Traffic Sign Recognition Using CNN

Wajeh E. Elside, Ahmed J. Abougarair*

Electrical and Electronics Engineering, University of Tripoli, Tripoli, Libya.

* Corresponding author. Tel: (+218)916094184; Email: a.abougarair@uot.edu.ly (A.J.A.)
Manuscript submitted November 20, 2024; revised November 29, 2024; accepted December 25, 2024; published January 17, 2025.
doi: 10.18178/JAAI.2025.3.1.19-39

Abstract: Traffic Sign Recognition (TSR) is a crucial component of intelligent transportation systems, aiming to enhance road safety and support autonomous vehicle navigation. This paper focuses on developing a traffic sign recognition system using Convolutional Neural Networks (CNNs). The system employs two distinct models: a custom sequential CNN and a pre-trained VGG19 model. Both models were trained on the German Traffic Sign Recognition Benchmark (GTSRB) dataset, which comprises 43 classes of traffic signs, over 30 epochs. The paper investigated two training scenarios. In the first case, both models were trained with a batch size of 8 and a learning rate of 0.001. In the second case, the batch size was increased to 16, and the learning rate was decreased to 0.0001. The models were evaluated based on various metrics, including accuracy, validation curves, test accuracy, confusion matrix, F1-Score, precision, and recall. Results showed that the second case (batch size of 16 and learning rate of 0.0001) yielded better overall performance, particularly in test accuracy. In this scenario, the sequential model achieved a training accuracy of 99.77% and a validation accuracy of 97.48%. The VGG19 model outperformed the sequential model, achieving a training accuracy of 99.94% and a validation accuracy of 98.46%. The results of this paper contribute to the advancement of traffic sign recognition systems, supporting their implementation in real-world intelligent transportation and autonomous driving applications.

Keywords: Convolutional Neural Network (CNN), German Traffic Sign Recognition Benchmark (GTSRB), German Traffic Sign Recognition Benchmark (GTSRB), VGG

1. Introduction

TSR is a critical component of modern transportation systems, aimed at enhancing road safety and efficiency. Traffic signs serve as visual cues that communicate essential information to drivers, such as speed limits, warnings, and regulatory instructions. However, interpreting and recognizing these signs in diverse environmental conditions and from varying perspectives pose significant challenges for automated systems [1, 2]. Traffic signs exhibit standardized shapes, colors, and symbols designed to convey specific messages swiftly and effectively to drivers. They encompass a wide array of visual cues, including geometric shapes, textual information, and symbolic representations, each tailored to communicate distinct instructions and warnings [1]. Unlike natural languages, traffic signs are universally standardized across regions and countries, ensuring consistency in communication regardless of linguistic or cultural differences. Computer vision technology employs deep learning models, such as Sequential Convolutional Neural Network (CNN) and VGG 19 architectures, to tackle the challenge of traffic sign recognition. These sophisticated models are designed to identify and categorize traffic signs by analyzing their unique visual characteristics. This research explores the application of these CNN models in creating reliable systems that can accurately detect traffic signs across diverse scenarios. The main objectives of the paper are as follows:

develop a system to accurately recognize and classify different traffic signs, aiming for high recognition accuracy. Design and implement deep learning models, specifically multi-dimensional Convolutional Neural Network (CNN) models, for traffic signs recognition. Conduct simulations and training procedures to optimize the performance of the deep learning models, focusing on accuracy and robustness. Evaluate and compare the performance of the two deep learning models for traffic signs recognition, analyzing their strengths and weaknesses [3–5].

2. Convolutional Neural Networks (CNNs)

CNNs have shown significant potential in practical applications. The word "convolutional" describes how convolution, a particular kind of linear process, is used in the network. CNNs as shown in Fig. 1 can be separated from conventional neural networks by this, as the latter mostly use matrix multiplication. Convolution, which takes the place of matrix multiplication in at least one layer of CNNs, allows for more efficient processing of grid-like data [6, 7]. It is commonly believed that features of CNNs shouldn't be dependent on spatial location while solving issues. In other words, one does not need to focus on the precise locations of the faces in the images while using a face detection algorithm. Anywhere they may be in the supplied photos, the only thing to worry about is where to discover them. Developing abstract qualities as input advances into deeper levels is another essential CNN property. When categorizing photographs, for example, the edge may be in the top layers, then simpler shapes in the layers below it, and finally higher-level traits in the layers beyond it [8, 9].

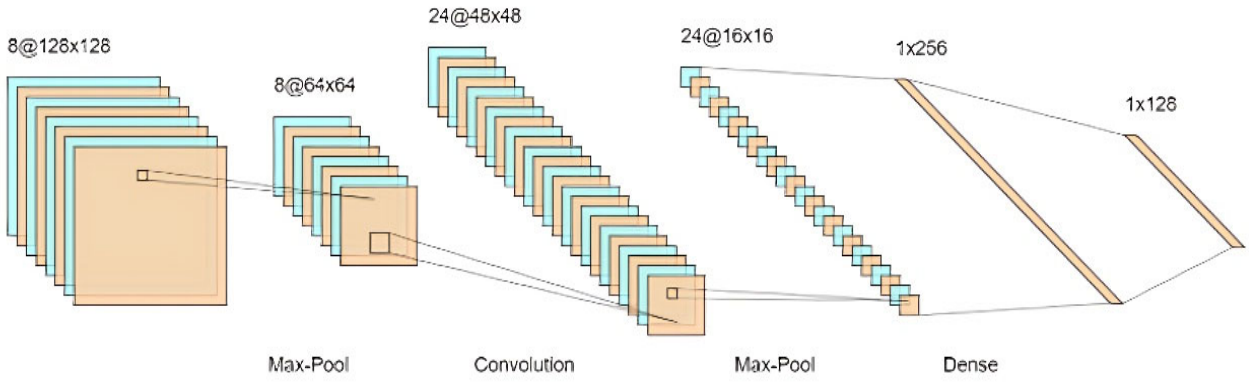

Fig. 1. A CNN with five hidden layers.

2.1. Convolutional Neural Networks Layers

There are many possible architectures for a CNN. most of them typically contain three primary types of layers, which are the convolution layers, pooling layers and fully connected layers. However, some CNN architectures don't contain all of them, an example of such a case is the U-net CNN architecture which doesn't contain the fully connected layer. Each type of the three primary types of layers will be explained below [10, 11].

Convolution Layers: The convolutional layer forms the fundamental building block of a CNN's structure. This layer consists of a collection of filters, also known as kernels. These specialized filters are designed to process N-dimensional input data, typically in the form of images. As the filters are applied across the input, they generate an output known as a feature map, which captures essential characteristics of the original data. This process, detailed earlier, forms the basis of how CNNs extract and learn meaningful features from visual information.

Pooling Layers: pooling layer's main role is to downsample the feature maps that result from convolutional

operations. This downsampling process shrinks larger feature maps into more compact versions while preserving the most crucial information or features at each step.

Fully Connected Layers: Typically positioned at the conclusion of a CNN's structure, fully connected layers derive their name from their comprehensive connectivity: each neuron in these layers' links to every neuron in the preceding layer. Within the CNN framework, fully connected layers primarily function as classifiers while also extracting high-level features. The initial fully connected layer generally receives input in the form of a flattened, one-dimensional vector, which is created from the feature maps produced by the preceding convolutional and pooling layers. The output from these fully connected layers represents the CNN's final predictions, effectively translating the learned features into meaningful classifications or outcomes.

2.2 Evaluation Matrix of CNN

Convolutional Neural Networks (CNNs) can be evaluated using various metrics depending on the specific task they are designed for. Here are some common evaluation metrics used for CNNs [12–15]:

2.2.1. Classification tasks

For image classification tasks, the following metrics are often used:

1. Accuracy: The proportion of correct predictions among the total number of cases examined
2. Precision: The ratio of true positive predictions to the total positive predictions.
3. Recall (Sensitivity): The ratio of true positive predictions to the total actual positive cases.
4. F1-score: The harmonic mean of precision and recall, providing a balanced measure of the model's performance.
5. Area Under the Receiver Operating Characteristic (AUC-ROC) curve: A plot of the true positive rate against the false positive rate at various threshold settings.

2.2.2. General performance metrics

Some general performance metrics that apply to various CNN tasks include:

1. Loss Function: The optimization objective used during training, such as cross-entropy loss for classification or mean squared error for regression.
2. Training and Validation Accuracy/Loss: These metrics help monitor the model's performance during training and detect overfitting.
3. Confusion Matrix: A table that describes the performance of a classification model on a set of test data for which the true values are known.

Testing the models to ensure their ability to accurately identify and categorize images is crucial for validation. This step verifies that the models have effectively learned the intrinsic patterns and features required to correctly associate each image with its designated sign or category. Such validation is indispensable for instilling confidence in the models' performance in real-world applications. The test dataset, comprising 7,300 diverse images distributed across various classes, serves as a comprehensive and demanding benchmark. Successfully classifying these images showcases the models' resilience and underscores their potential as dependable tools for diverse image recognition challenges. It is significant to save the trained model so that it can be loaded later in the testing phase without the need of training all over again.

3. Methodology

In this section, the methodology and implementation details of the Traffic Signs Recognition (TSR) paper will be presented. The focus of this section is to provide a comprehensive description of the steps involved in developing the TSR system using multiple models [16–20]. The flow chart presented in Fig. 2 outlines the

process.

The main objective of this section is to offer a detailed explanation of each block in the flow chart, allowing for a clear understanding of the paper's methodology. By explaining each step individually, the underlying processes and techniques employed to achieve accurate Traffic Signs recognition will be elucidated.

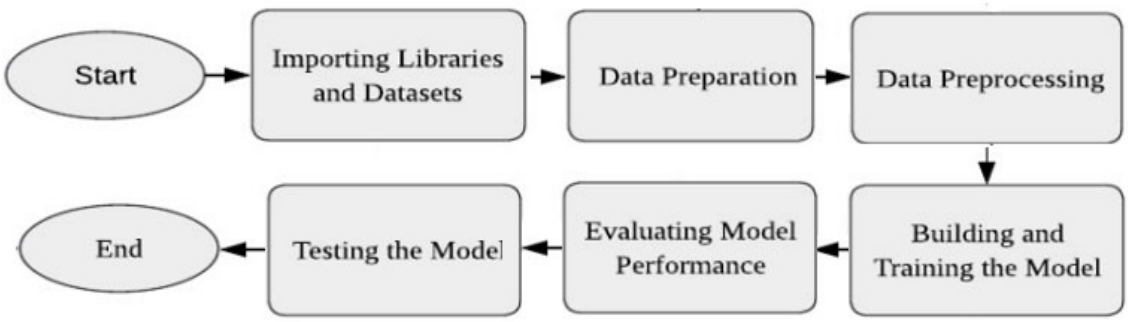

Fig. 2. Stages of system implementation.

3.1. Dataset Preparation

The GTSRB dataset was utilized for traffic sign recognition training, with 43 classes representing different traffic signs. The dataset was divided into 39209 training photos and 12630 testing images, which were zoomed-in traffic signs with varying lighting and backgrounds [14]. The dataset photos have various resolutions, including 29×27 and 145×131. Fig. 3 shows samples from the dataset. Fig. 4 shows a name and image for each of 43 classes. Fig. 5 explains the distribution of the images used in training the model across the 43 classes of the dataset [21–26].

Fig. 3. Five samples from the GTSRB dataset.

1. Training Set (approximately 90% of the data):

- Used to evaluate the model's performance during the training process.
- Helps in fine-tuning the model and its hyperparameters.
- Assists in preventing overfitting by providing a measure of how well the model generalizes to unseen data.
- Can be used to determine when to stop training (early stopping) if the model's performance on the validation set starts to degrade.

2. Test Set (approximately 10% of the data):

- Kept completely separate and unused until the final evaluation.
- Provides an unbiased assessment of the model's performance on truly unseen data.
- Used to estimate how well the model will perform in real-world scenarios.

This split methodology is crucial in machine learning for several reasons:

1. It helps prevent overfitting, where a model performs well on training data but poorly on new, unseen data.

2. It allows for proper model evaluation and comparison of different models or approaches.
3. It simulates real-world scenarios where the model will be applied to new, unseen data.

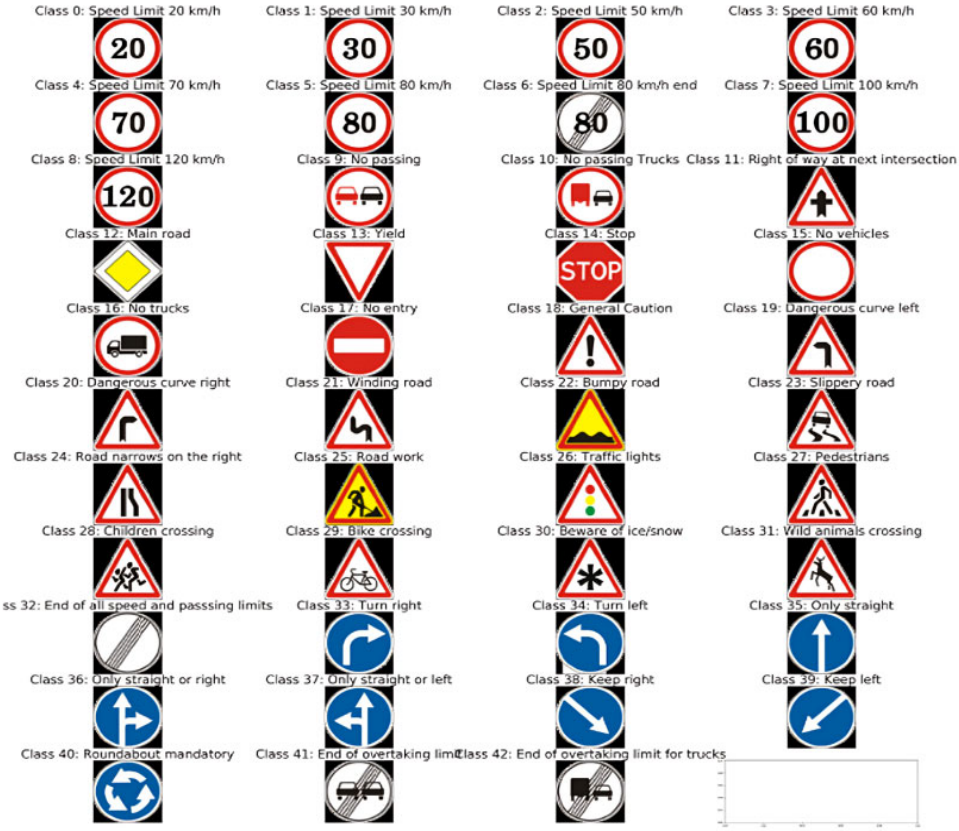

Fig. 4. The mapping for each sign image to its corresponding name for the 43 classes in the GTSRB.

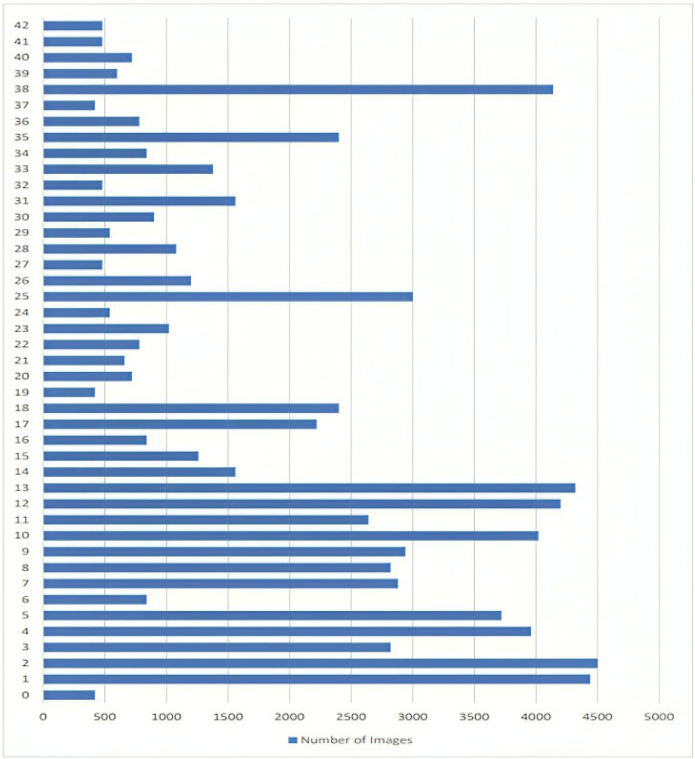

Fig. 5. The number of images within each class of the 43 classes contained in training portion of the GTSRB dataset Validation Set (approximately 10% of the data).

3.1.1. Shuffling

Shuffling the dataset is a pivotal practice in machine learning, particularly during the training phase. It serves a critical role in ensuring that data is evenly and randomly distributed across the training, validation, and test sets. This randomization is essential as it helps mitigate the influence of any inherent biases or patterns within the data, thereby enhancing the model's performance and reliability. The primary objective of shuffling is to generate a representative subset from the entire dataset, thereby optimizing the training process. By randomizing the data, we guarantee that each subset which are the training, validation, or test sets maintains a balanced distribution of different classes or labels. This balance is crucial for training a model that can generalize effectively and make precise predictions when exposed to new, unseen data. Moreover, it promotes robustness by preventing the model from overfitting to specific patterns in the dataset, ensuring it learns the underlying relationships more comprehensively. Thus, shuffling not only improves the training efficiency but also fosters a more accurate and adaptable machine learning model overall [27-32].

3.1.2. Rescaling

The dataset's images underwent normalization to the range [0, 1] by dividing each pixel value by 255. This rescaling procedure ensures that all pixel intensities are uniformly scaled down to a normalized range. Dividing by 255 transforms the original pixel range of [0, 255] into [0, 1], where each increment represents a step size of 1/255. This normalization is crucial as it prevents any single pixel feature (such as a very bright or very dark pixel) from disproportionately influencing the learning process due to its larger numerical value. Additionally, it aligns well with certain activation functions like sigmoid or SoftMax, which are more responsive to input values within the [0, 1] range. By maintaining pixel values within this standardized range, the input to these activation functions remains optimal for achieving consistent and effective model performance. Moreover, rescaling to [0, 1] simplifies comparisons between images and eases the optimization process during training, facilitating convergence to better-performing models. This normalization step thus enhances both the stability and efficiency of the learning process across various machine learning tasks involving image data [33–38].

3.2. Building the Model

This study utilized various neural network architectures implemented through the Keras framework. Specifically, two types of models were employed: a custom sequential model and a pre-existing model known as VGG19. To optimize the performance of these models, the Adam algorithm was applied for adjusting the weights and biases of the network parameters during training.

3.2.1. Sequential model architecture

The architecture typically begins with convolutional layers, which use learnable filters to extract features like edges and textures from the input images. These filters move across the input, performing dot product calculations. Each convolutional layer is usually followed by an activation function, commonly ReLU, to introduce non-linearity into the model. After the convolutional layers, pooling layers are often employed to reduce the spatial dimensions of the feature maps. This downsampling process helps maintain important information while decreasing the number of parameters and computational requirements. Next, a flattening layer transforms the 2D matrices of pooled feature maps into a 1D vector. This prepares the data for input into fully connected (dense) layers, which are traditional neural network layers where each neuron connects to every neuron in the previous layer. These dense layers learn global patterns and relationships from the features extracted earlier in the network. The final layer of a Sequential CNN is typically a dense layer with an activation function appropriate for the specific task, such as SoftMax for classification problems. This layer

produces the model's predictions based on the learned features. This sequential structure allows the model to progressively learn and refine features from low-level details to high-level abstractions, making it particularly effective for image processing tasks [13, 33].

3.2.2. VGG 19 architecture

A pre-trained model has undergone training on a big dataset for a specific job, such as image classification. These models are often trained on high-performance hardware with large amounts of data and processing resources. Pre-trained models have fixed architectures, which means the layer structure and arrangement cannot be changed. Popular pre-trained models, such as ResNet, VGG, and Inception, have predefined architectures and connections [17, 30]. VGG 19, developed by the Visual Geometry Group at Oxford University, is an advanced Convolutional Neural Network (CNN) architecture renowned for its depth and performance in computer vision tasks. The "19" in VGG 19 refers to its structure comprising 19 layers, characterized by 16 convolutional layers followed by ReLU activation functions, and interspersed with max pooling layers for spatial down sampling. These convolutional layers are adept at capturing intricate spatial features from input images, gradually learning hierarchical representations of visual information. In addition to its convolutional layers, VGG 19 includes three fully connected layers at the end of the network. These dense layers assimilate the high-level features extracted by the convolutional layers, enabling the model to make accurate predictions for tasks such as image classification and object recognition. Initially trained on the ImageNet dataset, which includes a vast array of images across multiple categories, VGG 19 achieves impressive performance metrics, demonstrating its robustness and effectiveness in handling complex visual data.

4. Simulation Results

This segment offers an in-depth examination and interpretation of the outcomes derived from various traffic sign recognition models, each employing distinct methodological approaches. The effectiveness of the proposed CNN-based systems in identifying and categorizing traffic signs is thoroughly evaluated. To gauge the models' performance, key metrics including accuracy, precision, recall, and F1-score are employed. Furthermore, test confusion matrices are utilized to assess the models' strengths and pinpoint potential areas for enhancement. All the simulation results prepared by Matlab [39–56].

4.1. The Sequential CNN Model

This Sequential CNN model architecture is configured with multiple layers tailored for effective TSR. The model initiates with Conv2D layers, each applying a set of 3×3 convolutional filters to input images, progressively extracting diverse features like edges and textures. The subsequent MaxPooling2D layers reduce spatial dimensions by halving width and height, facilitating efficient computation while retaining essential information. Batch-Normalization layers normalize activations across each batch, enhancing stability and convergence speed during training. Dropout layers are strategically placed to prevent overfitting by randomly dropping units during training, promoting robust model generalization. The Flatten layer reshapes the 3D output into a 1D vector, preparing it for processing by Dense layers. Dense layers then perform classification tasks, with the final layer outputting predictions tailored to the specific task, such as identifying objects in images or recognizing patterns in data. Each layer's output shape and parameter count are meticulously tuned to optimize performance and ensure effective learning throughout the network. Shown below is Table 1 which shows the layers design for this model.

Table 1. The Sequential Model Architecture for the Traffic Sign Recognition Task

Layer Type	Output Shape	Parameters
Convd2d (conv2D)	(None, 46, 46, 16)	1216
Convd2d_1 (conv2D)	(None, 42, 42, 32)	12832
Max_pooling2d(Maxpooling2d)	(None, 21, 21, 32)	0
Batch normalization	(None, 21, 21, 32)	128
Convd2d_2 (Convd2D)	(None, 19, 19, 64)	18496
Convd2d_3 (Convd2D)	(None, 17, 17, 64)	36928
Max_pooling2d_1(Maxpooling2d)	(None, 8, 8, 64)	0
Batch normalization_1(batch)	(None, 8, 8, 64)	256
Dropout (Dropout	(None, 8, 8, 64)	0
Flatten(flatten)	(None, 4096)	0
Dense (Dense)	(None, 512)	2097664
Batch normalization_2(batch)	(None, 512)	2048
Dropout_1(Dropout)	(None, 512)	0
Dense_1(Dense)	(None, 43)	22059

4.2. VGG19 Model

This section introduces a pretrained model called VGG 19 for traffic sign recognition. Fine-tuning is the process of adapting pretrained models to specific tasks. These models are initially trained on large datasets to learn general features from various images. By fine-tuning them, the models can be customized to excel in traffic sign recognition. This involves adjusting the model's parameters or architecture while leveraging the pre-learned features.

The VGG 19 model, as depicted in the provided Table 2, showcases a deep convolutional neural network architecture optimized for intricate image recognition tasks. The model initiates with the "VGG19" functional layer, which outputs a tensor shape of (None, 1, 1, 512), indicating the dimensional transformation through successive convolutional and pooling layers. Following this, batch normalization is applied via "batch_normalization_8", ensuring stable and efficient gradient propagation during training with 2048 parameters. The subsequent "flatten_4" layer collapses the tensor into a 1D vector of 512 elements, preparing for dense layer processing. "dense_8" and "dense_9" represent the fully connected layers, with 512 and 43 units respectively, culminating in the final output layer for classification. Each layer's configuration is meticulously tuned to optimize model performance, balancing computational efficiency with robust feature extraction and classification accuracy.

Table 2. The Sequential Model Architecture for the Traffic Sign Recognition Task

Layer Type	Output Shape	Parameters
vgg19 (functional)	(None, 1, 1, 512)	20024384
batch_normalization_8	(None, 1, 1, 512)	2048
flatten_(flatten)	(None, 512)	0
dense_8 (Dense)	(None, 512)	262656
dense_9 (Dense)	(None, 512)	22059

The model architecture exhibits a total parameter count of 20,311,147 Among these parameters, 20,310,123 are trainable, representing the weights and biases that are learned during the training process to optimize the model's performance. These trainable parameters enable the model to adapt and specialize for the task at hand, such as recognizing Traffic Signs recognition. In contrast, there are 1,024 non-trainable parameters, which consist of fixed weights and biases that are not updated during training. These non-

trainable parameters are often associated with pre trained layers or frozen layers, providing architectural constraints or leveraging pre-existing knowledge. The distribution of trainable and non-trainable parameters within the model plays a crucial role in its complexity, capacity, and ability to learn from the training data.

4.3. Models Parameters

The parameters indicated in Table 3 have been homogenized across the three models, making their performances more comparable. By keeping the parameters consistent, the impact of model design and training procedures on their respective outcomes may be isolated. This provides a better understanding of the models' relative strengths and flaws.

Table 3. Model Parameters Summary

Parameters	Description
Metrics	Confusion matrix, Precision, Accuracy, Recall and F1-Score
Optimizer	Adam
Loss Function	categorical_crossentropy

Adam (Adaptive Moment Estimation) is an optimizer used in machine learning. It adjusts the parameters of a model during training to minimize the error or loss. It combines adaptive learning rates and momentum to efficiently update the parameters [22]. Other parameters were varied throughout the learning process in order to implement the optimum modal performance.

- **Number of epochs** represents the complete cycles through which the entire training dataset is processed, moving both forward and backward within the neural network during model training. Considered a critical hyperparameter, epochs define the total iterations the model undergoes to learn and extract meaningful patterns from the input data. Each epoch allows the model to progressively refine its understanding and improve its predictive capabilities by adjusting internal parameters based on the training dataset.
- **Batch size** represents the quantity of training samples simultaneously processed in a single computational iteration during neural network training. As an important hyperparameter, batch size significantly influences the learning dynamics and computational efficiency of the model. It determines the number of data points evaluated and used to update the network's weights in each training step, thereby affecting the model's convergence, memory usage, and overall performance characteristics.
- **Learning rate** is a crucial hyperparameter in machine learning algorithms that governs the magnitude of adjustments made to a model's parameters during optimization. It essentially dictates the size of steps taken when updating the model's internal variables. This hyperparameter plays a vital role in controlling the impact of each update on the model's learning process, influencing how quickly or slowly the model adapts to the training data. The learning rate's value can significantly affect the model's convergence speed and its ability to find optimal solutions.

4.4. Performance Analysis

The performance is evaluated using the training and validation dataset by measuring the models' accuracy, loss, precision, recall, F1-score. The developed models were examined for their performance in two study cases during training. The models' learning progress was assessed by monitoring validation and training losses, as well as accuracy metrics. The validation loss and accuracy were determined on a different dataset, whereas the training loss and accuracy were calculated on the training dataset.

4.4.1. First case

In the first case, the training process involved a training process with a common set of hyperparameters. This included a training duration of 30 epochs, a batch size of 8, and a learning rate of 0.001 for the two proposed models. To assess the learning progress of these models, the validation and training losses, as well as accuracy metrics, were closely monitored. Figs. 6 and 7 provide a visual representation of the accuracy and loss curves for each model during the 30 epochs. These figures illustrate the performance of the Sequential and VGG19 models, respectively.

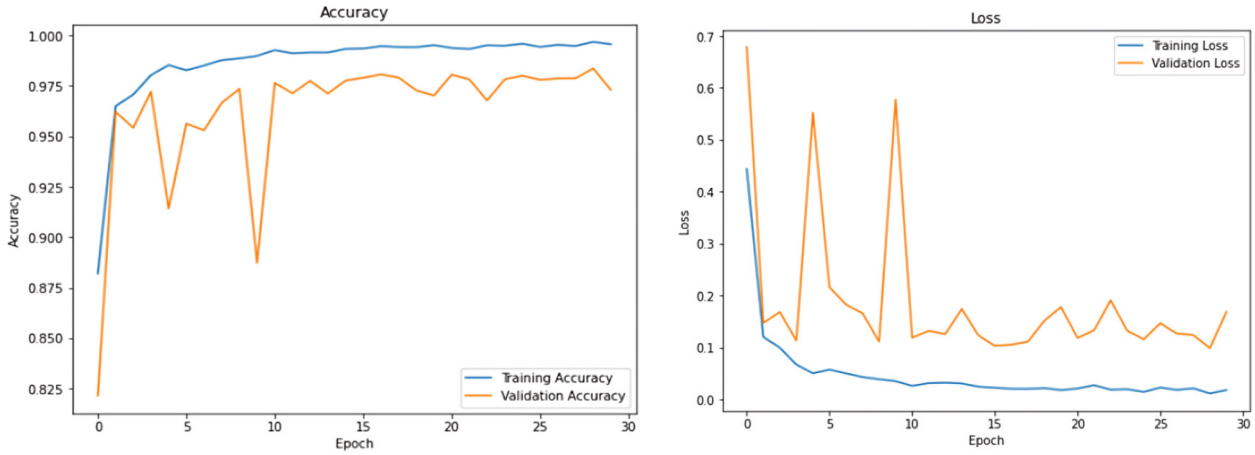

Fig. 6. Sequential model's accuracy and loss curves during 30 epochs.

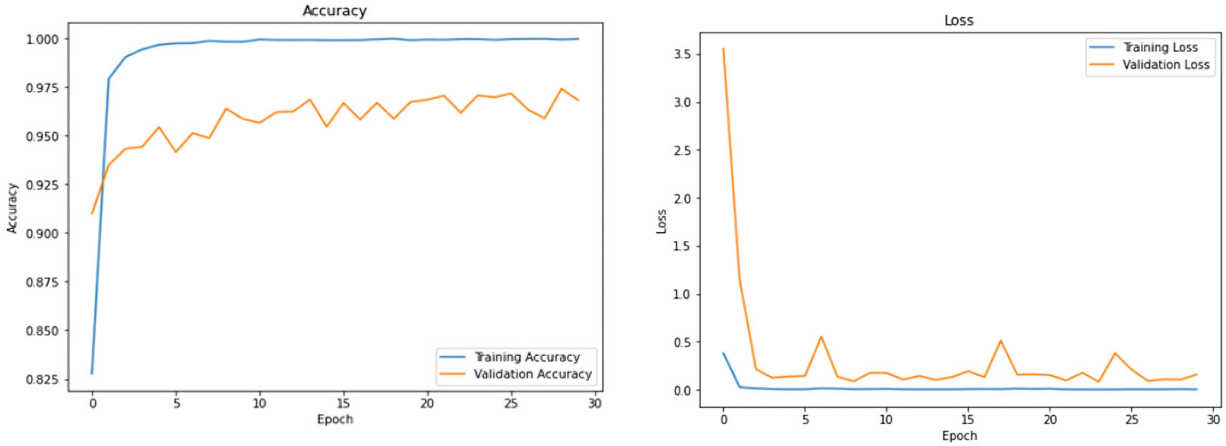

Fig. 7. VGG19 model's accuracy and loss curves during 30 epochs.

Training the Sequential model for TSR over 30 epochs with a batch size of 8 and a learning rate of 0.001 yields promising results as presented in Fig. 6. The model achieves a final training accuracy of 98.2% and a validation accuracy of 96.66%. The training loss reduces to 0.0382, and the validation loss decreases to 0.1687. The proximity between training and validation accuracies suggests effective generalization to new data, a crucial aspect of model performance. The model not only learns well from the training data but also demonstrates its ability to make accurate predictions on unseen validation data, indicating a successful training process.

Fig. 7 indicates the training process of the VGG19 model for TSR recognition over 30 epochs with a batch size of 8 and a learning rate of 0.001 demonstrates substantial progress. The model achieves a final training accuracy of 99.2% and a validation accuracy of 97.43%. The training loss decreases to 0.0127, and the

validation loss decreases to 0.1566. Notably, the training and validation accuracies are close to each other, indicating effective generalization. This is advantageous as it suggests that the model can make accurate predictions not only on the training data but also on new, unseen data.

Based on the training results, both models show excellent performance, but the VGG19 model appears to have a slightly higher final validation accuracy of 97.43% compared to the Sequential model's accuracy of 96.66%. Additionally, VGG19 exhibits a lower final validation loss of 0.1566 compared to the Sequential model's loss of 0.1687. Therefore, in this specific scenario, the VGG19 model appears to be the better-performing model, as it achieved higher accuracy and lower loss on the validation dataset after 30 epochs.

4.4.2. Second case

In the second case, an extended and modified training regimen was adopted with the aim of achieving improved model performance. Each of the three proposed models underwent an extended training period of 30 epochs, accompanied by a decreased batch size of 16, and a learning rate of 0.0001. This adjustment aimed to provide the models with more opportunities to learn and adapt to the data. The results of this comprehensive evaluation are vividly depicted in Figs. 8 and 9, showcasing the accuracy and loss curves for the Sequential and VGG19 models over the course of these 30 epochs.

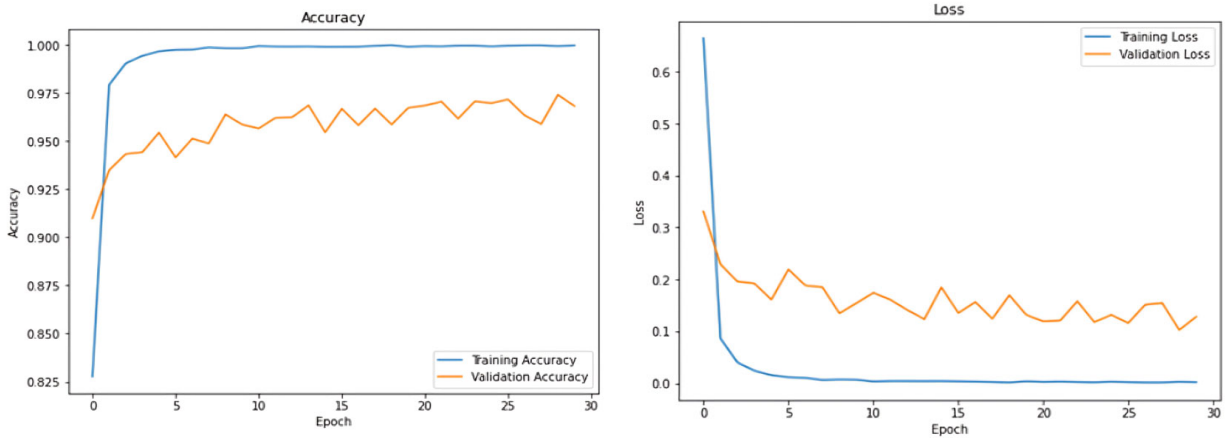

Fig. 8. The sequential model's accuracy and loss curves during 30 epochs.

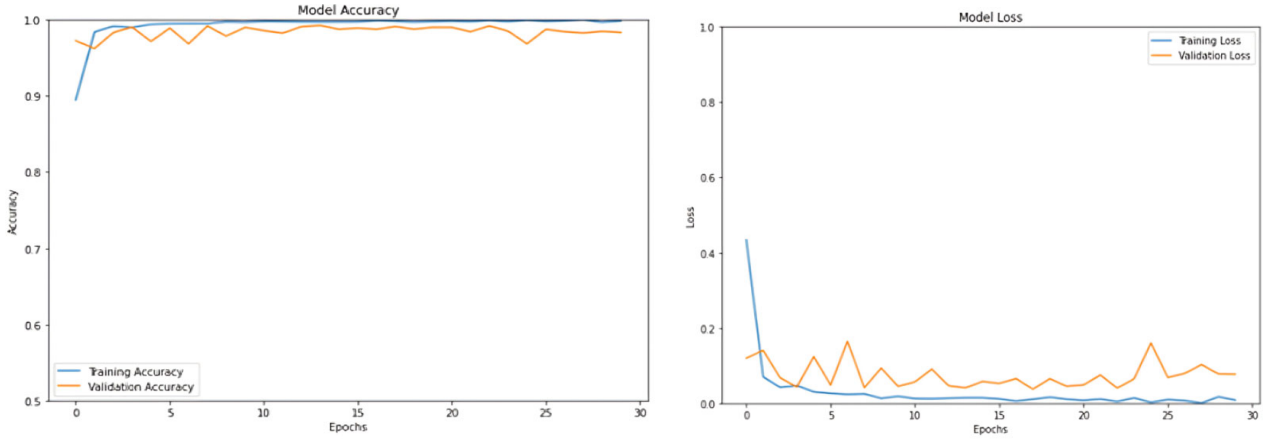

Fig. 9. VGG 19 model's accuracy and loss curves during 30 epochs.

Fig. 8 shows a further enhancement in model performance by extending the training to 30 epochs with a larger batch size of 16 and a lower learning rate of 0.0001. The Sequential model achieves a final training accuracy of 99.77%, while the validation accuracy peaks at 97.48%. The training loss drops to 0.0024, and

the validation loss diminishes to 0.1216. Comparing the two cases of the sequential model, the second case results in a more refined model. The higher accuracy and lower loss values indicate improved learning and robustness. The close alignment between training and validation accuracies in both cases reinforces the model's ability to generalize effectively, making it well-suited for accurate TSR on previously unseen data.

As shown in Fig. 9, the training process of the VGG19 model for TSR over 30 epochs with a batch size of 16 and a learning rate of 0.0001 demonstrates substantial progress. The model achieves a final training accuracy of 99.94% and a validation accuracy of 98.46%. The training loss decreases to 0.00267, and the validation loss decreases to 0.0892. Comparing the two cases of the VGG19 model, the second case results in a more refined model. The higher accuracy and lower loss numbers indicate improved learning and robustness. The close alignment of training and validation accuracies highlights the model's capacity to generalize effectively, making it ideal for accurate TSR on previously unknown data.

When comparing the two cases, it is evident that the second case, with a batch size of 16 and a learning rate of 0.0001, generally results in better model performance for both the Sequential and VGG19 models. The Sequential model in the second case achieves a higher training accuracy of 99.77% and a higher validation accuracy of 97.48%, compared to 98.2% and 96.66%, respectively, in the first case. Similarly, the VGG19 model in the second case reaches a training accuracy of 99.94% and a validation accuracy of 98.46%, which are improvements over the 99.2% training accuracy and 97.43% validation accuracy in the first case. Additionally, both models in the second case exhibit lower training and validation losses, indicating more efficient learning and better generalization. These results suggest that the adjustment to a lower learning rate and larger batch size in the second case provided the models with better learning conditions, leading to improved performance on the traffic sign recognition task.

4.4.3. Confusion matrix

The evaluation of the confusion matrix, a powerful tool that unveils the correspondence between true labels and predicted labels, has been thoroughly conducted on the test dataset for all three models, as applied in the two case scenarios previously presented. This careful examination on the test dataset provides us with a robust understanding of the models' performance in real-world, unseen situations, highlighting their proficiency in correctly classifying data previously unseen. Figs. 10 and 11 showcases the confusion matrix of the Sequential and VGG 19.

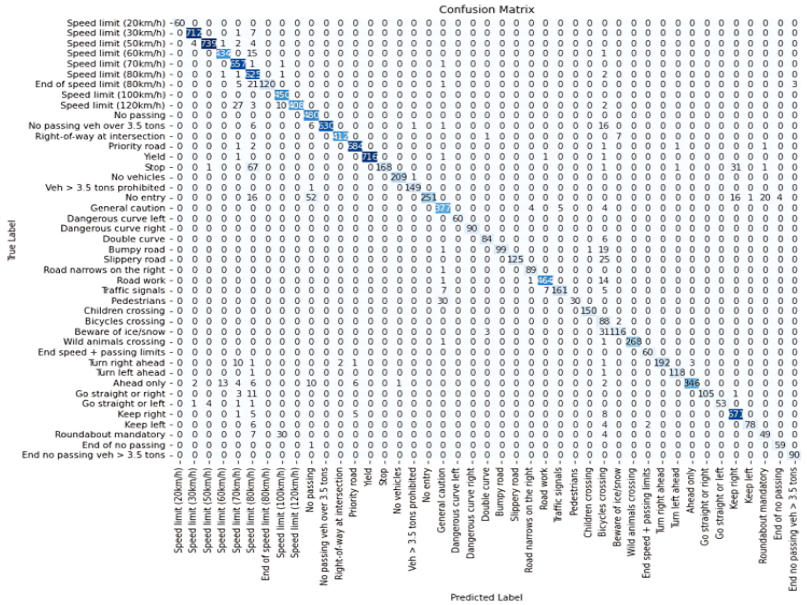

Fig. 10. Confusion matrix of the sequential model.

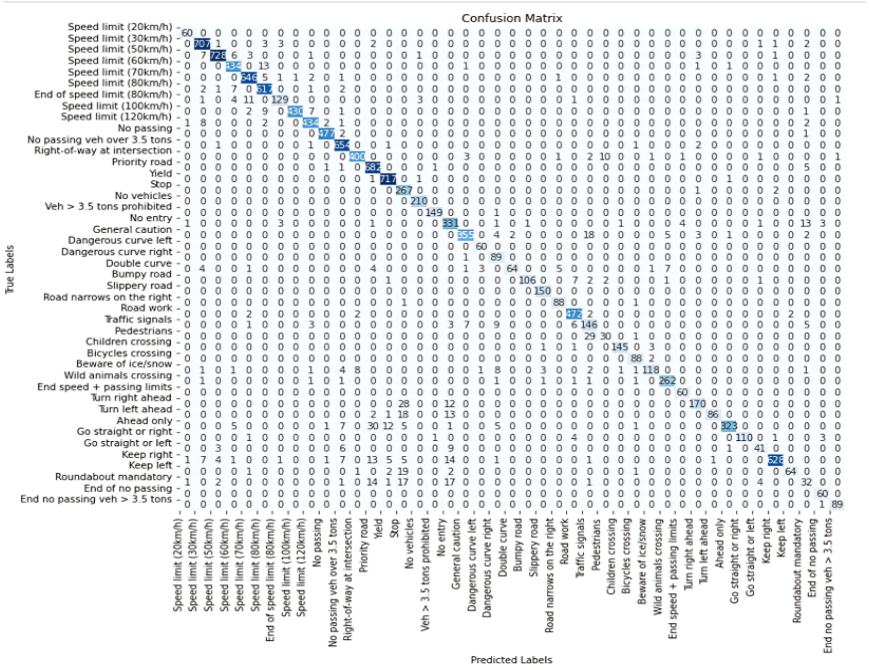

Fig. 11. Confusion matrix of the VGG.

Upon analyzing the confusion matrix, it becomes evident that both diagonal and off-diagonal elements play a crucial role. The diagonal elements indicate correctly predicted labels, showcasing the model's accurate recognition of specific traffic signs. Conversely, the off-diagonal elements represent misclassifications.

In Case 1, the sequential CNN model demonstrated varying performance across different classes. Some classes achieved high accuracy and were correctly recognized, while others experienced misclassifications.

In contrast, Case 2 exhibited a notable improvement in accuracy compared to Case 1. However, similar to Case 1, variations in performance were observed across different classes. Most classes achieved high accuracy, indicating accurate recognition. Although Case 2 showed overall better performance compared to Case 1, there is room for further improvement to enhance the model's accuracy across all classes. Moreover, the VGG19 model showcased superior performance in classifying the 43 classes compared to the sequential CNN model in both cases. The majority of classes were predicted correctly, highlighting the VGG19 model's high accuracy and effectiveness in identifying and classifying various traffic signs.

This can be attributed to the VGG19 model's deeper architecture and utilization of pre-trained weights, which enhance its ability to accurately recognize and classify different types of traffic signs.Comparing the two cases, Case 2 exhibited even higher accuracy compared to Case 1. This indicates that the changes implemented in Case 2, such as adjusting the learning rate, batch size, and epochs, positively influenced the model's learning efficiency and recognition capabilities. The improvements in Case 2 highlight the importance of fine-tuning the model to achieve better overall performance.

4.4.4. Precession, recall and F1-Score

Precision, Recall, and F1-Score for each of the 43 classes were measured for the two models under two different conditions that has been mentioned. Case 1 involved training for 30 epochs, with a batch size of 16 and a learning rate of 0.0001. In Case 2, the models were trained for 30 epochs, with a batch size of 8 and a learning rate of 0.001. These metrics help us understand how well the models perform in terms of accuracy, capturing all relevant information, and overall effectiveness in classification.

Tables 4 and 5 listed Precision, Recall, and F1-Score for each of the 43 classes for the sequential model and VGG 19 Respectively. The sequential CNN model was evaluated in two different cases: Case 1 and Case 2. In

Case 1, the model achieved an overall F1-Score of 0.92, while in Case 2, the overall F1-Score was 0.89. The precision, recall, and F1-Score values varied across the different traffic sign classes.

Table 4. Precession, Recall and F1-Score for the Sequential Model

Case 1				Case 2			
Class	**Precession**	**Recall**	**F1-Score**	**Class**	**Precession**	**Recall**	**F1-Score**
0	0.86	0.85	0.86	**0**	0.94	0.55	0.69
1	0.92	0.95	0.93	**1**	0.90	0.95	0.92
2	0.98	0.98	0.98	**2**	0.84	0.93	0.88
3	0.80	0.97	0.88	**3**	0.98	0.85	0.91
4	0.97	0.97	0.97	**4**	0.93	0.97	0.95
5	0.92	0.98	0.95	**5**	0.80	0.97	0.88
6	0.99	0.88	0.93	**6**	1.00	0.71	0.83
7	0.94	0.96	0.95	**7**	0.94	0.96	0.95
8	0.96	0.94	0.95	**8**	0.90	0.96	0.93
9	0.95	0.95	0.95	**9**	0.73	0.97	0.83
10	0.93	0.99	0.96	**10**	0.96	0.96	0.96
11	0.94	0.98	0.96	**11**	0.98	0.96	0.97
12	0.81	0.60	0.69	**12**	0.97	0.84	0.90
13	0.96	0.99	0.98	**13**	0.98	0.99	0.99
14	0.36	0.64	0.46	**14**	0.54	0.97	0.69
15	0.87	0.79	0.83	**15**	0.99	0.50	0.66
16	1.00	0.99	0.99	**16**	1.00	0.97	0.98
17	0.82	0.81	0.82	**17**	0.92	0.97	0.94
18	0.94	0.89	0.91	**18**	0.80	0.94	0.87
19	0.95	1.00	0.98	**19**	0.92	0.98	0.95
20	0.81	0.96	0.88	**20**	0.50	1.00	0.67
21	0.94	0.88	0.91	**21**	0.78	0.91	0.84
22	0.99	0.97	0.98	**22**	1.00	0.81	0.89
23	0.87	0.95	0.91	**23**	0.82	0.95	0.88
24	0.88	0.92	0.90	**24**	0.89	0.97	0.93
25	0.97	0.96	0.97	**25**	0.93	0.96	0.94
26	0.82	0.77	0.79	**26**	0.75	0.43	0.55
27	0.84	0.62	0.71	**27**	1.00	0.47	0.64
28	0.83	0.99	0.90	**28**	0.98	0.75	0.85
29	1.00	0.93	0.97	**29**	0.73	0.94	0.82
30	0.95	0.80	0.87	**30**	0.93	0.67	0.78
31	0.89	0.98	0.93	**31**	0.91	0.97	0.94
32	0.91	1.00	0.95	**32**	1.00	1.00	1.00
33	0.90	0.54	0.67	**33**	0.84	0.70	0.76
34	0.88	0.38	0.53	**34**	0.05	0.01	0.01
35	0.94	0.32	0.48	**35**	0.98	0.33	0.49
36	0.94	0.91	0.92	**36**	1.00	0.82	0.90
37	1.00	0.45	0.62	**37**	0.00	0.00	0.00
38	0.70	0.68	0.69	**38**	0.93	0.86	0.89
39	0.75	0.10	0.17	**39**	0.35	0.88	0.94
40	0.09	0.41	0.15	**40**	0.02	0.04	0.03
41	0.95	0.95	0.95	**41**	0.96	0.73	0.83
42	0.99	1.00	0.99	**42**	0.58	0.81	0.68

Table 5. Precession, Recall and F1-Score for the VGG 19 model

Case 1				Case 2			
Class	**Precession**	**Recall**	**F1-Score**	**Class**	**Precession**	**Recall**	**F1-Score**
0	0.91	0.97	0.94	0	0.86	0.85	0.86
1	0.89	0.99	0.94	1	0.92	0.95	0.93
2	0.93	0.98	0.95	2	0.98	0.98	0.98
3	0.90	0.97	0.93	3	0.80	0.97	0.88
4	0.98	0.98	0.98	4	0.97	0.97	0.97
5	0.96	0.97	0.97	5	0.92	0.98	0.95
6	1.00	0.87	0.93	6	0.99	0.88	0.93
7	0.95	0.93	0.94	7	0.94	0.96	0.95
8	0.96	0.91	0.94	8	0.96	0.94	0.95
9	0.96	0.96	0.96	9	0.95	0.95	0.95
10	0.92	0.99	0.95	10	0.93	0.99	0.96
11	0.96	0.96	0.96	11	0.94	0.98	0.96
12	0.81	0.47	0.59	12	0.81	0.60	0.69
13	0.81	1.00	0.89	13	0.96	0.99	0.98
14	0.64	0.82	0.72	14	0.36	0.64	0.46
15	0.88	0.99	0.93	15	0.87	0.79	0.83
16	1.00	0.98	0.99	16	1.00	0.99	0.99
17	0.73	0.53	0.61	17	0.82	0.81	0.82
18	0.94	0.87	0.91	18	0.94	0.89	0.91
19	0.94	1.00	0.97	19	0.95	1.00	0.98
20	0.79	0.99	0.88	20	0.81	0.96	0.88
21	0.96	0.77	0.85	21	0.94	0.88	0.91
22	0.95	0.75	0.84	22	0.99	0.97	0.98
23	0.95	0.93	0.94	23	0.87	0.95	0.91
24	0.98	0.98	0.98	24	0.88	0.92	0.90
25	0.86	0.97	0.92	25	0.97	0.96	0.97
26	0.67	0.80	0.73	26	0.82	0.77	0.79
27	0.97	0.57	0.72	27	0.84	0.62	0.71
28	0.97	0.96	0.96	28	0.83	0.99	0.90
29	0.99	0.89	0.94	29	1.00	0.93	0.97
30	0.96	0.86	0.91	30	0.95	0.80	0.87
31	0.88	0.99	0.93	31	0.89	0.98	0.93
32	0.94	1.00	0.97	32	0.91	1.00	0.95
33	0.78	0.45	0.57	33	0.90	0.54	0.67
34	0.90	0.73	0.81	34	0.88	0.38	0.53
35	0.92	0.62	0.74	35	0.94	0.32	0.48
36	0.89	0.78	0.83	36	0.94	0.91	0.92
37	0.69	0.40	0.51	37	1.00	0.45	0.62
38	0.90	0.72	0.80	38	0.70	0.68	0.69
39	0.15	0.14	0.15	39	0.75	0.10	0.17
40	0.01	0.04	0.02	40	0.09	0.41	0.15
41	0.91	1.00	0.95	41	0.95	0.95	0.95
42	0.94	0.93	0.94	42	0.99	1.00	0.99

In Case 1, the model demonstrated varying performance across the classes. Several classes achieved high precision, recall, and F1-Scores, while others exhibited lower scores, indicating challenges in accurately recognizing those signs. The low performance in some classes can be attributed to the lack of data in the dataset, as these signs are not very popular or even used. Similarly, in Case 2, the model showed improvements in performance for some classes, but other classes still needed improvement.

Comparing the two cases, Case 1 demonstrated slightly better overall performance with a higher F1-Score compared to Case 2. The sequential CNN model showed potential in accurately recognizing a wide range of traffic signs, but certain classes presented more challenges than others due to limited data availability.

Further fine-tuning and optimization may be explored to enhance the model's performance, particularly for the classes that exhibited lower scores in both cases. With continued refinement, the sequential CNN model has the potential to be a reliable tool for traffic sign recognition tasks. The VGG19 pretrained model was trained and evaluated using two different cases: Case 1 with a learning rate of 0.001 and batch size of 8, and Case 2 with a learning rate of 0.0001 and batch size of 16. Both cases involved training the model for 30 epochs and testing it on the provided test dataset. The evaluation metrics used were precision, recall, and F1-Score. In Case 1, the VGG19 model achieved an overall F1-Score of 0.94. The precision and recall values varied across different traffic sign classes, ranging from 0.04 to 1.00. Notably, classes 12 and 31 exhibited lower precision and recall values, suggesting potential challenges in accurately identifying those specific signs. This lower performance in some classes can be attributed to the lack of data in the dataset, as these signs are not very popular or even used. Despite this, the model demonstrated satisfactory performance overall with relatively high precision, recall, and F1-Score values.

In Case 2, the VGG19 model achieved an overall F1-Score of 0.95. The precision and recall values ranged from 0.36 to 1.00 across the different traffic sign classes. Similar to Case 1, classes 12 and 31 showed lower precision and recall values, indicating potential difficulties in correctly recognizing those signs due to limited data availability. However, the model exhibited reliable performance in traffic sign recognition with slightly better F1-Scores compared to Case 1. Comparing the two cases, Case 2 with a lower learning rate and larger batch size demonstrated slightly better performance in terms of F1-scores. However, Case 1 with a higher learning rate and smaller batch size still achieved satisfactory results. These findings indicate that different hyperparameter settings can influence the model's performance, and further fine-tuning may be necessary to optimize the trade-off between precision and recall for specific traffic sign classes.

4.4.5 Testing and comparing the models

Two CNN models, namely the sequential model and a pretrained model called VGG19, were selected for the task of Traffic Sign Recognition (TSR). To optimize their performance, these models underwent testing in two distinct cases. In the first case, training occurred over 30 epochs with a batch size of 8 and a learning rate of 0.001, aiming to capture initial learning patterns. The second case involved 30 epochs, a batch size of 16, and an increased learning rate of 0.0001, providing insights into model stability and generalization. The two models were thoroughly tested to ensure they can make accurate predictions on new, unseen data. This verification step validates their learning and readiness for real-world use. Figs. 13 and 14 present the true and predicted labels of random sample images to provide a visual insight into the models' performance on the test dataset. This comprehensive evaluation confirms the models' reliability for practical application. Table 6 presents a thorough comparison of the test accuracy between the two models. Overall, both models demonstrated impressive performance in traffic sign recognition across Case 1 and Case 2. The VGG19 model showcased its superiority in Case 1, achieving a significantly higher test accuracy of 94.214% compared to the sequential model's accuracy of 81.995%. This highlights the effectiveness of the VGG19 model's deeper architecture and pre-trained weights in accurately recognizing traffic signs. Similar to Case 1, Case 2 demonstrated that the VGG 19 model significantly outperformed the sequential model, despite a slight decline from Case 1. The VGG 19 model achieved a test accuracy of 94.658%, whereas the sequential model's accuracy 85.589%. This indicates that both models experienced increased performance due to changes in batch size and learning rate.

Fig. 13. Random samples of predicted image labels of the Sequential model.

Fig. 14. Random samples of predicted image labels of the VGG 19 model.

Table 6. Comparison of the Test Accuracy between the Two Models

Parameter	Case	Sequential	VGG 19
Test accuracy	1	81.995%	94.214%
	2	85.589%	94.658%

5. Conclusion

Traffic sign recognition is a critical area within intelligent transportation systems, focusing on improving road safety and autonomous vehicle navigation. Accurate recognition and classification of traffic signs can have diverse applications, including driver assistance systems, traffic management, and autonomous driving. In recent years, researchers have explored various methodologies for traffic sign recognition, leveraging both custom-designed CNN models and pre-trained networks such as VGG19. These models, trained on extensive datasets like GTSRB, enable the learning of intricate features of traffic signs. The application of these models contributes to improved recognition accuracy, making traffic sign recognition more effective for practical use

in real-world scenarios. This paper successfully achieved its objectives of enhancing the accuracy and reliability of traffic sign classification. The recognition and classification of traffic signs using both a sequential custom CNN model and the pre-trained VGG19 model proved to be effective. The paper involved meticulous customization of the models and training on the GTSRB dataset, which includes 43 classes of traffic signs, evaluated over 30 epochs in two different scenarios.

Throughout the evaluation process, comprehensive metrics such as accuracy, precision, recall, F1-score, and confusion matrices were employed, and validation curves were scrutinized across different scenarios. The findings revealed that the first case, with a batch size of 16 and a learning rate of 0.0001, showed better overall results, particularly in test accuracy. Moreover, the VGG19 model consistently demonstrated superior performance across all aspects, highlighting its robustness in traffic sign recognition. Therefore, within the realm of traffic sign recognition, the VGG19 model emerges as a particularly robust choice, showcasing superior accuracy alongside other critical metrics. These insights were derived from an exhaustive evaluation process that considered multiple parameters, including model architecture, batch size, learning rate, and performance variations across distinct training scenarios.

6. Future Work

The current study has laid a foundation for traffic sign recognition using CNN models. However, there is still a need for improvement. Several avenues of future research in this domain can contribute to refining the accuracy and robustness of traffic sign recognition systems.

1. Real-Time Implementation: Transition from offline recognition to real-time implementation. Investigate the feasibility and performance of deploying the models on real-time video streams, considering the constraints of computational resources and latency requirements.

2. User Interaction and Feedback: Integrate user interaction and feedback mechanisms to further enhance the user experience. This may involve incorporating user-specific adaptations and real-time feedback loops to improve the models' responsiveness to individual driving behaviors and environmental conditions.

3. Dataset Expansion: Add more images to the existing classes, especially those with a small number of images. This augmentation improves model generalization, enhances class discrimination, increases robustness, and facilitates fine-grained feature learning. By including diverse and representative images, the model gains better understanding of class-specific variations, leading to improved accuracy and robustness in traffic sign recognition.

Conflict of Interest

The authors declare no conflict of interest.

Author contributions

Wajeh E. Elside: Original draft preparation, Software; Ahmed J. Abougarair: Manuscript writing and review, Formal analysis, Validation of results. All authors have read and agreed to the published version of the manuscript.

References

[1] N., Bhatt, N., Laldas, P., & Lobo, V. (2022). A real-time traffic sign detection and recognition system on hybrid dataset using CNN, *Proceedings of 7th International Conference on Communication and Electronics Systems.*

[2] Chollet, F. (2018). Deep learning with python. *Manning Publications*.

[3] Burkov. (2019). Machine learning basics. In *The Hundred-Page Machine Learning Book*, 1st ed. Quebec

City, QC, Canada: Andriy Burkov, ch. 2, sec. 1, p. 32.

[4] Sarker, I. (2021). Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions, *SN Computer Science, 2,* 420. doi: 10.1007/s42979-021-00815-1

[5] Guma, W., *et al.* (2022). Design and implementation of a traffic control system based on congestion, *International Journal of Robotics and Automation Technology*, 96–105.

[6] Ma'arif, A., *et al.* (2024). Deep learning-based automated approach for classifying bacterial images, *International Journal of Robotics and Control Systems,4(2).*

[7] Enheba, H., *et al.* (2024). CNNs for automatic skin cancer classification. *Journal of Advances in Artificial Intelligence, 2(2)*.

[8] Hinton, G., *et al.* (2006). A fast-learning algorithm for deep belief nets. *Neural Computation, 18(7)*, pp. 1527–1554. doi: 10.1162/neco.2006.18.7.1527

[9] Dastres R., & Soori, M. (2021). Artificial neural network systems. *Int. J. Imaging Robot., 21(2)*, pp. 13–25. Retrieved from https://hal.science/hal-03349542

[10] Abougarair A., & Aburakhis, M. (2022). Real time traffic sign detection and recognition for autonomous vehicle. *International Robotics & Automation Journal, 8(3)*, 2022.

[11] Nwadiugwu, M. (2021). Neural networks, artificial intelligence and the computational brain. *Proceedings of the IEEE International Conference on Artificial Intelligence (ICAI).*

[12] Heaton, J. (2022). Applications of Deep Neural Networks with Keras, *Heaton Research, Inc*.

[13] Dubey, S., & Chaudhuri, B. (2022). Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, *503*.

[14] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*, The MIT Press.

[15] Alzubaidi, L., *et al.* (2018). Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data, 8,* 2021.

[16] Oun, A., *et al.* (2022). Traffic sign classification Using deep learning based convolutional neural networks. *Azzaytuna University Journal, (42),* 306–326.

[17] Elwefati, S., *et al.* (2023). Identification and control of epidemic disease based neural networks and optimization technique. *International Journal of Robotics and Control Systems, 3(4),* 780–803.

[18] Tilamon, H., *et al.* (2023). Artificial pancreas control using optimized fuzzy logic based genetic algorithm. *International Robotics & Automation, 9(2).*

[19] Sharma, N., *et al.* (2021). Deep learning applications. *A Vision, Global Transitions Proceedings*, *2(1).*, 24–28. https://doi.org/10.1016/j.gltp.2021.01.004.

[20] GTSRB-German Traffic Sign Recognition Benchmark (kaggle.com). Retrieved from https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign

[21] Ke, Z., Cheng, H., & Huang, H. (2021). Analyzing the interplay between random shuffling and storage devices for efficient machine learning. *Proceedings of 2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS),* Stony Brook, NY, USA, (pp. 276–287). doi: 10.1109/ISPASS51385.2021.00050

[22] Abougarair, A. (2020). Neural networks identification and control of mobile robot using adaptive neuro fuzzy inference system. *ICEMIS'20: Proceedings of the 6th International Conference on Engineering & MIS*. https://doi.org/10.1145/3410352.3410734

[23] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* Las Vegas, NV, USA.

[24] Tang, M., Wang, F., & Nguyen, D. (2010). Transfer learning via unsupervised task discovery for visual recognition, *IEEE Transactions on Image Processing, 19(7)*.

[25] Hossin, M., & Sulaiman, M. (2015). A review on evaluation metrics for data classification evaluations.

International Journal of Data Mining & Knowledge Management Process, 5, 01–11.

[26] Attawil, I., & Abougarair, A. (2024). Lateral control of autonomous vehicles through adaptive model predictive control. *Proceedings of 2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA)*, Tripoli, Libya.

[27] Davis J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. *Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh.*

[28] Abosdel, A., *et al.* (2022). Simulation design and hardware implementation of optimization real traffic light control system. *International Science and Technology Journal.*

[29] Zhang, J., *et al.* (2020). Lightweight deep network for traffic sign classification. *Annals of Telecommunications, 75(7)*.

[30] Alshaibi, M., *et al.* (2024). Blood cells cancer detection based on deep learning. *Journal of Advances in Artificial Intelligence, 2*. doi: 10.18178/JAAI.2024.2.1.108-121

[31] Abougarair, A. (2016). Segmentation of road borders based on texture features, *Azzaytuna University Journal.*

[32] Abougarair, A. (2023). Adaptive neural networks based optimal control for stabilizing nonlinear system. *Proceedings of 2023 IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA2023),* Benghazi, Libya.

[33] Ellafi, M., *et al.* (2023). Analysis of mobile accelerometer sensor movement using machine learning algorithms. *Proceedings of 2023 IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA2023),* Benghazi, Libya.

[34] Salih, O., *et al.* (2023). An overview of skin lesion segmentation methods: Techniques, challenges, and future directions. *Proceedings of 2023 IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA2023),* Benghazi, Libya.

[35] Abougarair, A. (2022). Integrated controller design for underactuated nonlinear system. *Proceedings of Second International Conference on Power Control and Computer Technologies (ICPC2T 2022).*

[36] Aburakhis, M., *et al.* (2022). Performance of anti-lock braking systems based on adaptive and intelligent control methodologies. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI), 10.*

[37] Edardar, M. (2021). Tracking control with hysteresis compensation using neural networks. *Proceedings of 2021 IEEE 1st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA2021),* Tripoli, Libya.

[38] Almgallesh, H., *et al.* (2024). Dynamics and optimal control of quadcopter. *Proceedings of 2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA),* Tripoli, Libya.

[39] Ashreadi, M., *et al.* (2024). Model predictive control for optimizes battery charging process. *Proceedings of 2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA),* Tripoli, Libya.

[40] A. Emhemmed, A., *et al.* (2022). Optimization wireless power transfer using simulation relevant parameters. *Proceedings of IEEE 21st International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA),* Sousse, Tunisia.

[41] Salih, O., *et al.* (2022). Simulation analysis and control of wireless power transfer for implantable medical devices. *Proceedings of IEEE 21st International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA),* Sousse, Tunisia.

[42] Abougarair, A. (2022). Position and orientation control of a mobile robot using intelligent algorithms based hybrid control strategies. *Journal of Engineering Research, (34),* 67–86.

[43] Swedan, M., *et al.* (2023). Stabilizing of quadcopter flight model. *Proceedings of 2023 IEEE 3rd*

International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA2023), Benghazi, Libya.

[44] Alkaber, I., *et al.* (2024). Comparative evaluation of PID controller tuning through conventional and genetic algorithm. *Proceedings of 2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA),* Tripoli, Libya.

[45] Abougarair, A. (2018). Intelligent control design for linear model of active suspension system. *Proceedings of 30th International Conference on Microelectronics (IEEE),* Tunisia. doi: 10.1109/ICM.2018.8703995

[46] Saheri, W., *et al.* (2023), Feature extraction for fault diagnosis based on recursive generalized extended least squares algorithm. *Proceedings of 2023 IEEE International Conference on Advanced Systems and Emergent Technologies (IC_ASET'2023),* Hammamet, Tunisia.

[47] Abdulrahman, A., *et al.*, (2024). *Optimizing Cancer Treatment Using Optimal Control Theory, 9(11)*, 31740–31769, 2024. doi: 10.3934/math.20241526

[48] Sawan, S., *et al.*, (2024). Cancer treatment precision strategies through optimal control theory, *Journal of Robotics and Control (JRC), 5(5)*, 1261–1290.

[49] Abougarair, A., *et al.* (2024). Model predictive control for achieving lane. *Proceedings of International Conference on Artificial Intelligence and its Applications in the Age of Digital Transformation*, Springer Conference, Nouakchott, Mauritania.

[50] Aboud, M. (2023). Robust H-infinity controller synthesis approach for uncertainties system. *Proceedings of 11th International Conference on Systems and Control (ICSC'2023)*, December 18–20, Sousse, Tunisia.

[51] Buzkhar, I., *et al.* (2023). Design and implementation of hydric controller for two wheeled robot, *IJEIT on Engineering and Information Technology*, *11(1)*.

[52] Elmulhi, A., *et al.* (2023). Sliding mode control for the satellite with the influence of time delay. *Proceedings of 2023 IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA2023),* Benghazi, Libya.

[53] Buzkhar, I., *et al.* (2023). Modeling and control of a two-wheeled robot machine with a handling mechanism. *Proceedings of 2023 IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA2023*), Benghazi, Libya.

[54] Sadek, M., *et al.* (2022). Sliding mode control design for magnetic levitation system. *Journal of Robotics and Control (JRC)*, *3(6)*.

[55] Arebi, W., *et al.* (2022). Smart glove for sign language translation. *International Robotics & Automation Journal*, *8(3)*.

[56] Abougarair, A. (2022). Optimal control synthesis of epidemic model*, IJEIT International Journal on Engineering and Information Technology, 8(2)*.