up so rapidly in each iteration and it starts from a high point and continues to increase, the same concept with loss that is decreasing at a rapid rate.

Table 2 displays the generated accuracy and performance of the proposed method and methods described in the studies [26] and [27]. These papers compare the performance of humans in traffic sign recognition to that of state-of-the-art machine learning algorithms and the fine-tuned model that is derived after taking transfer learning into consideration. To determine the accuracy of each of the methodologies

Table 2 Model training and classifying execution time for models with and without EarlyStopping (ES) callback, gradient descent (GD), and transferring learning (TL)

Hardware	Training Time (s)		Classifying Time (s)		Training time with TL (s)	Classifying time with TL (s)
	ES, GD	No ES, GD	ES, GD	No ES GD		
Inter(R) Core(TM) i7 CPU 2.90 GHz, 8 GB RAM	50	65	0.01	0.03	30	0.002
Inter(R) Core(TM) i7 CPU 2.70 GHz, 8 GB RAM	60	80	0.03	0.05	40	0.007

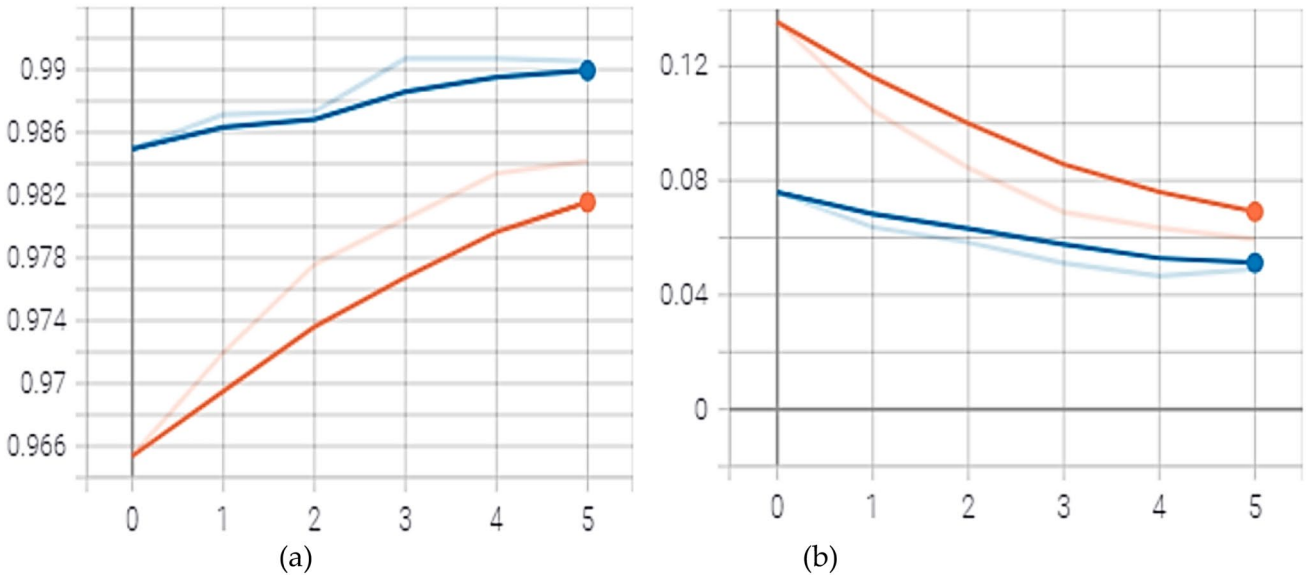


Fig. 9 TensorBoard visualization of the fine-tuned model; **a** accuracy **b** loss curves against epochs

presented in the table, the GTSRB dataset was utilized; however, the research conducted by [26] was limited to the detection and identification of red traffic signals.

As demonstrated in Table 3, one of the most efficient ways for Traffic Sign Recognition using the GTSRB dataset is to use a CNN pre-trained model with transfer learning to categorize traffic signs. The strategy suggested in this research outperforms [26], and [27] in terms of performance and accuracy. Images of traffic signs detected effectively by the suggested approach in this study are displayed in Fig. 10, thus it is evident that the proposed system is effective and accurate even when a low-resolution, blurred, minutely bright, or low-contrast image is fed. The images indicate that the applied approach produces strong classification results even with images of traffic signs that are difficult for humans to recognize.

Conclusions

This study delves into Traffic Sign Recognition (TSR), examining its research status, application domains, and database enhancements. During the course of this paper, we presented a model for traffic sign identification that utilized a convolutional neural network as well as the notion of transfer learning. To tweak the model, the study assessed the network under a variety of different parameters. It has been discovered that the utilization of transfer learning, the ReLU activation function in multiple convolutional layers, the appropriate drop-out value, the utilization of EarlyStopping callback and gradient descent, and the softmax function in the output layer significantly improved the overall result of the classification process. Additionally, it affected the performance of training time. The categorization is carried out on the dataset consisting of German traffic signs. Transfer learning allowed us to reduce the amount of time spent on performance and loss while simultaneously increasing the accuracy of the model. Specifically, the proposed fine-tuned

Table 3 Comparison of the proposed method and several existing methods

Work	Algorithm	Accuracy (%)	Processing time
Fine tuned model	CNN-transfer learning	99.44	30 s
Base model	CNN	97.2	50 s
[26]	SVM	95.71	43 s
[27]	CNNs	99.46	–
	Human (best individual)	99.22	–
	LDA (HOG 2)	95.68	–

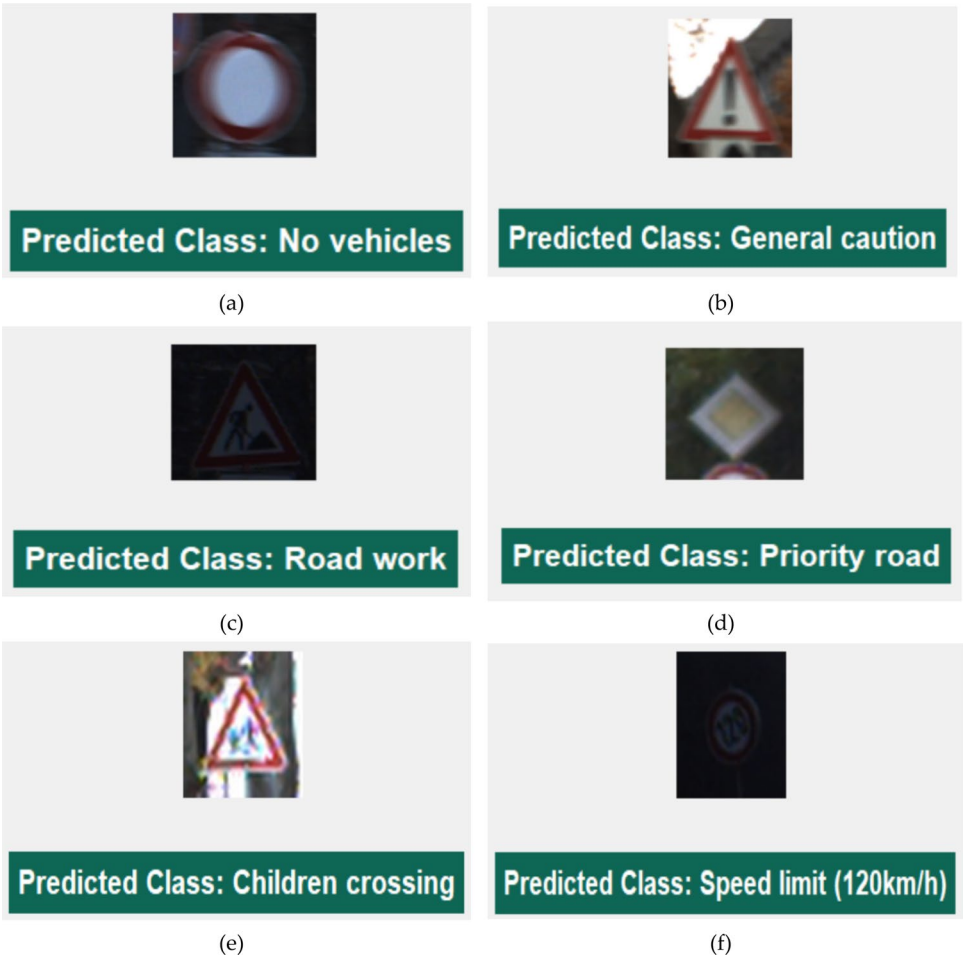


Fig. 10 Sample test images (each of varying brightness, contrast, and blurriness) of unseen data along with predicted class; **a** image with no vehicle, **b** blurred image having general caution sign, **c** visually blacked image having road work sign, **d** low bright, blurred and low-resolution image having priority road sign, **e** low-resolution image having children crossing sign, **f** minute bright image having speed limit sign

transfer-learning model enhanced the performance from 0.01s to 0.002s (classifying time), 50s to 30s (training time), and 97 to 99.44% (accuracy). The experiment primarily focused on the German traffic sign and Turkish traffic sign datasets. Future research could evaluate the model's performance on datasets from different countries to assess its generalization capabilities and robustness across diverse traffic sign designs and regulations. Moreover, the experiment utilized dropout regularization to prevent overfitting. Future research could explore other regularization techniques such as L1 and L2 regularization, as well as data augmentation methods, to enhance the model's generalization ability and robustness to variations in input data.

Funding This research received no external funding.

Data Availability The data that support the findings of this study are publicly available [19, 20].

Declarations

Conflict of Interest The authors declare no conflict of interest.

References

1. World Health Administration, Global Plan for the Decade of Action for Road Safety 2021–30. 2018. Available: https://www.who.int/publications/m/item/global-plan-for-the-decade-of-action-for-road-safety-2021-2030.
2. Abid F, Rasheed J, Hamdi M, Alshahrani H, Al Reshan MS, Shaikh A. Sentiment analysis in social internet of things using contextual representations and dilated convolution neural network. Neural Comput Appl. 2024. https://doi.org/10.1007/s00521-024-09771-2.
3. Albawi S, Mohammed TA, Al-Zawi S. Understanding of a convolutional neural network. In: 2017 International Conference on Engineering and Technology (ICET). 2017. pp 1–6.
4. Waziry S, Wardak AB, Rasheed J, Shubair RM, Rajab K, Shaikh A. Performance comparison of machine learning driven approaches for classification of complex noises in quick response code images. Heliyon. 2023;9(4):e15108. https://doi.org/10.1016/j.heliyon.2023.e15108.
5. Farooq MS, et al. A conceptual multi-layer framework for the detection of nighttime pedestrian in autonomous vehicles using deep reinforcement learning. Entropy. 2023;25(1):135. https://doi.org/10.3390/e25010135.
6. Boujemaa KS, Bouhoute A, Boubouh K, Berrada I (2017) Traffic sign recognition using convolutional neural networks. Proceedings—2017 International Conference on Wireless Networks and Mobile Communications, WINCOM 2017; 2017. p. 1–12.

7. MM Lau, KH Lim, AA Gopalai. Malaysia traffic sign recognition with convolutional neural network. In: International Conference on Digital Signal Processing, DSP, vol. 2015-Septe. 2015. p. 1006–10.
8. Fuchao W, Bin F, Luo H, Yang Y, Tong B. Traffic sign recognition using a multi-task convolutional neural network. IEEE Trans Intell Transp Syst. 2018;19(4):1100–11. https://doi.org/10.1109/TITS.2017.2714691.
9. Shustanov A, Yakimov P. CNN design for real-time traffic sign recognition. Procedia Eng. 2017;201:718–25. https://doi.org/10.1016/j.proeng.2017.09.594.
10. Liu C, Chang F, Chen Z, Liu D. Fast traffic sign recognition via high-contrast region extraction and extended sparse representation. IEEE Trans Intell Transp Syst. 2016;17(1):79–92. https://doi.org/10.1109/TITS.2015.2459594.
11. Zeng Y, Xu X, Fang Y, Zhao K. Traffic sign recognition using deep convolutional networks and extreme learning machine. In: He X, Gao X, Zhang Y, Zhou ZH, Liu ZY, Fu B, Hu F, Zhang Z, editors. Intelligence Science and Big Data Engineering. Image and Video Data Engineering, Image Proc., vol. 9242. Springer; 2015. p. 272–80.
12. Wei W, et al. A lightweight network for traffic sign recognition based on multi-scale feature and attention mechanism. Heliyon. 2024;10(4):e26182. https://doi.org/10.1016/j.heliyon.2024.e26182.
13. Hamza A, Nawal S. Traffic sign classification using deep learning comparative study. Procedia Comput Sci. 2024;233:939–49. https://doi.org/10.1016/j.procs.2024.03.283.
14. Saxena S, Dey S, Shah M, Gupta S. Traffic sign detection in unconstrained environment using improved YOLOv4. Expert Syst Appl. 2024;238:121836. https://doi.org/10.1016/j.eswa.2023.121836.
15. Youssouf N. Traffic sign classification using CNN and detection using faster-RCNN and YOLOV4. Heliyon. 2022;8(12):e11792. https://doi.org/10.1016/j.heliyon.2022.e11792.
16. Rani AR, Anusha Y, Cherishama SK, Laxmi SV. Traffic sign detection and recognition using deep learning-based approach with haze removal for autonomous vehicle navigation. E-Prime Adv Electr Eng Electr Energy. 2024;7:100442. https://doi.org/10.1016/j.prime.2024.100442.
17. Latif G, Alghmgham DA, Maheswar R, Alghazo J, Sibai F, Aly MH. Deep learning in Transportation: Optimized driven deep residual networks for Arabic traffic sign recognition. Alex Eng J. 2023;80:134–43. https://doi.org/10.1016/j.aej.2023.08.047.
18. Qiao X. Research on traffic sign recognition based on CNN deep learning network. Procedia Comput Sci. 2023;228:826–37. https://doi.org/10.1016/j.procs.2023.11.102.
19. Für Neuroinformatik I. GTSRB—German Traffic Sign Recognition Benchmark | Kaggle. 2019. https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign. Accessed 05 Jan 2024.
20. Yucesan E. Traffic Sign Images From Turkey. 2020. https://www.kaggle.com/datasets/erdicem/traffic-sign-images-from-turkey. Accessed 10 Jan 2024.
21. Lin C, Li L, Luo W, Wang KCP, Guo J. Transfer learning based traffic sign recognition using inception-v3 model. Period Polytech Transp Eng. 2019;47(3):242–50. https://doi.org/10.3311/PPtr.11480.
22. Filus K, Domańska J. Software vulnerabilities in TensorFlow-based deep learning applications. Comput Secur. 2023;124:102948. https://doi.org/10.1016/j.cose.2022.102948.
23. Huang SC, Le TH. Environment installation. In: Principles and Labs for Deep Learning. Elsevier; 2021. p. ix–xxv.
24. Haagsman E. Collaboration with Anaconda, Inc. PyCharm Blog. 2019. https://blog.jetbrains.com/pycharm/2019/04/collaboration-with-anaconda-inc/. Accessed 01 Feb 2023.
25. Wang L, Wang X, Hawbani A, Xiong Y, Zhang X. An analysis of deep neural network models for image recognition applications. J Intell Fuzzy Syst. 2021. https://doi.org/10.3233/jifs-219081.
26. Wali SB, Hannan MA, Hussain A, Samad SA. An automatic traffic sign detection and recognition system based on colour segmentation shape matching, and SVM. Math Probl Eng. 2015. https://doi.org/10.1155/2015/250461.
27. Stallkamp J, Schlipsing M, Salmen J, Igel C. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. Neural Netw. 2012;32:323–32. https://doi.org/10.1016/j.neunet.2012.02.016.