Int J Syst Assur Eng Manag
https://doi.org/10.1007/s13198-021-01127-6

ORIGINAL ARTICLE

# Deep learning using computer vision in self driving cars for lane and traffic sign detection

**Nitin Kanagaraj[1] · David Hicks[1] · Ayush Goyal[1] · Sanju Tiwari[2] · Ghanapriya Singh[3]**

Received: 11 November 2020 / Revised: 8 April 2021 / Accepted: 26 April 2021


**Abstract** Recently, the amount of research in the field of self-driving cars has grown significantly with autonomous vehicles having clocked in more than 10 million miles, providing a substantial amount of data for use in training and testing. The most complex part of training is the use of computer vision for feature extraction and object detection in real-time. Much relevant research has been done on improving the algorithms in the area of image segmentation. The proposed idea presents the use of Convoluted Neural Networks using Spatial Transformer Networks and lane detection in real time to increase the efficiency of autonomous vehicles. The depth of the neural network will help in training vehicles and during the testing phase, the vehicles will learn to make decisions based on the training data. In case of sudden changes to the environment, the vehicle will be able to make decisions quickly to prevent damage or danger to lives. Along with lane detection, a self-driving car must also be able to detect traffic signs. The proposed approach uses the Adam Optimizer which runs on top of the LeNet-5 architecture. The LeNet-5 architecture is analyzed and compared with the Feed Forward Neural Network approach. The accuracy of the LeNet-5 architecture was found to be 97% while the accuracy of the Feed Forward Neural Network was 94%.

**Keywords** Computer vision · Deep learning · Self-driving cars · Autonomous vehicles

## 1 Introduction

In 2017 alone, over 40,000 people died in the United States due to car accidents. Across the globe, the number increases to more than a million people. Most of the accidents could have been avoided if the drivers had paid attention to their surroundings. A number of automobile brands and autonomous vehicle companies are investing billions in self-driving technology. Examples include tech heavyweights such Tesla, Google's Waymo, Uber, and Apple. Traditional car companies including Audi, BMW, Ford, and Volvo have also shown interest in self-driving technology. By the year 2025, self-driving cars are expected to comprise 20% of the total number of cars sold in the United States. Though people might have concerns about letting a computer drive their car, self-driving technology can actually improve safety. When a human driver is tired and fails to notice or accurately read a sign or if a driver is speeding or fails to notice a pedestrian crossing the road unexpectedly the results can be fatal. Self-driving

✉ Sanju Tiwari
tiwarisanju18@ieee.org

Nitin Kanagaraj
nitin.kanagaraj@students.tamuk.edu

David Hicks
David.Hicks@tamuk.edu

Ayush Goyal
ayush.goyal@tamuk.edu

Ghanapriya Singh
ghanapriya@nituk.ac.in

[1] Department of Electrical Engineering and Computer Science, Texas A&M University-Kingsville, Kingsville, TX, USA

[2] Universidad Autonoma de Tamaulipas, Ciudad Victoria, Mexico

[3] National Institute of Technology Uttarakhand, Srinagar Garhwal, India

Published online: 14 May 2021

technology can mitigate such situations and improve safety on the road.

Various components are required to enable self-driving technology to function properly. These components include a long distance radar system, ultrasonic sensors, cameras paired with image recognition software, and real-time traffic data supported by satellite imagery. Long distance radar systems and ultrasonic sensors can be used to determine the distance between a vehicle and another vehicle or an obstacle. Image recognition software used in conjunction with cameras can enable the recognition of other vehicles, the recognition of pedestrians on the road, and the detection and interpretation of traffic signs. Real time traffic data can be used to determine the optimum route to be used to reach a destination.

There are different degrees of self-driving technology that can be present in a vehicle. Various stages in the self driving technology include no automation, driver assistance, partial automation, conditional automation, high automation, and full automation. A vehicle with no automation relies completely on input from a human at all times to guide vehicle. Driver assistance refers to techniques and technology that aid the human that is operating the vehicle, for example, lane assistance, reversing of car etc. Partial automation refers to an approach in which self-driving technology is present, but human assistance is still required approximately 50% of the time, especially when making turns. Conditional automation employs even more self-driving technology but still relies on a human driver to recognize and interpret traffic signs. High automation reduces the amount of human assistance required to 20%. Finally, full automation relies completely on self-driving technology and requires no human assistance in guiding a vehicle. Deep learning and machine learning techniques (Rahul et al. 2019; Gaurav et al. 2020; Mishra et al. 2018; Chatrati et al. 2020) are playing a significant role in feature extraction, object detection and data pre-processing.

This paper presents the results of research efforts targeted at facilitating improvements in two areas that are very important for self-driving vehicles: lane detection and traffic sign detection. In the paper, authors' contribution can be summarized as follows:

1. Convolution Neural Network with learnable module Spatial Transformer Networks (STN) is used to detect lane for autonomous self-driving cars.
2. For detection of traffic signal an CNN with LeNet-5 architecture is implemented having Adam optimizer.
3. To test the validity of the proposed algorithm, it was implemented on German traffic sign dataset.
4. For the comparison, the accuracy of the proposed methodology was compared to methodology based on feed forward network.

Rest of the paper is organized as follows. Section 2 examines the related literature in the areas of lane detection, obstacle detection, and computer vision for self-driving cars. The Sect. 3 of the paper describes the approaches designed to facilitate both lane detection and also traffic sign identification. Section 4 describes the German Traffic Sign dataset and its use in evaluating the approaches suggested. The results of the evaluation are described in Sect. 5. After a discussion of the approaches for lane detection and traffic sign identification in Sect. 6, the final Sect. 7 concludes the paper and also briefly discussed future research directions.

## 2 Related literature

The most important and relevant aspects of self-driving technology for the research project described in this paper are lane detection, vehicle and obstacle detection, and feature extraction (Rahul et al. 2019; Dwivedi et al. 2021) using cameras paired with computer vision software. This section examines recent research (Sant et al. 2021) going on in each of these key areas and how it applies to the research project described in this paper.

### 2.1 Lane detection for self-driving cars

In self-driving cars lane detection is used to assist keeping a car in a particular lane. It also plays an important role in moving a vehicle to an alternate lane. Aziz et al. (2017) have proposed a model for lane detection based on color region, line selection, edge selection, and the Hough transformation. The approach suggested combines the information obtained from computer vision and sensor fusion with path planning. Sensor fusion is a combination of real-time data from the multiple sensors that are attached to the car. Path planning is used as a way to determine the optimal path between two points. The initial step in the suggested approach is the capture of images. Next color selection is performed followed by the masking and edge selection process for detecting a lane. In color selection, every image point is examined for image features and masking is done for the selected image points using the Hough transform (Shapiro and Stockman 2001). In the Hough transform, a voting procedure is used to determine the shapes of objects in the image points. The algorithm works well when used in daylight, but fails to detect image points during the night since images are then darker, preventing lane detection.

The vehicles on a roadway often travel at different speeds. This speed difference can be used to help detect lanes. Kim et al. (2016) have proposed an approach for lane detection that uses the distance ratio between vehicles. The position of a vehicle is determined by comparing the image

frames that are captured by a camera. The region of interest (ROI) within the frames is determined since only those regions are to be processed for lane detection. The images are also converted into gray scale and edge detection is performed. The edges are detected by sharp changes in the brightness and discontinuities in the image. Next a speed adaptive algorithm is used in which a vehicle's distance is obtained using the vehicle speed and the camera frame speed. The formula upon which the speed adaptive algorithm is based is given in Eq. 1 below.

$$D_a = \frac{0.28V}{F} \tag{1}$$

In Eq. 1, $V$ is the current vehicle speed and $F$ is the frame rate per second of the camera. This approach might not be applicable for lane detection in all situations, but the information provided can be used to help develop alternative strategies.

Lane detection in a different environment such as under changing lighting conditions becomes very difficult. In this situation, relying on an algorithm based on a single input feature can be futile. Gupta et al. (2018) propose a multi feature lane detection algorithm that can be applicable in these conditions. It considers and processes multiple types of input including: gradient-based features, intensity-based features, and texture-based features. They found that edge detection doesn't work well in low lighting conditions and does not support lane detection. To overcome this difficulty they have incorporated the use of gradient-based features. A line segment detector (LSD) processes images captured by a camera to identify line segments. Geometric constraints such as the intersection point of line segments with each other are considered and a score is computed to rate the likelihood of a segment being part of a lane marker, enabling non lane marker segments to be removed.

The approach suggested in Gupta et al. (2018) can also utilize intensity-based feature information in the presence of shadows that might hinder edge detection in different illumination conditions. Lane markers have a different brightness level so they can be detected by a low–high–low pattern and obtained from the image. Although his can pose a perspective distortion problem, it can be resolved using a top down view of the road image. For roads with no man-made markings, like dirt roads, the approach can utilize information on texture-based features. A texture-based feature encoder-decoder architecture supports classification detection and semantic segmentation to enable lane detection for these conditions.

### 2.2 Obstacle detection for self-driving cars

Vehicle and obstacle detection with a high degree of confidence is very important for maneuvering and controlling a self-driving car. Ren et al. (2015) have proposed a deep learning system utilizing a region based convolutional neural network for detecting and classifying obstacles such as vehicles, pedestrians, and animals. Vehicle detection is complicated by differences in vehicle size, lighting conditions, and the surrounding environment. The approach suggested uses a variant of a convolutional neural network (CNN) known as a Faster Region based CNN (R-CNN) to detect and classify objects (Prabhakar et al. 2017). The R-CNN first takes in a raw input image and extracts features. It then uses sub-sampling in different layers of the neural network and also applies activation functions to classify objects.

In the Faster Region based CNN approach, precision is a measure of the accuracy of predictions that are made. The mean average precision (mAP) is defined as the average of the maximum precisions at different recall values. Also in the Faster Region based CNN approach, a regional proposal network (RPN) is used to separate images into different regions. The RPN analyzes the image and proposes regions with a high probability of containing an object of interest. The proposed regions can be marked with bounding boxes overlaying real time video. The bounding boxes are also passed to the next convolutional layer which can plan moves for a self-driving car. The model proposed was found to have a calculated mAP value of 97.42% for different obstacles. The model was used to estimate the trajectory of moving vehicles found in bounding boxes but began to fail to perform as the number of vehicles in a frame increased significantly.

It is very important for obstacle detection to be performed accurately regardless of lighting conditions. Renjith et al. (2017) have proposed a model in which obstacles can be detected using a feature matching image processing algorithm. The primary advantage of the feature matching algorithm is that it is independent of lighting conditions in the environment. The algorithm extracts features from an image and characterizes them into key points, used to identify different objects in the image. Next a feature matching phase compares features of the current frame to features of known objects. When the features match an object can be detected. Filtering is done to control for noise using David Lowe's ratio test (Lowe 2004). The process of finding objects in this way is known as homography. After processing, a filtered image can be drawn with a bounding polygon to represent the region of interest. The distances between objects can be calculated using a distance formula, given in Eq. 2 below.

$$D = \frac{2H}{0.026A} \tag{2}$$

In Eq. 2, $D$ is the distance of an object from the camera, $H$ is the height of an object, and $A$ is the computed height

of the object obtained from the camera. A threshold value is set for $D$, below which certain vehicle actions must take place. In this way, the model uses feature extraction (Rahul et al. 2019; Gaurav et al. 2020; Mishra et al. 2018) in image processing to detect objects, and then the distance between the vehicle and the detected objects provides information to inform vehicular decisions for different scenarios.

### 2.3 Computer vision for self-driving cars

Computer vision for self-driving cars is a complicated topic with several problems to be addressed including occlusion, inter-class variability, and pose variability. Occlusion refers to when part of an object is being blocked from view by another object. When this occurs it makes it more difficult to identify the object being occluded, although it is still important to do so. Inter-class variability refers to classification done when there is high variability within classes and little difference between the classes. For example, it is easier to differentiate between dogs and cats but difficult to identify specific types of dogs or cats. The pose variability problem is caused by the way in which an object's color and texture in a 2-dimensional plane will vary as the object is rotated.

To address the occlusion problem, Lowe (2004) has proposed a multi-level framework to detect and handle vehicle occlusion. In the proposed model, the compactness ratio and interior distance ratio of vehicles and a cutting region are used to remove occluded vehicles. A tracking level is used to keep track of occluded vehicle images and perform a bidirectional occlusion reasoning algorithm. In occlusion detection, the shape of a non-occluded vehicle is a convex one while the shape of a partially occluded vehicle is concave. Occlusion detection is performed using subtractive clustering (Zhang et al. 2008) in which cluster members are not required to be specified in advance. Intra-frame and inter-frame levels can be used for the case of partial occlusion but for severe occlusion a bidirectional occlusion reasoning algorithm is required (Singh et al. 2020). The proposed model was applied to the ImageNet dataset and it detected the occlusion present in the dataset correctly, proving the viability of the method.

YOLO (You Only Look Once) is one of the real time CNN methods suggested for detection of objects in images. Nugraha and Su (2017) have proposed a model for object detection based on the YOLO method. The model speeds up the computation time and was able to achieve a processing rate of 7 frames per second. YOLO is a deep CNN architecture containing 24 convolutional layers along with 2 fully connected layers. The model proposed in Nugraha and Su (2017) uses a smaller version of the original YOLO method. The input image is resized to $448 \times 448 \times 3$ pixels and sent to the CNN. For a detected object the model produces the x mid-point coordinate, the y mid-point coordinate, the width, and the height. In general the model gives accurate results for highway areas but fails to detect white lines and sand on the edge of the road.

Table 1 has presented a tabular structure to characterize different parameters of the existing work that is closely related with the proposed work.

## 3 Methodology

In the past few years, deep learning models especially Convolutional Neural Network (CNN) has shown tremendous potential in image processing area that includes image segmentation, image detection and recognition etc. The structure of a CNN is illustrated in Fig. 1. The neural network consists of several perceptrons which are used to minimize processing and increase the efficiency of the model prediction. Neural networks usually perform better than other image classification algorithms since they use little pre-processing power. The neural network is comprised of convolutional layers, pooling layers, fully connected layers and normalization layers.

The number of parameters in a neural network grows rapidly with the increase in the number of layers. This can make training for a model computationally heavy (and sometimes not feasible). Tuning so many of parameters can be a very huge task. The time taken for tuning these parameters is diminished by CNNs. CNNs are fully connected feed forward neural networks. CNNs are very effective in reducing the number of parameters without losing on the quality of models. Images have high dimensionality (as each pixel is considered as a feature) which suits the above described abilities of CNNs. Also, CNNs were developed keeping images into consideration but have achieved benchmarks in text processing too. CNNs are trained to identify the edges of objects in any image.

### 3.1 Proposed system for lane detection in self-driving cars

The image processing used for lane detection is done using computer vision techniques. The following describes an image processing pipeline and how it employs advanced computer vision concepts to perform lane detection to support self-driving cars.

#### *3.1.1 Image processing pipeline for lane detection*

A number of sensors are used to detect other vehicles and for lane detection in self-driving cars. They enable an input image to be captured in real time and fed into an image

**Table 1** Main features of existing literature

| Study | Objective | Used techniques | Datasets | Significant for Proposed Work |
|---|---|---|---|---|
| Aziz et al. (2017) | Lane detection algorithm by comparing color-based lane detection algorithms | Edge extraction and Gaussian filters and a Median filter | NA | Yes |
| Gupta et al. (2018) | A lane model using geometric constraints on lane shape and fit the lane model to the visual cues extracted | Classification and segmentation | KITTI dataset | Yes |
| Ren et al. (2015) | Introduce novel Region Proposal Networks (RPNs) that share convolutional layers with state-of-the-art object detection networks | CNN | MS COCO datasets | Yes |
| Prabhakar et al. (2017) | A DL system using region-based CNN trained with PASCAL VOC image dataset is developed for the detection and classification of on-road obstacles such as vehicles, pedestrians and animals | DL, CNN | KITTI dataset | Yes |
| Renjith et al. (2017) | Implementation of the core feature of a self-driving car which is traffic sign and obstacle detection | Feature matching Image processing algorithm | NA | Yes |
| Gadekallu et al. (2020) | Applying machine learning model for classifying tomato disease image dataset | Hybrid-principal component analysis | Plant–village dataset | Yes |
| Sucharitha et al. (2020) | Clustering by using fuzzy method and CNN for segmenting the brain into three tissue | Fuzzy-method and CNN | OASIS Database | Yes |
| Park (2018) | Implementation of lane detection algorithm on toll road Cipularang as parts of self-driving car system | Line selection, canny edge detection, and Hough transform | NA | Yes |
| Dorj et al. (2020) | Estimate parameters of the road turning and define geometric shapes | Image pre-processing | NA | Yes |

There are total 9 studies has been discussed to present the primary objectives, techniques, used datasets and relevancy with proposed work. It is observed that most of the studies have used CNN to implement their work. The proposed work also motivated with existing work to use the CNN approach

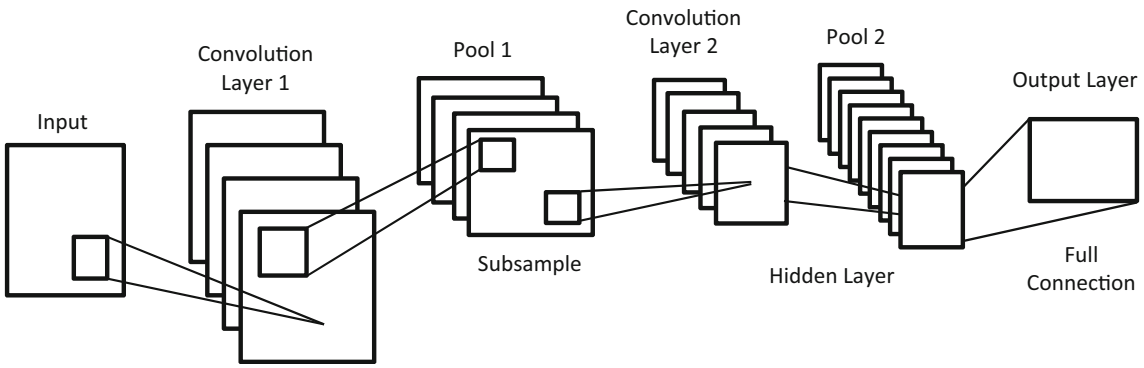



**Fig. 1** Structure of a convoluted neural network

processing pipeline as illustrated in Fig. 2. The first step in the pipeline is to convert the image captured in real time into grayscale and to smooth its edges to reduce noise by using a Gaussian Blur. The next step in the pipeline is to apply a Canny function to facilitate edge detection. After application of the Canny function the edges in the image are obtained by measuring the gradients of adjacent pixels. A high change in gradients is indicative of an edge. A region of interest is created corresponding to the bottom half of the image since that is where the lanes will be found. The lane lines in the image are then obtained in the next step using a Hough transformation.

The parameters of the Hough transformation can be fine-tuned to improve the model accuracy. The parameters are:

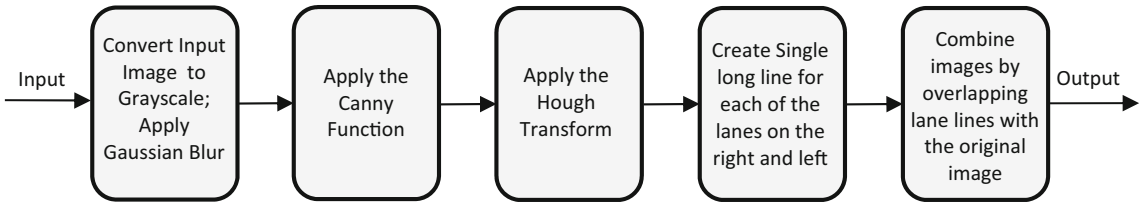



**Fig. 2** Image processing pipeline to detect lane lines

minimum pixels required to create a line, maximum pixel distance between line segments, and the minimum votes required for the algorithm to pick lines based on a confidence level (and to ignore the outliers). Then a single long lane line is created for the left and right lanes respectively. This is done through filtering the lines by slope to determine which ones belong to a certain range and ignoring the others. By doing this, the left and right lanes are found for the region of interest. The final step is to combine the images by overlapping the lane lines with the original image.

### 3.2 Advanced lane detection using computer vision

The lane detection can be improved through the use of sliding window techniques, different thresholds for color spaces and gradients, perspective transformations, and polynomial fit techniques. To achieve this improvement the following steps must be completed:

- Computing the camera calibration matrix and distortion coefficients
- Applying a distortion correction to raw images
- Creating a threshold binary image using color transforms and gradients
- Generating a bird's-eye view of the image using a perspective transformation
- Detection of lane pixels and the lane boundary
- Detection of vehicle position and curvature of the lane with respect to the center
- Displaying numerical estimation of lane curvature and vehicle position and overlapping detected lane boundaries onto the original image

#### *3.2.1 Computing the camera calibration matrix and distortion coefficients*

A problem that some camera lenses have is known as radial distortion. In the approach described here, functions from the Open Computer Vision (OpenCV) are used to remove this distortion (OpenCV 2020). As an example, the correct camera matrix and distortion coefficients for chessboard images can be calculated using OpenCV functions. This is done by determining the corners inside an image and then the appropriate matrix for undistorting the image. The distortion matrix can be tested by using it to undistort a calibration image and show that the calibration is correct.

#### *3.2.2 Applying a distortion correction*

This stage in the pipeline applies a distortion correction to raw images and uses color transforms and gradients to create a threshold binary image. The calibration that was obtained in the previous stage (step 1) can be applied to raw images for distortion correction. The image processing pipeline contains a combination of thresholds, color spaces, and gradients. The selected combination is the S channel threshold in the HLS color space, the V channel threshold in the HSV color space, along with gradients to detect lane lines.

#### *3.2.3 Applying a perspective transformation*

In this stage a bird's-eye view of the image is generated by applying a perspective transformation. The perspective causes lane lines in an image to appear as though they are converging at a distance even though they are parallel to each other. Utilizing this perspective it is easier to remove the curvature of lane lines.

#### *3.2.4 Detecting the lane pixels and lane boundary*

A number of approaches can be used to detect the lane lines. Convolution is used which is the sum of the product of two separate signals: the window template and the vertical slice of the pixel image. The convolution is applied with a sliding window which maximizes the number of hot pixels in each window. The convolved signal is created by sliding the window template across the image from left to right and the overlapped values are summed together. The highest overlap of pixels is the peak of the convolved signal and is the position for the lane marker.

### 3.3 Proposed system for computer vision using spatial transformer networks

The Spatial Transformer Network (STN) applies a learnable transformation which removes spatial invariance from

images followed by interpolation. The structure of the STN is illustrated in Fig. 3. The STN rotates an input image in order to focus on the target object and removes rotational variance. The STN block improves the accuracy of the classifier when placed in a convolutional neural network. Convolution neural networks can experience a lack of robustness due to input variations. These variations include scale, viewpoint, and background clutter. The STN helps to reduce these issues caused by input variations. Note that due to their modularity an STN can be injected into any part of the model. Also, they can be trained with a single back propagation algorithm.

As illustrated in Fig. 3, the STN operates in stages. In the first stage a transformation matrix must be defined to describe the linear transformation. Different transformations correspond to difference matrices. An identity matrix results in an output of the same image. Counter clockwise images are obtained by rotation matrices. A sample mesh grid of the initial image is generated instead of applying the transformation directly to the initial image. A mesh grid is a set of indices that covers the whole input image space and does not contain any color information. The mesh grid can be defined as a matrix–vector multiplication. Finally, a localization neural network (a regressor) learns and produces the correct theta (θ) using the loss back propagated through the sampler. Theta is the activation function with is passed to the bilinear sampler for changing the input of the next network.

### 3.4 Traffic sign detection in self-driving cars

The accurate detection and interpretation of traffic signs is very important for the proper functioning of a self-driving car. In the approach described here, a convolutional neural network utilizing an Adam Optimizer is used for the detection of traffic signs. The optimization algorithm is important as it improves the processing time by minutes, hours, and even days.

#### *3.4.1 Adam optimizer*

The Adam Optimizer is an extension of the stochastic gradient descent algorithm (Cong et al. 2017) and is widely used in solutions for computer vision problems. The word "Adam" refers to the Adaptive Moment Estimation involved in the approach. The advantages of using an Adam Optimizer include:

- The implementation of the algorithm is straightforward.
- The algorithm is computationally efficient.
- The algorithm has a small memory requirement.
- The diagonal rescale of gradients is invariant to the surroundings.
- It is well suited for a complex problem with very large amounts of data.
- Non-stationary objects can be easily detected.

The stochastic gradient descent has a learning rate that does not change during training. The learning rate must adapt itself to a changing environment to determine the changing traffic signs. For this reason the Adam Optimizer is used. It combines the advantages of the Adaptive Gradient Algorithm and the Root Mean Square Propagation technique. The adaptive gradient algorithm is used on problems with sparse gradients since it uses a per parameter learning rate. It works best for computer vision and natural language processing algorithms. The root mean square propagation technique has a per-parameter learning rate that adapts to changes in the weights of the gradient. It works best on noisy data and objects that move.

The Adam Optimizer uses the average of the first and second moments of the gradient. It calculates the squared gradient, average of the gradients, and the decay rates for the averages. The initial values are close to 1.0 which indicates bias. The bias can be removed by calculating the biased estimates. The bias corrected estimates are calculated as specified in Eqs. 3 and 4.

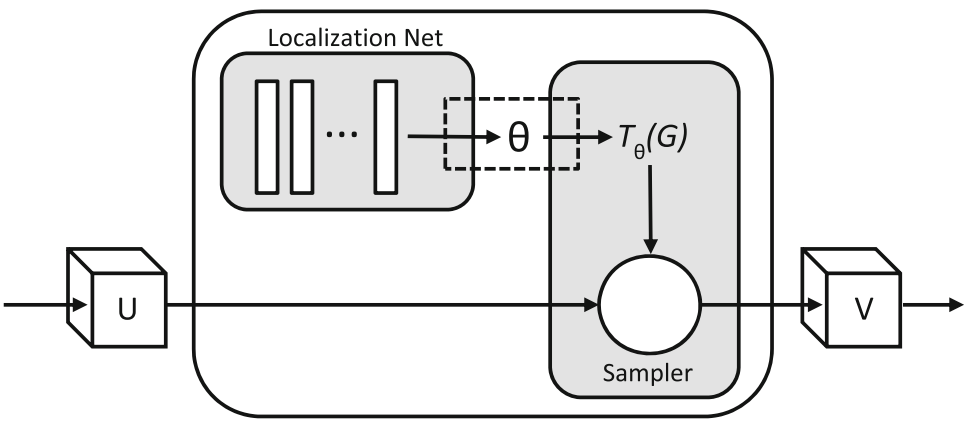



**Fig. 3** Structure of a spatial transformer network

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\alpha_t \quad (3)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\alpha_t^2 \quad (4)$$

The parameters used in the Adam Optimizer algorithm are M, v, $\alpha_t$, $\beta_1$, and $\beta_2$.

- m, v: Both "m" and "v" denote moving averages of the moments. The first moment (m) is the mean, and the second moment (v) is the variance.
- $\alpha_t$: denotes the learning rate of the algorithm. If the value of alpha is high, then the learning rate will be high, and there could be a chance of missing the global optimum. If the learning rate is small, then learning is slower and it can take a prohibitive amount of time to reach the global optimum.
- $\beta_1$: $\beta_1$ is the decay rate for the estimate of the first moment.
- $\beta_2$: $\beta_2$ is the decay rate for the estimate of the second moment. For sparse gradients the value should be set closer to 1.0.

Reasonable values for a "typical" machine learning problem are as follows: $\alpha_t = 0.001$, $\beta_1 = 0.09$ and $\beta_2 = 0.999$.

#### *3.4.2 LeNet-5 architecture*

The LeNet-5 architecture is used for traffic sign detection. As illustrated in Fig. 4, the LeNet-5 architecture is structured into 7 layers. The first layer is a convolutional one followed by an average pooling layer in the second layer. This is repeated again with a convolutional layer in the third stage followed by an average pooling layer in the fourth stage. This is followed in stage 5 with a flattening convolutional layer, then two fully-connected layers, and then a SoftMax classifier.

The first convolutional layer in stage 1 receives as input a 32 × 32 grayscale image that has 6 feature maps with a 5 × 5 size and has a stride of 1. The dimensions of the image are transformed from 32 × 32 × 1 to 28 × 28 × 6. The average pooling layer has a filter size of 2 × 2 with a stride of 2. The image becomes resized to 14 × 14 × 6. Next it is passed onto the second convolutional layer which has 16 feature maps with a size of 5 × 5 and a stride of 1. Of the 16 feature maps only 10 features are connected to 6 feature maps of the previous layer. This is done to keep the number of connections within a reasonable range and to break the network symmetry. This is the reason why the training parameters are 1,516 instead of 2,400 and the connections are 151,600 instead of 240,000.

The fourth layer is also an average pooling layer with a stride of 2 and a filter size of 2 × 2. This layer is similar to the second layer, however it has 16 feature maps so the output will be 5 × 5 × 16. The fifth layer has 120 feature maps each of size 1 × 1 and is a fully connected convolutional layer. Each of the 120 features is connected to the 400 nodes in the fourth layer. The sixth layer has 84 units and is a fully connected layer. The final output layer is also a fully connected layer with 10 values ranging from 0 to 9, as illustrated in Fig. 4.

## 4 German traffic sign dataset

The German Traffic Sign Dataset is an open source dataset that contains over 50,000 images (Stallkamp et al. 2011). The images of the dataset are organized into 43 classes. Each of the classes contains a minimum of 30 images of single physical traffic signs. Each image within the dataset contains a single traffic sign.

### 4.1 Data collection

Each of the classes in the German Traffic Sign Dataset contains images of examples of the same traffic sign. For example, as illustrated in Fig. 5, while one class (class 0) contains images of traffic signs designating a speed limit of 20 km/h, another (class 2) is used to contain images of traffic signs designating a speed limit of 50 km/h. The information in the dataset is represented in a dictionary structure with key and value pairs. There are 4 key/value

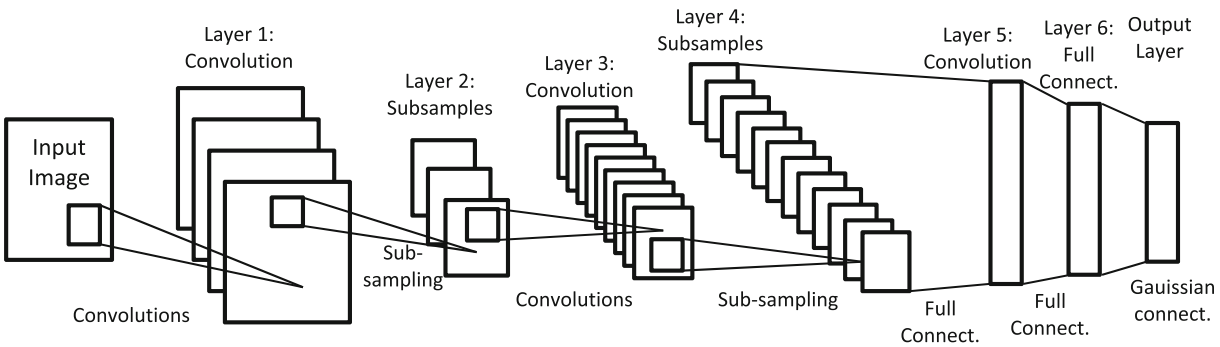



**Fig. 4** Structure of the LeNet-5 model architecture

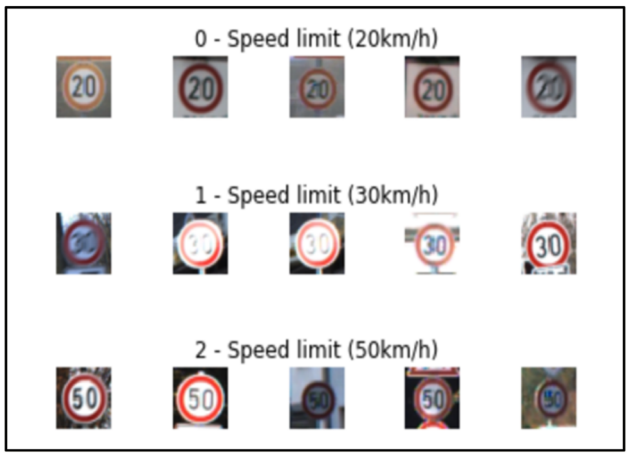



**Fig. 5** Example images from classes of the German traffic dataset (Stallkamp et al. 2011)

pairs relevant for traffic sign detection: features, labels, sizes, and coordinates. Features contain the raw pixel data of different traffic signs. It is a 4-D array containing the width, height, channels, and number of examples. Labels contain the label id of the traffic sign. It is a 1-D array which contains the mappings for each label id. Sizes contain the original width and height of the image. Coordinates represents the bounding box which encloses the image. The coordinates of the original image are present in the coordinates. The resized versions of the images are available in the pickled data.

The model is trained on a previously available dataset containing images of traffic signs. The Adam Optimizer along with the neural network is used to predict and interpret new traffic signs based on the training set. The accuracy of the model can be analyzed using the SoftMax probabilities of the new images.

## 4.2 Testing the LeNet-5 architecture

The following steps were involved in testing the LeNet-5 Architecture:

- Preprocessing by conversion to grayscale and normalization
- Setting up training, validation and testing data
- Training the model
- Measuring accuracy of the LeNet-5 model
- Determining SoftMax probabilities for predictions

The first three of these steps are described in the remainder of this Section. The following two are reported below in the Results Section.

### *4.2.1 Preprocessing—conversion to grayscale and normalization*

The training set data contains approximately 43 classes with varying numbers of images. As illustrated in Fig. 6, the numbers of images in classes ranges from a low of about 200 images to a high of 1800 images. In this preprocessing step, all of the images are converted to grayscale since the color information in images is unnecessary. The grayscale images contain axes which represent the position from which the sliding should take place from pixel to pixel for the purposes of sign detection.

During processing the image is also normalized by dividing the image by 255 as illustrated in Fig. 7. An example of gray scaling is given in Fig. 8.

### *4.2.2 Setting up for training, validation and testing data*

The German Traffic Sign Dataset contains three files relevant to this step. Those files are: train.p, valid.p, and test.p. Note the ".p" file extension refers to a pickle file. A pickle file is one that contains information that represents a Python object hierarchy that has been converted to a byte stream. The training data contains 34,799 images, the testing data contains 12,630 images, and the validation data contains 4410 images. The deep learning model is trained on training data and then the trained model is applied to the valid file to determine the model accuracy and the model loss. Note the model accuracy refers to how well the model performed for unseen images and the model loss corresponds to the difference between the label predicted by the model for an image and the true label for the image.

### *4.2.3 Model training*

The model is trained by using the Adam Optimizer in the neural network. The steps require are as follows:

- The training data is sent through the training pipeline to train the model.
- The training set is shuffled at each epoch. At each epoch approximately 2000 images are trained for the model.
- The accuracy and the loss of the validation set is measure after each epoch.
- The model is saved after training and is applied to the test data to correctly predict the unseen traffic signs.

During the model training, the activation function used is SoftMax and the loss function used is categorical cross entropy. SoftMax is an activation function that turns numbers into probabilities that sum to one. Categorical entropy compares the predicted label for signs and the true label for signs in order to calculate the loss.

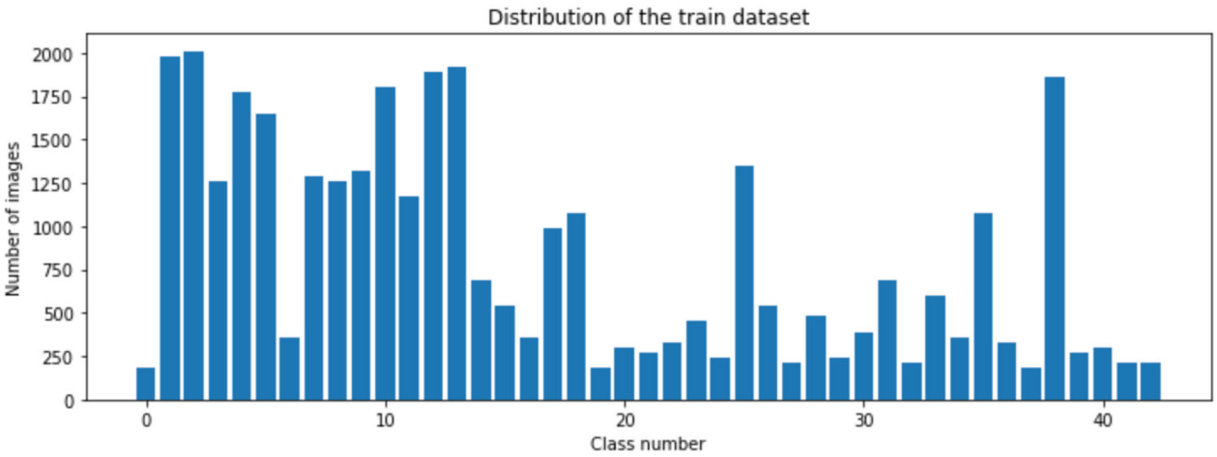



**Fig. 6** Distribution of images across different classes

**Fig. 7** A traffic sign image with normalization

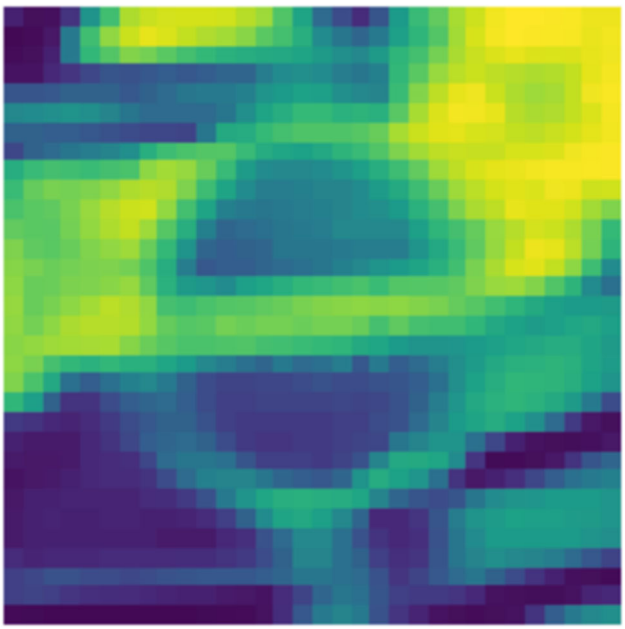


**Fig. 8** A traffic sign image after gray scaling

## 5 Results

After performing training on the deep learning model and determining its accuracy using the validation data, the model is used to determine user provided traffic signs. It must be able to correctly predict traffic signs.

### 5.1 Model accuracy

The accuracy of the training model shows how well the model performed after training. Based on this training the model must be able to detect and determine user provided traffic signs correctly. As illustrated in Fig. 9, the accuracy of the model is 97%.

The loss value is used to determine how well the algorithm models the given data. Figure 10 illustrates the loss value. Sometimes the model over fits the data. To correct this number of neurons can be increased by 3–4 times, then the number of labels will increase, and the overfitting can be removed. The addition of extra layers is also used to remove any bias in training the model. Bias causes the model to be prejudiced to a particular class of images in the training set and when the model is used on the test set, it decreases the efficiency of the approach.

### 5.2 SoftMax probabilities for each prediction

The SoftMax probability calculates the probability distribution over $n$ different events. The SoftMax function

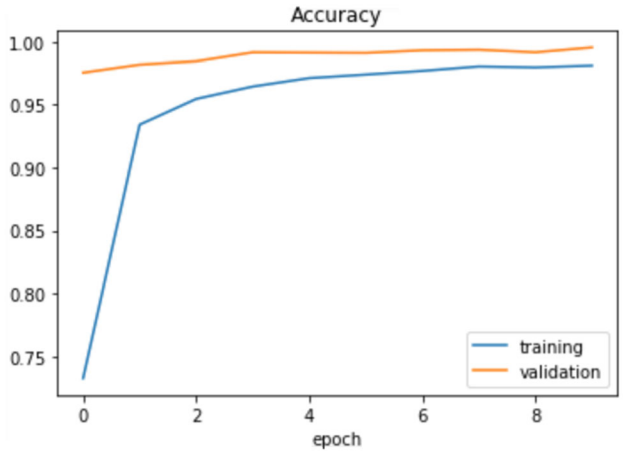



**Fig. 9** The accuracy of training and validation data for the LeNet-5 architecture

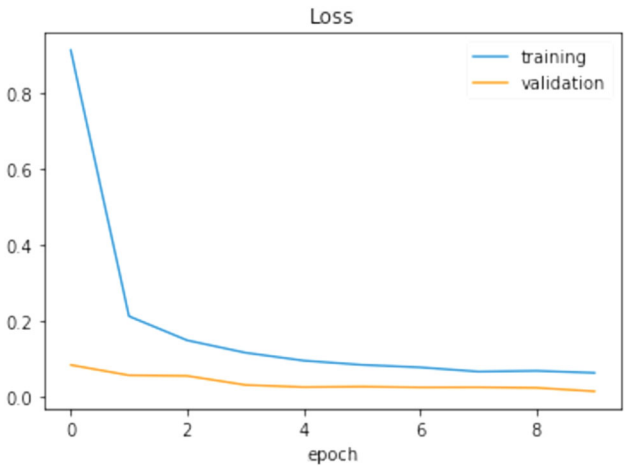



**Fig. 10** The loss value of training and validation data for the LeNet-5 architecture

calculates the probabilities of each target class with the target classes, and then these probabilities based on the given input will help in determining the target class. The range of SoftMax probabilities is from 0 to 1, and the sum of all the probabilities is equal to one. For each new image in the dataset, the model's SoftMax probabilities shows the certainty of the model's predictions. It returns the values and indices of the top $k$ predictions. The model is tested on new images and the probabilities for the prediction are as shown in Table 2.

The user provided traffic signs are obtained from the data archive provided by German researchers at Ruhr-Universitaet Bochum (Stallkamp et al. 2011). The following pairs of images depict an image of a traffic sign that was provided as input to the model along with the prediction that was made by the model. Figure 11a depicts an image of the traffic sign for a speed limit of 30 km/h. given as input to the model, and Fig. 11b shows the model's

**Table 2** The SoftMax probability for the predicted images

| Image | Top 5 probabilities | | | | |
|---|---|---|---|---|---|
| 1st image | 1 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2nd image | 1 | 0.0 | 0.0 | 0.0 | 0.0 |
| 3rd image | 1 | 0.0 | 0.0 | 0.0 | 0.0 |
| 4th image | 1 | 0.0 | 0.0 | 0.0 | 0.0 |
| 5th image | 0.82 | 0.07 | 0.06 | 0.02 | 0.01 |

correct prediction of class 1 for speed limit of 30 km/h. with an accuracy of 96%.

In Fig. 12a the image of a traffic sign for a left turn that was given as input to the model is depicted. The correct prediction by the model of class 34 is illustrated in Fig. 12b.

In Fig. 13a the image of a slippery road traffic sign that was given as input to the model is depicted. As illustrated in Fig. 13b the model accurately predicted that image as belonging to class 23.

In Fig. 14a the image of a traffic sign for yield right of way that was given as input to the model is depicted. The correct prediction by the model of the image belonging to class 13 is illustrated in Fig. 14b.

Finally, in Fig. 15a the image of a traffic sign for yield to cycles that was given as input to the model is depicted. The correct prediction by the model of the image belonging to class 29 is illustrated in Fig. 15b.

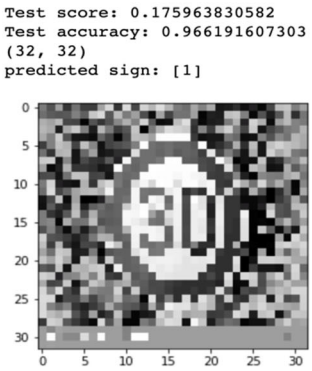
### 5.3 Model performance comparison with a feed forward neural network

A feed forward neural network is an artificial neural network where all the connections between the nodes move in one direction and no cycles exist between the nodes (Garimella et al. 2017). The information is initially provided to the input layer and then it is passed into the following consecutive layers of the network. Each layer of the network performs computation with the output from one layer being passed on to the next. However, as no cycles exist, the information never flows in a backward direction. Hence it is known as a feed forward neural network. As illustrated in Fig. 16, the computed information emerges from the output layer of the neural network.

The accuracy of the feed forward neural network for training data decreases after each epoch. The accuracy initially was 90% and reduced to 83% by the last epoch. The lower accuracy also affects the validation data as the accuracy does not increase significantly after each epoch. The accuracy of the validation data increases from 87 to 88% as illustrated in Fig. 17.

**Fig. 11** **a** 30 km/h traffic sign image. **b** Correct model prediction of Class 1

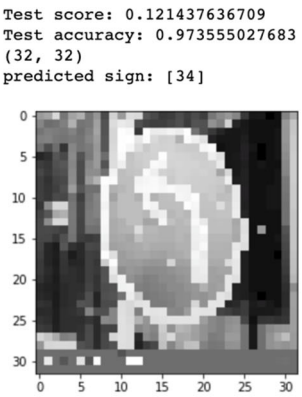
**(a)** 30 km/hr. traffic sign image.




**(b)** Correct model prediction of Class 1.

**Fig. 12** **a** Left turn traffic sign image. **b** Correct model prediction of Class 34

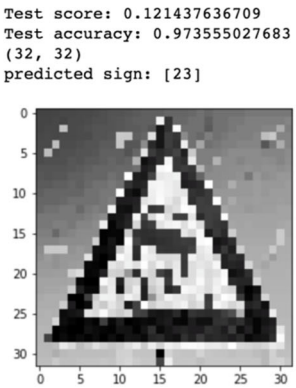
**(a)** Left turn traffic sign image.




**(b)** Correct model prediction of Class 34.

**Fig. 13** **a** Slippery road traffic sign image. **b** Correct model prediction of Class 23

**(a)** Slippery road traffic sign image.




**(b)** Correct model prediction of Class 23.

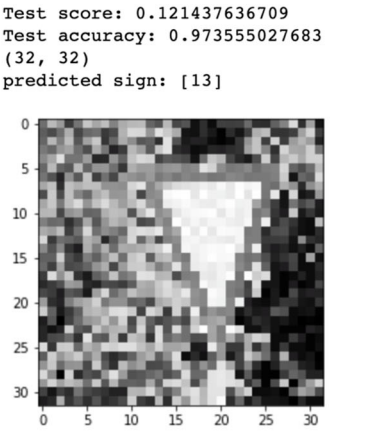
As shown in Fig. 18, the loss in the training and validation data is also high. The loss for the validation data decreases from 46 to 41% but this small reduction in loss is significant, and shows that the feed forward neural network does not perform well compared to the LeNet-5 architecture.

## 6 Discussion

As technology marches forward, self-driving cars will become an increasingly integral part of society in the near future. Spatial transformer networks that run on top of an image processing pipeline such as described in this paper can be used as an excellent solution for handling the

**Fig. 14** **a** Yield right of way traffic sign image. **b** Correct model prediction of Class 13

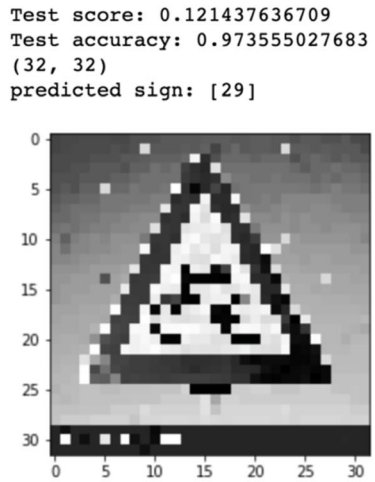
**(a)** Yield right of way traffic sign image.




**(b)** Correct model prediction of Class 13.

**Fig. 15** **a** Yield to cycles traffic sign image. **b** Correct model prediction of Class 29

**(a)** Yield to cycles traffic sign image.




**(b)** Correct model prediction of Class 29.

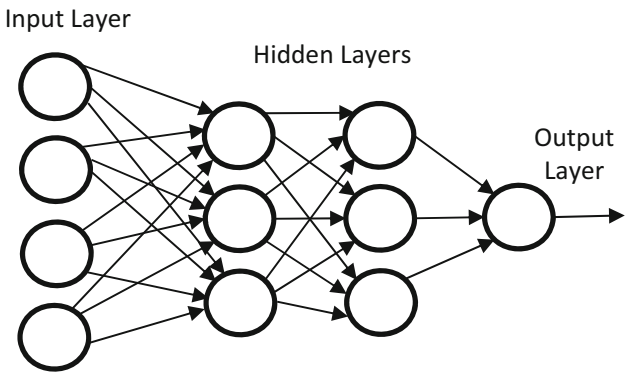



**Fig. 16** The structure of a feed forward neural network

critically important lane detection task for supporting autonomous vehicles. Additionally, traffic sign detection using a LeNet-5 architecture along with an Adam Optimizer as described in this paper provides an accurate strategy for detecting traffic signs which also has the important benefit of being computationally efficient.

Though the LeNet-5 architecture was able to detect most of the traffic signs, on some particular signs it performed less well than others. For example, it sometimes did not correctly detect the bicycle traffic sign and wrongly classified it as a bicycle crossing sign. A characteristic of the dataset used is that the number of images for each class of sign can vary considerably. While some classes have numerous images, some have much fewer. In this case, the error shows that the model requires more data for a particular sign (e.g., the bicycle traffic sign) in order to consistently classify it correctly.

As indicated in Table 3, the accuracy of the LeNet-5 model was higher than that of the feed forward neural network. This is especially true when used with the Adam Optimizer. The combination of the two as utilized in the approach suggested in this paper was found to have a high model accuracy of 97%. The presence of additional pooling

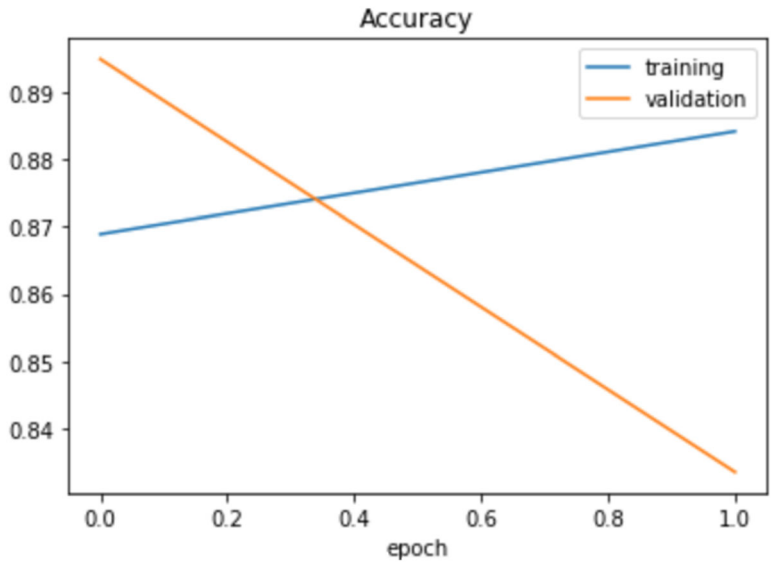



**Fig. 17** The accuracy of training and validation data

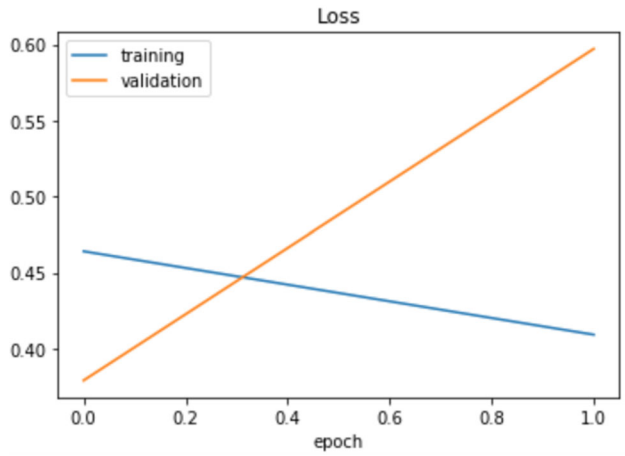



**Fig. 18** The loss of training and validation data

**Table 3** Comparison of model accuracies

| Model | Accuracy (%) |
|---|---|
| Feed Forward Neural Network | 94 |
| LeNet-5 with Adam Optimizer | 97 |

layers increases the efficiency of the model. Thus the LeNet-5 model has better accuracy than the feed forward neural network with an accuracy of 94%. The feed forward neural network suffers from the fact that the information can only flow in one direction in the neural network. Information does not reach every node and this results in information loss which affects the efficiency and accuracy of the model.

## 7 Conclusion

The paper has presented a deep learning methodology of Convolution Neural Network (CNN) using Spatial Transformer Network (STN) that utilizes a calibration matrix and distortion coefficients for lane detection of self-driving autonomous cars. The lane pixels and lane boundaries are detected in order to keep track of the lane in advance. LeNet-5 architecture with Adam Optimizer is used for traffic sign detection. The methodology is demonstrated on an open source German Traffic Sign dataset. The Adam Optimizer algorithm was used in reaching the global optimum and to determine the correct learning rate. The Adam Optimizer was used as it requires less memory to run and can process complex image data quickly. It is also efficient in detecting objects that are in constant motion. The proposed methodology provides greater accuracy in comparison with other methodologies such as feed forward network. Research in areas such as these has the potential to make important contributions towards the ultimate goal of a fully automated self-driving vehicle.

**Acknowledgements** The authors would like to thank the researchers at the German Ruht-Universitaet Bochum in Germany for making available the very useful and comprehensive German Traffic Sign Dataset (http://benchmark.ini.rub.de/) that was used in this research (Stallkamp et al. 2011). This research was completed as a master's graduate research project at the Department of Electrical Engineering and Computer Science at Texas A&M University - Kingsville.

**Declarations**

**Conflict of interest** The authors declare that there is no conflict of interest.

## References


Aziz MVG, Prihatmanto AS, Hindersah H (2017) Implementation of lane detection algorithm for self-driving car on toll road cipularang using Python language. In: 2017 4th international conference on electric vehicular technology (ICEVT), pp 144–148. IEEE

Chatrati SP, Hossain G, Goyal A, Bhan A, Bhattacharya S, Gaurav D, Tiwari SM (2020) Smart home health monitoring system for predicting type 2 diabetes and hypertension. J King Saud Univ Comput Inform Sci 24:1–9

Cong G, Bhardwaj O (2017) A hierarchical, bulk-synchronous stochastic gradient descent algorithm for deep-learning applications on gpu clusters. In: 2017 16th IEEE international conference on machine learning and applications (ICMLA), pp 818–821. IEEE

Dorj B, Hossain S, Lee DJ (2020) Highly curved lane detection algorithms based on Kalman filter. Appl Sci 10(7):2372

Dwivedi R, Dey S, Chakraborty C, Tiwari S (2021) Grape disease detection network based on multi-task learning and attention features. IEEE Sens J

Gadekallu TR, Rajput DS, Reddy MPK, Lakshmanna K, Bhattacharya S, Singh S, Alazab M (2020) A novel PCA–whale optimization-based deep neural network model for classification of tomato plant diseases using GPU. J Real Time Image Process 493:1–14

Garimella G, Funke J, Wang C, Kobilarov M (2017) Neural network modeling for steering control of an autonomous vehicle. In: 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp 2609–2615. IEEE

Gaurav D, Tiwari SM, Goyal A, Gandhi N, Abraham A (2020) Machine intelligence-based algorithms for spam filtering on document labeling. Soft Comput 24(13):9625–9638

Gupta T, Sikchi HS, Charkravarty D (2018) Robust lane detection using multiple features. In: 2018 IEEE intelligent vehicles symposium (IV), pp 1470–1475. IEEE

Kim S, Lee J, Kim Y (2016) Speed-adaptive ratio-based lane detection algorithm for self-driving vehicles. In: 2016 international SoC design conference (ISOCC), pp 269–270. IEEE

Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput vis 60(2):91–110

Mishra S, Sagban R, Yakoob A, Gandhi N (2018) Swarm intelligence in anomaly detection systems: an overview. Int J Comput Appl 43:1–10

Nugraha BT, Su SF (2017) Towards self-driving car using convolutional neural network and road lane detector. In: 2017 2nd international conference on automation, cognitive science, optics, micro electro-mechanical system, and information technology (ICACOMIT), pp 65–69. IEEE

OpenCV (Open Source Computer Vision Library). https://opencv.org. Accessed 4 April 2020

Park H (2018) Implementation of lane detection algorithm for self-driving vehicles using tensor flow. In: International conference on innovative mobile and internet services in ubiquitous computing, pp 438–447. Springer, Cham

Prabhakar G, Kailath B, Natarajan S, Kumar R (2017) Obstacle detection and classification using deep learning for tracking in high-speed autonomous driving. In: 2017 IEEE region 10 symposium (TENSYMP), pp 1–6. IEEE

Rahul M, Kohli N, Agarwal R, Mishra S (2019) Facial expression recognition using geometric features and modified hidden Markov model. Int J Grid Util Comput 10(5):488–496

Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: towards real-time object detection with region proposal networks. arXiv preprint arXiv:1506.01497

Renjith R, Reshma R, Arun KV (2017) Design and implementation of traffic sign and obstacle detection in a self-driving car using SURF detector and Brute force matcher. In: 2017 IEEE international conference on power, control, signals and instrumentation engineering (ICPCSI), pp 1985–1989. IEEE

Sant A, Garg L, Xuereb P, Chakraborty C (2021) A novel green IoT-based pay-as-you-go smart parking system. CMC Comput Mater Cont 67(3):3523–3544

Shapiro LG, Stockman GC (2001) Computer vision. Prentice Hall PTR, Upper Saddle River

Singh G, Chowdhary M, Kumar A, Bahl R (2020) A personalized classifier for human motion activities with semi-supervised learning. IEEE Trans Consum Electron 66(4):346–355

Stallkamp J, Schlipsing M, Salmen J, Igel C (2011) The German traffic sign recognition benchmark: a multi-class classification competition. In: The 2011 international joint conference on neural networks, pp 1453–1460. IEEE

Sucharitha M, Chakraborty C, Rao SS, Reddy VSK (2020) Computer vision for brain tissue segmentation. In: Green computing and predictive analytics for healthcare, pp 81–94. Chapman and Hall/CRC

Zhang W, Wu QJ, Yang X, Fang X (2008) Multilevel framework to detect and handle vehicle occlusion. IEEE Trans Intell Transp Syst 9(1):161–174