BOHR Journal of Computational Intelligence and Communication Network
2023, Vol. 1, No. 1, pp. 7–15
DOI: 10.54646/bjcicn.2023.02
www.bohrpub.com

REVIEW

Recognizing traffic signs with synthetic data and deep learning

Avaz Naghipour* **and Rahim Pasbani**

Department of Computer Engineering, University College of Nabi Akram, Tabriz, Iran

***Correspondence:**
Avaz Naghipour,
naghipour@ucna.ac.ir

Received: 02 January 2023; **Accepted:** 10 January 2023; **Published:** 19 January 2023

Recently, in-depth learning about computer vision and object classification tasks has surpassed other machine learning (ML) algorithms. This algorithm, alike similar ML algorithms, requires a dataset for training. In most real cases, developing an appropriate dataset is expensive and time-consuming. Also, in some situations, providing the dataset is unsafe or even impossible. In this paper, we proposed a novel framework for traffic sign recognition using synthetic data and deep learning. The main feature of the proposed method is its independence from the real-life dataset, which leads to high accuracy in the real test dataset. Creating one-by-one synthetic data is more labor-intensive and costlier than providing real data. To tackle the issue, the proposed framework uses a procedural method, which gives the possibility to develop countless high-quality data that are close enough to the real data. Due to its procedural nature, this framework can be easily edited and tuned.

Keywords: deep learning, convolutional neural networks, computer graphics, synthetic data, traffic sign recognition

Introduction

Nowadays, many kinds of research and innovations are conducted to enhance autonomous vehicle technologies. Giving a semantic perception to artificial intelligence (AI)-based drivers to recognize environmental objects is one of the main goals in research (1–3). Indisputably, traffic sign recognition ability plays a significant role in AI-based vehicles. These signs are guidance that makes drivers aware of upcoming situations. Thus, traffic sign detection and recognition that computer vision applications are trying to address are considered a significant issue. While having a good dataset for ML applications (especially in supervised ML) is mandatory, there are a few premade datasets available on demand (4). Accessible free datasets are usually used to benchmark competition or evaluate state-of-the-art applications. For preparing production-level applications, first, it is crucial to provide a proper dataset with adequate quantity and quality. In the case of an image classifier, the datasets are normally images captured with cameras from real-life instances. At the first glance, providing images seems to be a handy and cost-efficient procedure; however, when the required number of images for the train classifiers is taken into consideration, the difficulty of preparing such datasets becomes bold. ML algorithms in general and deep learning in particular use thousands or even millions of images to give a reliable and practical result. Obviously, providing this amount of image in many cases is wearisome and costly, if not impossible.

Other than the cost and expenses, a key issue with providing a traffic sign dataset is time. For more clarity, let us conjure up the traffic sign dataset obtaining procedure, and how time-consuming it would be to capture, crop, edit, and label singly and manually. A more significant problem is the time needed to capture photos in different seasons and conditions of a year. For instance, if images are captured only during a hot summer, the dataset would not include the images of signs covered with snow during winter, and encountering such images causes trouble for the classifier. Thus, generalizing the model comprehensively

requires at least 1 year of waiting, to include all seasonal visual appearances.

Another solution is using CAD datasets. Synthetic images rendered from a 3D virtual scene have been used vastly in computer vision tasks (5). Recently, they are used in object detection and classifier applications. Flying Chairs (6), FlyingThings3D (7), SYNTHIA (8), and Scene Net (9) are examples of synthetic images based on datasets that are used to train or evaluate relevant ML algorithms.

Fortunately, by progressing in the computer graphics (CG) industry, a number of online CAD datasets and premade 3D objects are growing. However, except for very few datasets, there has been no access to CAD models yet. Considering the issue of making CAD data, it is perceptible that capturing real images may be cheaper than developing synthetic ones. Modeling even a very simple 3D model is a time-consuming process that requires experts to be accomplished.

Another problem with most synthetic images is their dissimilarity to real objects. These images are far distinctive from real-world references in terms of appearance. By browsing some accessible CAD datasets, it can be explicitly seen that the objects do not have the proper lighting as we have in the real world. Another significant issue that makes CAD models look rough is the texture and material of models. Instead of resembling real materials such as wood, fibers, and metals, those models seem to be made of solid clay. If the ML application is trained by non-real-like images, the result will not give adequate accuracy in ground-truth cases. This is why sometimes researchers choose to mix them with some real images to improve the functionality of the models (10).

To overcome these challenges, an efficient approach is proposed in the present work to develop synthetic traffic signs. To this end, a procedural way is used to provide the desired dataset without any quantity limitation. In the proposed method, every small detail is taken into account to make the images quite real-looking, so that they can hardly be recognized from real images. To do so, various ML classifiers were employed to be trained by the developed dataset. The outline of the paper is as follows: Section 2 discusses related works. Section 3 presents synthetic image generation. The overview of image processing filters and image augmentation are studied in Sections 4 and 5, respectively. Section 6 describes setting up deep convolutional neural network (DCNN) architecture. In Section 7, experiments and results are reported. Section 8 concludes this paper.

Related works

There are various algorithms for classifying traffic signs. The most regarded algorithms in this field are based on ML methods; however, there are some research projects that employ the color and shape of the signs. These characteristics cannot be directly used to classify the traffic signs, but they can remarkably help the actual classifier.

The support vector machine (SVM) for classification has always been a selection at hand. In ref. (11), Maldonado et al. used SVM for automatic multiclass traffic sign detection and classification using a one-vs-all approach with a Gaussian kernel. In the other attempt (12), by considering the limitation in the shape and color of signs, authors used a color segmentation and shape matching approach, and then, the dataset has classified using SVM. The obtained results are promising. In the method suggested in ref. (13), after the detection of a sign via the MSER procedure, the HSV-HOG-LBP features are extracted, and then, a random forest is used to finalize the recognition process. Ref. (14) has tried to prove the effectiveness of the random forest recognition algorithm in both accuracy and speed on traffic sign recognition. Nowadays, modern computer vision classifiers mostly deploy CNN for recognition tasks. In ref. (15), the densely connected CNN is used for traffic sign detection. In ref. (16), the authors have shown the results of different architectures of CNN to solve the same recognition problem.

In the area of synthetic data deployment, some remarkable works have been presented so far. The authors in ref. (17) have suggested a 2D synthetic text generator engine, which places texts onto random backgrounds and employs the obtained data to train a CNN-based classifier to recognize texts in graphics. Furthermore, a 3D synthetic dataset was used in ref. (10) to predict hand gestures in an image. The accuracy has risen after adding some real images to the training dataset. The CAD data have been used in some research for classifying and object detection tasks (18, 19). Another important challenge in computer vision is viewport prediction. In ref. (20), rendered images trained a CNN, and the result was excellent. Most of the relevant methods have used premade online CAD libraries, such as Trimble 3D warehouse, TurboSquid, Yobi3D, and ShapeNet. Using premade datasets is handy to test new models or benchmark competitions. But, in real applications, any problem needs its own exclusively developed dataset to be prepared.

Synthetic image generation

The proposed method in this research contains two main steps. In the first step, a procedural virtual 3D scene is created, and in the second step, a specific DCNN architecture is designed to be trained by the generated dataset obtained in the previous step.

First of all, a virtual 3D scene needs a setup. Then, procedurally, in 3D world space, feasible variations, and randomizations are made, so that every state forms a believable arrangement of the scene's objects and components. By every run, a specific arrangement is made and rendered. In the next step, to make extra purposeful variations, some image processing-based filters

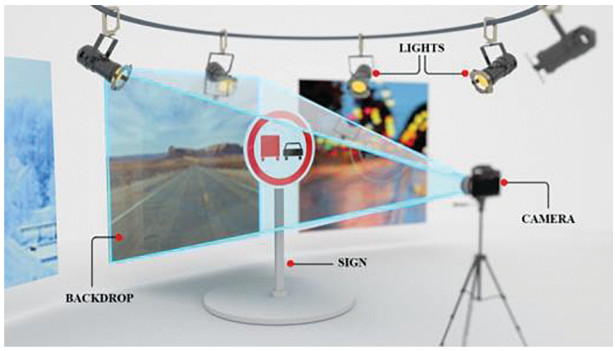

FIGURE 1 | 3D scene contains 3D objects, lights, sky, background, and some controllers, which control every aspect of the randomization process.

and manipulations are applied to the rendered images. Finally, the image augmentation technique is employed to generalize the proposed classifier, as well as, image augmentation increases the size of the training dataset.

The aim was to set up a versatile virtual scene that can automatically develop a feasible arrangement of the objects. To reach such a system, some objects are required such as some 3D objects, lights, sky objects, and several controllers that are used to control the properties of scene components. The controllers provide mathematical relationships among all objects in the scene. To avoid infeasible cases, some constraints are considered. In fact, the constraints are small codes written in Python, which control the randomization process to prevent impractical setups. **Figure 1** depicts a schematic view of the mentioned procedural scene.

For evaluating the proposed approach in the real world, the German Traffic Sign Recognition Benchmark (GTSRB) dataset is selected. This dataset was captured in different lighting situations. Some of them were captured on a sunny day and some in shadow or bluish-morning-like lighting. Moreover, there are some images that became overexposed due to the reflective surface of the sign board. Also, there can be seen motion blur in some images, indicating that the pictures are captured while driving.

To achieve a comprehensive model that is capable of classifying the different types of signs, the image generator has to be very versatile with the ability to cover all of the possible variations. Some of the variations applied on the scene are listed as follows.

- **Illumination:** One of the main issues in rendering photo-realistic images refers to the correct lighting of the scene. In CG, the lighting procedure is almost divided into two main parts, i.e., direct lighting and indirect lighting. Direct light takes charge of the main illumination, which usually casts sharp shadows on objects, and on the other hand, indirect light is an environmental light that is resulted from bouncing rays. Since traffic signs are almost always placed outdoors, they are mostly illuminated by the sun and sky. The sun is considered a direct light, and the sky is responsible for indirect lighting. In the real world rules, both the sun angle and sky color are intertwined (21). The sky color gradient varies according to the sun's position and some other factors such as the haze and aerosol in the air. Simulation of the sun as an infinite light source (direct lighting source) in most CG applications is simple. The technique that is usually used for indirect lighting simulation is referred to as image-based lighting (IBL). In this method, a big sphere or hemisphere surrounds the scene, and its texture sends light rays into the scene. In this research, the Preetham sky model was used to generate a virtual sky. This model needs the sun position, the viewing direction, and the turbidity factor to compute the color of the texture pixels (22). Turbidity is defined as the haziness of fluid-type materials. To achieve a different range of sky models, the sun's position is randomized around the scene; moreover, for every run, a random integer number between 2 and 10 is designated to the turbidity factor. The sky color is allowed to affect the background image to match the scene's overall color.
- **Position and rotation:** By any run, the basic spatial properties of the sign object, such as position and rotation, change, but they never get out of the camera view. Both camera and objects have a chance to relocate or spin. By looking at the GTSRB images as the reference, the minimum and maximum available space around the sign object can be estimated. We just try to limit the movement of the sign object, so that it sticks in the middle of the frame.
- **Motion blur and out of focus:** Motion blur may happen when we try to take a picture of fast-moving objects. This phenomenon directly depends on the shutter speed of a camera. Another effect is called out of focus. This effect occurs when a certain object is far from the camera's focal distance. Both of the effects above can be simply simulated by specific image processing filters even after rendering. To simulate the motion blur effect, usually, some filters are applied to images that stretch the image along the moving direction. In this research, the direction is selected randomly but is almost near the horizontal line.
- **Signboard damages and imperfections:** Usually road signs are exposed to physical damage and strikes. These damages often cause deformation. To mimic this effect, some deformers have been deployed. Deforming is usually done by using displacement maps. These maps are gray-scale noisy images projected onto object UV coordinates and push polygons up or down corresponding to the brightness of the projected map, along polygon normal vectors. By every run, this map is regenerated with a different noisy pattern.
- **Dust and scratches:** Rain, storm, snow, dust, and other natural phenomena may dirty the signs and

makes them unclear. Some controllers are designed to simulate these types of effects by adding some random pattern onto sign textures. For adding more details, divers' noisy images are deployed to fake dirtiness on the sign boards. Also, some mask textures specify the areas where this dirtiness should appear.

- **Backdrop and environment:** Each season has its own visual effects on the objects' appearance. In rendered images, these effects can be realized by changing the background image. An important factor to be taken into account is that neural networks can learn unwanted patterns such as backgrounds. Thus, we should be aware of using repetitive backdrops as much as possible. To prevent this side effect, the proposed method uses one hundred different images; however, the risk is yet probable by every 100 runs. Hence, for every run, the position, scale, and rotation of the background images are changed randomly. This can guarantee that final rendered images will never have the same background. These randomizations are controlled by controllers in order to prevent infeasible images.
- **Shadows:** The sign objects are placed in different positions that may receive any type of shadows cast by other objects. These shadows, when analyzed numerically, have a significant effect on their appearance and color. To implement these shadows in a virtual world, several objects with different shapes and sizes have been settled in the scene. With each run, some properties of these objects, such as positions, rotations, distances, and visibilities, become random. Shadows play a significant role in the overall looking of any image. For more clarity, in **Figure 2**, the same scene has been rendered four times just by changing in received shadows. Every time each image's histogram has been plotted. As seen on these plots, most pixel values were changed while semantically all of these images represent the same sign. In addition to the light and shadows, any change in position, rotation, scale, shear, color, and other properties will overturn pixel data. Thus, designing a classifier that remains invariant to all these variations is a big challenge both in image processing and computer vision fields. So the goal is to provide a comprehensive dataset that is able to include the road signs in any condition. This makes the classifier behave invariant toward unnecessary information.

Image processing filters

Pictures captured by ordinary cameras often contain some noise. This noise mostly can be seen explicitly in low-light situations. Also in rendering, due to indirect illumination, inherently all rendered images are noisy. However, for emphasizing, some subtle noise is randomly added to the rendered images.

By browsing GTSRB data more accurately, it can be seen that some images are very dark and some images are very bright. Despite the different lighting situations regarded in the 3D scene rendering step, some extra darkening and brightening filters are applied to some rendered images. As mentioned before, the motion blur and defocus can be faked by 2D filters. In this step, these effects are also applied to randomly selected rendered images.

After many trials and errors, it was found that the mentioned filters help the final result and accuracy get better.

Image augmentation

At the final step of dataset preparation, all the rendered images are candidates for applying augmentation. In this step, all the variations supposed to be applied on rendered images are offline, and there is no access to 3D scene options anymore. Intense variations on images are applied because the GTSRB dataset includes images captured in very diverse situations. These situations, even with the human eye, are hard to recognize. In general, image augmentation leads a robust training and a reduction in overfitting. The augmentation used in this work changes almost every property of rendered images, such as position, rotation, scale, shear, crop, contrast, distortion, random masking shapes, and some color perturbation. In **Figure 3**, some augmented images are illustrated.

Eventually, the proposed synthetic image generator produced 2,500 images for each class. Since this generator is entirely procedural, it is possible to create an infinite number of images without much effort. Moreover, this method avoids repetitive images in the generated dataset. Of these 2,500 images, 2,000 of them were allocated for training and 500 for validation (per class).

To assess the proposed method, 12 classes of the GTSRB dataset are selected. Intentionally, some challenging and difficult classes are chosen so that they are similar to each other in terms of shape, figure, or color. In **Figure 4**, these selected classes are illustrated. These 12 classes in the GTSRB dataset aggregately contain about 10,000 images that will be considered as a test set to evaluate the classifier efficiency.

Setting up DCNN architecture

CNN is one of the major types of feed-forward neural networks that can track the spatial position of elements in the images as detection features (23). These features carry meaningful data, which play the main role in detection and recognition tasks. This advantage makes the CNNs more efficient than the Multi-Layer Perceptron (MLP) in image classification tasks. Some other types of layers are embedded

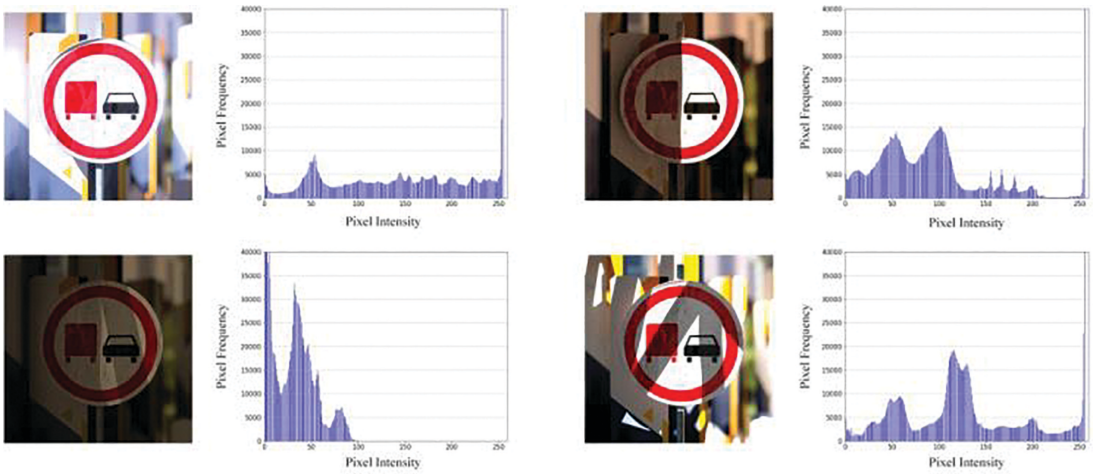

FIGURE 2 | Histogram of an object with four different shadow situations. Shadow cast by the environment; changes the entire distribution of pixels' data.

FIGURE 3 | Heavy augmentation is applied to rendered images. Most of the image properties have been changed during this operation, such as position, scale, crop, distortion, and color.

FIGURE 4 | Selected sign types for generating synthetic images.

between the convolution layers to reduce dimension (pooling layer) or add non-linearity (activation function) to the layer's output (24).

Since this work is aimed at providing the fact that synthetic data can be used to train the CNN models, the utilized model in this work is not precisely optimized.

The proposed DCNN architecture contains four blocks before connecting to the two fully connected layers. There are two convolution layers with 32 filters in the first block. Then, batch normalization is added to speed up and improve the accuracy of the training process (25). Later, a max pooling with the pool size of (2, 2) and stride 2 shrinks the size of the first block from 80 80-pixel to 40 40-pixel. After the first dense layer, a dropout layer is added as the regularization method to improve the generalization errors of the network. Additionally, dropout has a tremendous role in avoiding the overfitting problem (26). For the first two blocks, two convolution layers are successively used without pooling between them. One reason is the result of using pooling after each convolution layer, and the size of the tensors immediately gets smaller, so, some significant data may be lost. Besides, the consecutive convolution layers result in more spatial data in the feature map (27). The numbers of the filters used for the next convolution layers are 64, 64, 128, and 256, respectively.

In this work, the "max pooling" method is used for the pooling layer. All the utilized activation functions are Rectified Linear Units (ReLUs). These blocks finally ended with two fully connected layers. These layers usually are used to collect and optimize scores for each class. The first fully connected layer contains 128 neurons, and a batch

normal and dropout follow it. The last layer is the second fully connected with only 12 neurons, and softmax is used as the activation function. This layer decides that the input image belongs to which class. The mentioned architecture was schematically plotted and can be seen in **Figure 5**.

The proposed model is ready to receive the provided synthetic images as input to begin the training process. But for achieving optimum weights and biases, a proper loss function must be established. Imagine that **x** is an instance image vector and s_k (**x**) is the score of class k which softmax computes, so there is a linear relationship between **x** and the score as below (28):

$$s_k(\mathbf{x}) = \mathbf{x}^T \theta^{(k)} \tag{1}$$

In Equation (1), $\theta^{(k)}$ represents parameter vector for class k. We need the probability of belonging to class k, so the softmax function at the end of the model chain calculates this probability ($\hat{p}_k$) (28):

$$\hat{p}_k = \frac{e^{s_k(\mathbf{x})}}{\sum_{j=1}^{K} e^{s_j(\mathbf{x})}} \tag{2}$$

where K is the number of classes.

Since the softmax predicts only one class per time, this is suitable for our case as every sign only belongs to one class. Cross entropy is a proven way for classification problems to define a loss function (28):

$$J(\Theta) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} y_k^{(i)} \log\left(\hat{p}_k^{(i)}\right) \tag{3}$$

Now the cost function $J(\Theta)$ can be obtained by forming Eq. (3). In this equation, $y_k^{(i)}$ is the true label of instance i that belongs to class k. This value is 1 if i^{th} instance belongs to the class k and 0 in other cases. To obtain the gradient vector of class k, it needs to calculate the gradient of the cost function with respect to k^{th} parameter ($\theta^{(k)}$) (28):

$$\nabla_{\theta^{(k)}} J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} (\hat{p}_k^{(i)} - y_k^{(i)}) \mathbf{x}^{(i)} \tag{4}$$

Now using one of the gradient descent family optimizers, the model finds the parameters Θ that minimize the cost function. In fact, these parameters are the filters and other types of learnable variables (28).

Experiments and results

The designed synthetic data generator is capable of generating any number of images with any essential dimension. A total of 80 pixels for both height and width are chosen. In total, 24,000 images are included in training the proposed DCNN model. **Figure 5** (1&2) shows the train and validation loss/accuracy over 200 epochs.

As seen in **Figure 6** around epoch number 200, the model almost converges, and validation loss and accuracy are in an acceptable situation in terms of overfitting. To test the dataset, corresponding classes from the GTSRB are used. In the machine learning field and especially in supervised machine learning, the confusion matrix is considered one of the significant visualization methods for statistical classification tasks (29).

The confusion matrix for our proposed model on the test dataset is depicted in **Figure 7**. Classes that are more close to each other are confused with similar classes.

By referring to the plotted confusion matrix in **Figure 7**(1&2), it is clear that predicting classes 3 and 4 leads to higher errors than others. The first reason refers to the appearance of the two mentioned classes. They are very close to each other. The second reason refers to the GTSRB image size and aspect ratio. Some of the images of this dataset are very small in size and also are non-uniform in height and width ratio, while the generated train dataset is entirely square in size (80 × 80 pixels).

On the GTSRB website, recent benchmark competition results can be observed. Some of these results, close to this work result, are listed in **Table 1**. The main characteristic of the proposed method with other listed methods on the GTSRB website is the training dataset type. Most of them used GTSRB's own training dataset; however, in this work, the synthetic dataset is generated and used to train the model. Some real-life datasets, such as GTSRB, are biased in terms of distribution among classes; nevertheless, our dataset was evenly distributed (2,000 images per class). This may affect the decision-making results. Of course, sometimes this could be intentional because the distribution over classes is not even in real-life situations. For example, the number of priority signs in the city is normally much more than the number of roundabout signage. The obtained results show that the DCNN gives the best results among other image classification methods. Notably, DCNN architecture shows more than 91.91% accuracy in the GTSRB dataset with no view of any real traffic sign image.

Conclusion

Deploying machine learning in industry-level production is required to provide an exclusive dataset that meets the requirements. Providing labeled ground-truth datasets for computer vision tasks is usually expensive, time-consuming, and labor-intensive. Furthermore, there are some cases that create a real dataset that is not safe or practically impossible. Using CAD models is another option, but creating desired models one by one in most cases becomes more expensive than providing real datasets.

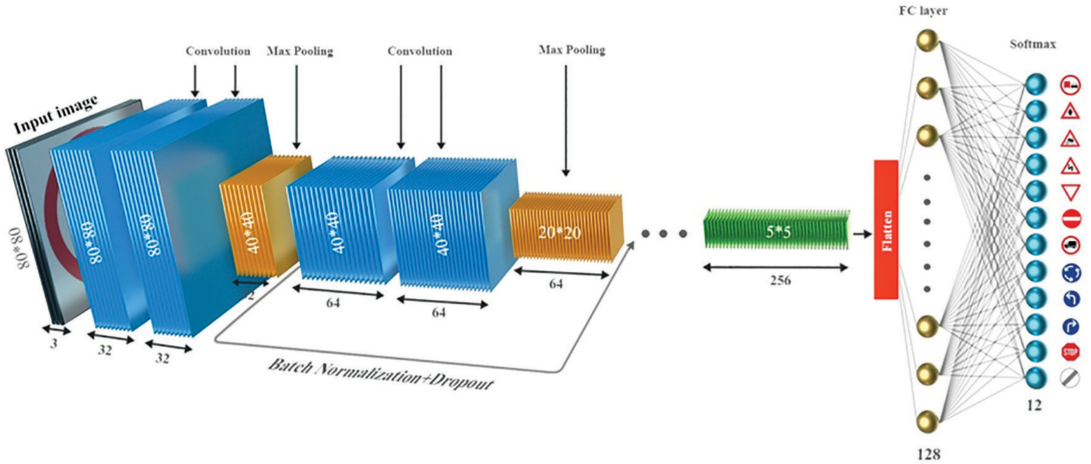

FIGURE 5 | Schematic diagrams of proposed model layers. This architecture is comprised of convolution, pooling, batch normalization, dropouts, and fully connected layers. Input images have 80 pixels for both height and width.

To cover the challenge in this paper and develop a synthetic dataset for the traffic sign recognition task, a procedural method was used. By using computer graphic tools, the proposed method facilitates generating numerous images that are precisely analogous to real-life instances. Moreover, a well-structured DCNN architecture was set up that decently fulfilled the classification task. Without seeing any real data, this classifier could categorize the real-world GTSRB dataset with more than 91.91% accuracy.

The provided dataset has more details than the requirements of the GTSRB dataset. We took many details into account that might not be necessary, but it made the classifier more reliable for complicated situations. Rendered images and real pictures captured by a camera intrinsically contain many dissimilarities. Using synthetic images to train machine learning models requires narrowing this similarity gap. Augmentation and other image processing filters are helpful in enhancing accuracy. Additionally, without augmentation and dropout, overfitting and generalization issues would be bold. For the next research, we decide to utilize this procedure for more complicated tasks like

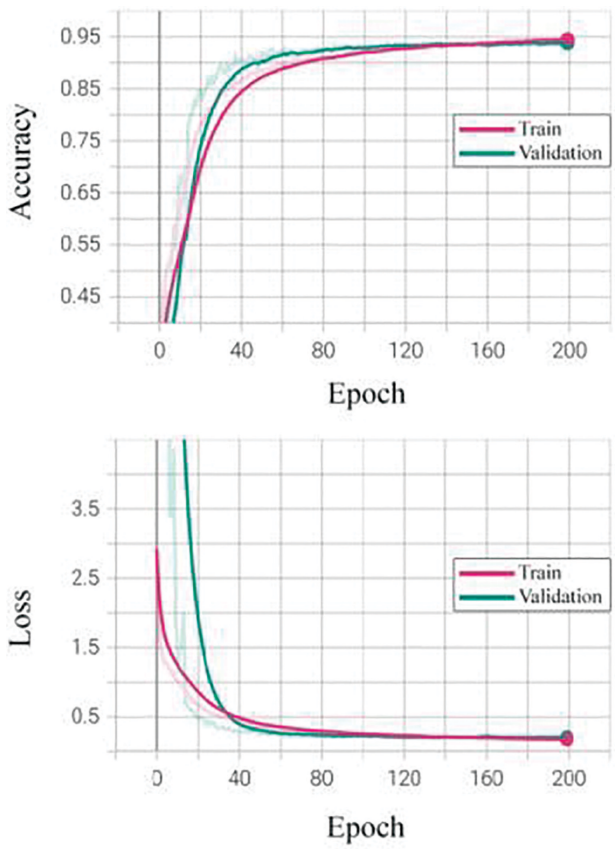

FIGURE 6 | Model almost convergence and validation, loss, and accuracy.

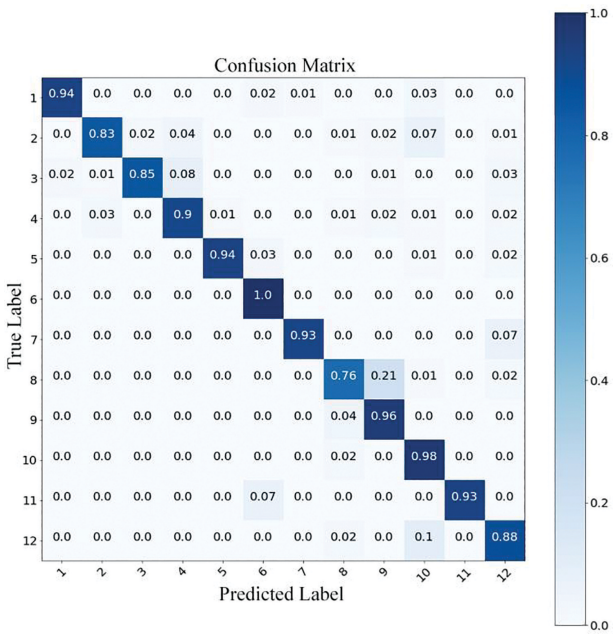

FIGURE 7 | Normalized confusion matrix for the German Traffic Sign Recognition Benchmark (GTSRB) dataset.

TABLE 1 | Comparison of the proposed method with other methods according to the German Traffic Sign Recognition Benchmark (GTSRB) benchmark (30).

#	Team	Method	Accuracy %
...	...	...	...
72	Italian-crash	Multi dataset algorithm	83.08
12	TDC	CVOG + CCV + NN (Team 2)	82.67
11	TDC	CVOG + CCV + NN (Team 1)	82.37
#	Our method	Synthetic data + DCNN	**91.91**
74	TDC	CVOG + ANN (Team 3)	81.80
97	RMULG	Subwindows + ETGRAY + LIBLINEAR	79.71
134	olbustosa	HOG_SVM	76.35
...	...	...	...

road and street object detection. Clearly, such a procedure requires higher attempts to set up a system that can provide credible rendered images.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Author contributions

RP conducted an initial literature review and data collection, performed the experiments, prepared the results, and drafted the manuscript. AN helped in writing–editing and conceptualization, analyzed the result, and contributed to supervision. Both authors read and approved the final manuscript.

References

1. Li L, Huang W, Liu Y, Zheng N, Wang F. Intelligence testing for autonomous vehicles: a new approach. *IEEE Trans Intell Vehic.* (2016). 1:158–66. doi: 10.1109/TIV.2016.2608003
2. Gidado UM, Chiroma H, Aljojo N, Abubakar S, Popoola SI, Al-Garadi MA. A survey on deep learning for steering angle prediction in autonomous vehicles. (2020) *IEEE Access.* (2020) 8:163797–817. doi: 10.1109/ACCESS.2020.3017883
3. Arnold E, Al-Jarrah OY, Dianati M, Fallah S, Oxtoby D, Mouzakitis A. A survey on 3D object detection methods for autonomous driving applications. *IEEE Trans Intell Trans Syst.* (2019) 20:3782–95. doi: 10.1109/TITS.2019.2892405
4. Gjoreski H, Ciliberto M, Wang L, Morales FJO, Mekki S, Valentin S, et al. The university of sussex-Huawei locomotion and transportation dataset for multimodal analytics with mobile devices. *IEEE Access.* (2018) 6:42592–604. doi: 10.1109/ACCESS.2018.2858933
5. Wang T, Wu DJ, Coates A, Ng AY. . End-to-End Text Recognition with Convolutional Neural Networks. *Proceedings of the 21st International Conference on Pattern Recognition*. Tsukuba (2012). p. 3304–8.
6. Dosovitskiy A, Fischer P, Ilg E, Hausser P, Hazırbas C, Golkov V. FlowNet: learning Optical Flow with Convolutional Networks. *Proceedings of the IEEE International Conference on Computer Vision*. Santiago: IEEE (2015). p. 2758–66. doi: 10.1109/ICCV.2015.316
7. Mayer N, Ilg E, Hausser P, Fischer P. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway, NJ: IEEE (2016). p. 4040–8. doi: 10.1109/CVPR.2016.438
8. Ros G, Sellart L, Materzynska J, Vazquez D, Lopez AM. The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition*. (2016). p. 3234–43. doi: 10.1109/CVPR.2016.352
9. Handa A, Patraucean V, Badrinarayanan V, Stent S, Cipolla R. Understanding Real World Indoor Scenes with Synthetic Data. *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway, NJ: IEEE (2016). p. 4077–85. doi: 10.1109/CVPR.2016.442
10. Tsai C, Tsai S, Hsu Y, Wu Y. Synthetic Training of Deep CNN for 3D Hand Gesture Identification. *Proceedings - 2017 International Conference on Control, Artificial Intelligence, Robotics and Optimization, ICCAIRO*. Piscataway, NJ: IEEE (2017). p. 165–70. doi: /10.1109/ICCAIRO.2017.40
11. Maldonado-Bascon S, Lafuente-Arroyo S, Gil-Jimenez P, Gomez-Moreno H, Lopez- Ferreras F. Road-sign detection and recognition based on support vector machines. *IEEE Trans Intell Trans Syst.* (2007) 8:264–78. doi: 10.1109/TITS.2007.895311
12. 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 Prob Eng.* (2015) 2015:1–11. doi: 10.1155/2015/250461
13. Kuang X, Fu W, Yang L. Real-time detection and recognition of road traffic signs using MSER and random forests. *Int J Online Eng.* (2018) 14:34–51. doi: 10.3991/ijoe.v14i03.7925
14. Ellahyani A, Ansari ME, Jafari IE. Traffic sign detection and recognition based on random forests. *Appl Soft Comput.* (2016) 46:805–15. doi: 10.1016/j.asoc.2015.12.041
15. Liang Z, Shao J, Zhang D, Gao L. Traffic sign detection and recognition based on pyramidal convolutional networks. *Neural Comput Appl.* (2019) 32:6533–43. doi: 10.1007/s00521-019-04086-z
16. Shustanov A, Yakimov P. CNN design for real-time traffic sign recognition. Proc Eng. (2017) 201:718–25. doi: 10.1016/j.proeng.2017.09.594
17. Jaderberg M, Simonyan K, Vedaldi A, Zisserman A. Synthetic data and artificial neural networks for natural scene text recognition. arXiv [Preprint]. (2014) doi: 10.48550/arXiv.1406.2227
18. Peng X, Sun B, Ali K, Saenko K. Learning Deep Object Detectors from 3D Models. *Proceedings of the IEEE International Conference on Computer Vision*. Piscataway, NJ: IEEE (2015). p. 1278–86. doi: 10.1109/ICCV.2015.151
19. Sun B, Saenko K. From virtual to reality: fast adaptation of virtual object detectors to real domains. *Proc Br Mach Vis Conf.* (2014). doi: 10.5244/C.28.82
20. Su H, Qi CR, Li Y, Guibas LJ. Render for CNN: Viewpoint Estimation in Images using CNNs Trained with Rendered 3D Model Views. *Proceeding of the IEEE International Conference on Computer Vision*. Piscataway, NJ: IEEE (2015). p. 2686-2694. doi: 10.1109/ICCV.2015.308
21. Satilmis P, Bashford-Rogers T, Chalmers A, Debattista K. A machine-learning-driven sky model. *IEEE Comput Grap Appl.* (2017) 37:80–91. doi: 10.1109/MCG.2016.67
22. Jung J, Lee JY, Kweon IS. One-day outdoor photometric stereo using skylight estimation. *Int J Comput Vis.* (2019) 127:1126–42. doi: 10.1007/s11263-018-01145-1

23. Bilal A, Jourabloo A, Ye M, Liu X, Ren L. Do convolutional neural networks learn class hierarchy?. *IEEE Trans Vis Comput Grap.* (2018) 24:152–62. doi: 10.1109/TVCG.2017.2744683

24. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. *Proc IEEE.* (1998) 86:278–324. doi: 10.1109/5.726791

25. Bjorck J, Gomes C, Selman B, Weinberger KQ. Understanding batch normalization. *Adv Neural Inform Process Syst.* (2018) 7694–705.

26. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. *Commun ACM.* (2017) 60:84–90. doi: 10.1145/3065386

27. Zhang Z, Wang H, Liu S, Xiao B. Consecutive convolutional activations for scene character recognition *IEEE Access.* (2018) 6:35734–42. doi: 10.1109/ACCESS.2018.2848930

28. Géron A. *Hands-on machine learning with scikit-learn, keras, and tensorflow: concepts, tools, and techniques to build intelligent systems*. 2th ed. Sebastopol, CA: O'Reilly Media (2019).

29. Stehman SV. Selecting and interpreting measures of thematic classification accuracy. *Remote Sens Environ.* (1997) 62:77–89. doi: 10.1016/S0034-4257(97)00083-7

30. INI. *German traffic sign benchmarks*. (2019). Available online at: https://benchmark.ini.rub.de/gtsrb_results_ijcnn.html.