information

Article

Sensitivity Analysis of Traffic Sign Recognition to Image Alteration and Training Data Size

Arthur Rubio [1,†], Guillaume Demoor [1,†], Simon Chalmé [1,†], Nicolas Sutton-Charani [2] and Baptiste Magnier [2,3,*]

[1] Department of Computer Science and Artificial Intelligence, IMT Mines Ales, Ales, France; arthurrubio0@gmail.com (A.R.); guillaume.demoor@etu.mines-ales.fr (G.D.); simon.chalme@etu.mines-ales.fr (S.C.)
[2] EuroMov Digital Health in Motion, Univ Montpellier, IMT Mines Ales, Ales, France; nicolas.sutton-charani@mines-ales.fr
[3] Service de Médecine Nucléaire, Centre Hospitalier Universitaire de Nîmes, Université de Montpellier, Nîmes, France
* Correspondence: baptiste.magnier@mines-ales.fr
† These authors contributed equally to this work.

Abstract: Accurately classifying road signs is crucial for autonomous driving due to the high stakes involved in ensuring safety and compliance. As Convolutional Neural Networks (CNNs) have largely replaced traditional Machine Learning models in this domain, the demand for substantial training data has increased. This study aims to compare the performance of classical Machine Learning (ML) models and Deep Learning (DL) models under varying amounts of training data, particularly focusing on altered signs to mimic real-world conditions. We evaluated three classical models: Support Vector Machine (SVM), Random Forest, and Linear Discriminant Analysis (LDA), and one Deep Learning model: Convolutional Neural Network (CNN). Using the German Traffic Sign Recognition Benchmark (GTSRB) dataset, which includes approximately 40,000 German traffic signs, we introduced digital alterations to simulate conditions such as environmental wear or vandalism. Additionally, the Histogram of Oriented Gradients (HOG) descriptor was used to assist classical models. Bayesian optimization and k-fold cross-validation were employed for model fine-tuning and performance assessment. Our findings reveal a threshold in training data beyond which accuracy plateaus. Classical models showed a linear performance decrease under increasing alteration, while CNNs, despite being more robust to alterations, did not significantly outperform classical models in overall accuracy. Ultimately, classical Machine Learning models demonstrated performance comparable to CNNs under certain conditions, suggesting that effective road sign classification can be achieved with less computationally intensive approaches.

Keywords: autonomous driving; traffic sign recognition; machine learning; deep learning; random forest; CNN; SVM; LDA; HOG; Bayesian optimization

check for updates

Citation: Rubio, A.; Demoor, G.; Chalmé, S.; Sutton-Charani, N.; Magnier, B. Sensitivity Analysis of Traffic Sign Recognition to Image Alteration and Training Data Size. *Information* **2024**, *15*, 621. https://doi.org/10.3390/info15100621

Academic Editors: Marco Leo and Heming Jia

Received: 7 August 2024
Revised: 12 September 2024
Accepted: 2 October 2024
Published: 10 October 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In the automotive technology landscape, the transition towards autonomous and semi-autonomous vehicles, driven by pioneering companies like Tesla and other manufacturers, marks a pivotal evolution in how we envision transportation. A key aspect of this transformation is the critical importance of accurate road sign recognition, which serves as a foundational element for ensuring the safety and efficiency of autonomous driving systems. The ability to correctly interpret road signs is essential for the proper functioning of these vehicles, directly impacting their reliability and safety.

Our study conducts a comprehensive analysis using various Machine Learning (ML) models on the German Traffic Sign Recognition Benchmark (GTSRB) dataset [1], focusing specifically on classification tasks. This analysis spans classical Machine Learning techniques, such as Support Vector Machines (SVM) and Random Forests [2], and extends to

more sophisticated Deep Learning methods employing Convolutional Neural Networks (CNNs) [3,4]. Beyond evaluating performance, our research emphasizes a sensitivity analysis aimed at evaluating the models' adaptability to alterations such as vandalism and changing environmental conditions (weather, lighting, etc.), as well as the number of training images per sign. These modifications replicate the environmental wear and deliberate markings, as well as the diverse conditions encountered by self-driving vehicles in real-world scenarios.

Given the increasing reliance on semi-autonomous systems and the historical precedence of accidents attributed to misinterpretation of surroundings by such technologies, our study highlights the urgent need for improving the robustness of Machine Learning models used in traffic sign recognition. Enhancing model reliability under various challenges is crucial for advancing the safety standards of future autonomous vehicular technologies, ultimately moving us closer to a future where autonomous driving is synonymous with unparalleled safety and reliability.

2. Literature Review

Traffic sign recognition presents a real-world multi-label classification challenge characterized by unbalanced class frequencies due to the varying prevalence of different types of traffic signs on roads [5]. This area has been studied for decades and encompasses a diverse range of methods and approaches. The introduction of the German Traffic Sign Recognition Benchmark (GTSRB) in 2011 represented a major milestone in this field, establishing a standardized benchmark that enables method comparison and has stimulated substantial research efforts [6].

Prior to GTSRB, researchers explored various approaches in traffic sign recognition, primarily relying on recognizing color and shape characteristics [7], often incorporating aspects like vehicle motion [8] and early neural networks [9].

The introduction of the GTSRB provided the first standardized way to compare different methods for recognizing traffic signs. Upon its release, a competition using the GTSRB dataset was held to test various Machine Learning techniques, including Convolutional Neural Networks (CNN), Support Vector Machines (SVM), Random Forests, Linear Discriminant Analysis (LDA), ensemble classifiers, slow feature analysis, and k-d trees. The Dalle Molle Institute for Artificial Intelligence Studies (IDSIA) team achieved an impressive accuracy of 98.98% using a group of Convolutional Neural Networks (CNNs), where the model's output was the average of several CNNs. In a later phase of the competition, IDSIA improved their accuracy to 99.15% by combining CNNs with Multi-Layer Perceptrons (MLPs) trained on Haar features and Histogram of Oriented Gradients (HOG) [3], thus winning the final contest. Le Cun and Sermanet's team came in second place in both phases using a multi-scale CNN approach [4], which feeds the output of the last convolution layer and the subsequent max-pooling layer output into the fully connected classifier, achieving an accuracy of 98.31%. Team VISICS secured third place in the first phase with a 97.88% accuracy using a one-vs.-all approach for SVM, combining two types of HOG descriptors. In the second phase, team CAOR (Center for Robotics at Mines Paris) achieved third place using a Random Forest approach with various HOG descriptors to reach an accuracy of 96.14% [2]. Other methods using Linear Discriminant Analysis (LDA) and K-Nearest Neighbors (KNN) also showed promising results. Those results can be seen in Table 1.

Table 1. Results of the GTSRB contest in 2011.

Rank	Team	Method	Correct Recognition Rate
1	IDSIA	Committee of CNNs	99.46%
2	-	Human Performance	98.84%
3	Sermanet	Multi-Scale CNNs	98.31%
4	CAOR	Random Forests	96.14%

Our study builds upon this foundation by exploring how the performance of classical Machine Learning and Deep Learning models evolves with different amounts of training data, especially under altered conditions. This exploration is crucial for developing models that can withstand the challenges posed by real-world driving environments. The principal conclusion drawn from this study is that classical ML models can achieve comparable performance to CNNs under specific conditions, suggesting potential for less computationally intensive solutions in practical applications.

3. Methodology

In this section, we discuss the methods used to prepare the dataset, the models we compared, the data transformations applied, and the evaluation protocol followed. The goal is to understand how different models perform on the task of traffic sign recognition when subjected to various alterations and input conditions.

3.1. Dataset Overview

In this section, we present the dataset used for this study, including its composition and diversity, which are crucial for ensuring robust model training and evaluation.

The German Traffic Sign Recognition Benchmark (GTSRB) [6] is a foundational dataset in the domain of automated driving systems and computer vision. Established to facilitate the development and benchmarking of Machine Learning models capable of recognizing traffic signs, the GTSRB offers a comprehensive collection of traffic sign imagery suited for tasks ranging from simple classification to more complex, real-world scenario simulations.

Comprising around 40,000 images scattered across 43 categories, the dataset includes a diverse array of signs encountered on German roads, varying in shape, size, and condition to mimic real-world variability. Each image is annotated with a label indicating its corresponding traffic sign class, providing a ground truth for training and testing purposes.

The dataset is distinguished by its real-life diversity, including variations in lighting, angles, and partial occlusions, simulating the challenges autonomous driving systems face in accurately identifying traffic signs under different environmental conditions. This complexity makes the GTSRB an invaluable resource for evaluating the robustness and effectiveness of Machine Learning models, particularly in the context of autonomous driving where precision and reliability are paramount. One can observe the various details of the GTSRB dataset in Table 2.

Table 2. German Traffic Sign Recognition Benchmark (GTSRB) dataset parameters [6].

Parameters	Value
Number of images	39,209
Number of classes	43
Number of images per class	211–2251
Mean number of images per class	912
Format	Portable Pixel Map (PPM)
Size	15 × 15–250 × 250
Number of channels	3
Quantification	3 × 8 = 24 bits

Our transformation strategy seeks to augment the inherent diversity of the GTSRB by amplifying existing variations and introducing controlled simulations of weather or vandalism. While the GTSRB already offers a substantial foundation with its natural variability in lighting and viewing angles, our transformations aim to extend these variations further. This approach allows us to not only exaggerate the changes present in the original dataset but also to incorporate the effects of environmental wear and vandalism, thereby enriching our analysis. Crucially, by controlling these transformations, we can precisely measure the impact of each modification on the different models' road sign recognition performance. This methodological control provides a detailed understanding of how each

model responds to intensified or altered conditions, offering valuable insights into necessary improvements for ensuring the reliability of autonomous driving systems in the face of real-world complexities.

In our study, the GTSRB serves as the primary dataset for the training and initial testing of the selected Machine Learning models. It provides a rigorous testing ground to assess the models ability to generalize from the learned representations of traffic signs to accurately classify unseen images, a critical capability for the deployment of autonomous driving technologies. You can see samples from the GTSRB dataset on Figure 1.



Figure 1. Traffic signs samples from the GTSRB dataset.

3.2. Compared Model

This section outlines the models selected for our study, categorizing them into classical Machine Learning models and Deep Learning models to compare their performance on traffic sign recognition tasks.

For our study, we will compare several types of models. These models can be classified into two major categories: classical Machine Learning models and Deep Learning models. In our study, we will examine three classical Machine Learning models and one Deep Learning model. While classical Machine Learning models typically require manual feature extraction and are often simpler, Deep Learning models automatically learn features from the data using multiple layers of neural networks, making them more powerful for complex tasks.

The selection of these models was based on their proven effectiveness in traffic sign classification tasks, as observed in the GTSRB contest. For Deep Learning, we focus on the Convolutional Neural Network (CNN) model, which is predominantly used in the literature. For classical models, we selected the Random Forest, Linear Discriminant Analysis (LDA), and Support Vector Machine (SVM) models, which have achieved top performance in previous studies.

3.2.1. Support Vector Machine

The first model we studied is the Support Vector Machine (SVM) [10]. This model is designed to solve classification problems by relying on concepts of margins. The margin refers to the distance between the boundary and the closest samples, called support vectors. This model is primarily used for its low number of hyperparameters and its good practical performance.

The diagram in Figure 2 depicts the fundamental components of a Support Vector Machine (SVM) classifier. The separating hyperplane (blue line) serves as the decision boundary that divides the two class, represented by red and blue dots. The support vectors (circled lines) are the critical data points that lie closest to the hyperplane and are used to define the margin. The margin (orange line) is the distance between the dashed lines parallel to the hyperplane, which pass through the support vectors. Maximizing this margin enhances the model's ability to generalize to unseen data.

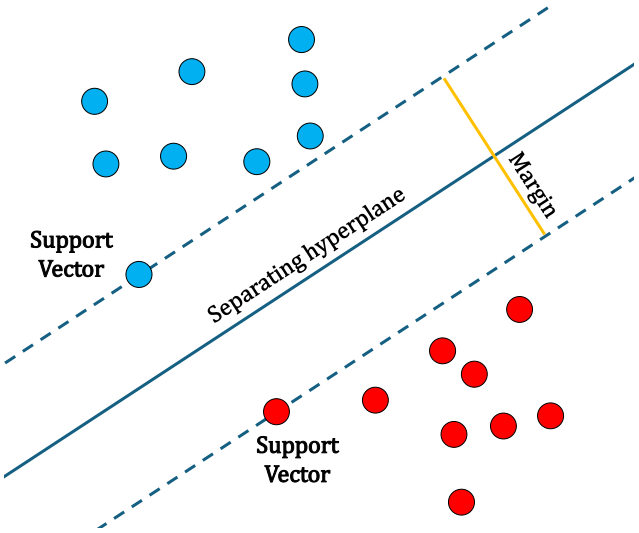


Figure 2. Illustration of a Support Vector Machine (SVM) classifier depicting the separating hyperplane (blue line), support vectors (blue dotted lines), and margin (orange lines). The two classes are represented by red and blue data points.

3.2.2. Random Forest

In our study, we also investigate Random Forest models [11], which are used for both classification and regression problems. The Random Forest as shown in Figure 3 is based on a concept stating that a crowd of people will always achieve better results than a single expert in their field of expertise. The Random Forest aggregates simpler decision trees trained on different datasets, enhancing prediction diversity and accuracy through the principle of bagging.

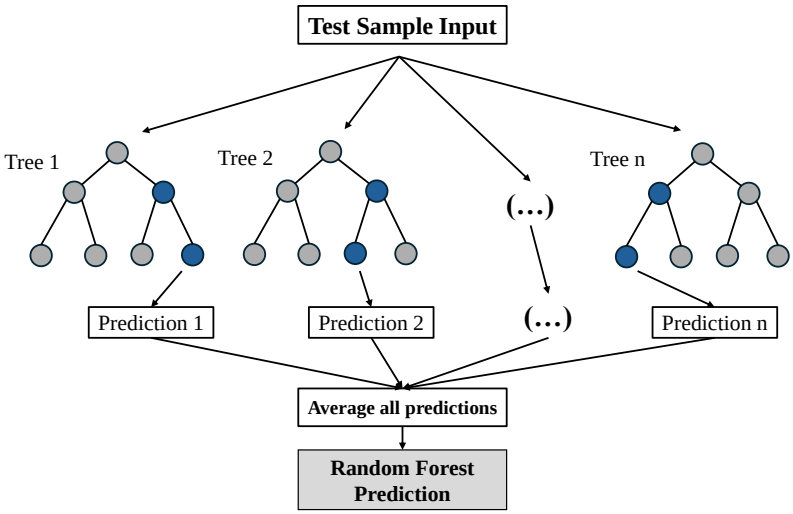


Figure 3. Random Forest classifier. The test sample input is evaluated by multiple decision trees, each providing an individual prediction. These individual predictions are then averaged to produce the final Random Forest prediction.

3.2.3. Linear Discriminant Analysis

The last classical model we are studying is Linear Discriminant Analysis (LDA) [12], which addresses overfitting issues common in models with numerous features by reducing data dimensionality while preserving class variability. LDA achieves this by finding a linear combination of features that maximizes class separability and minimizes within-class variance. This involves two main steps: identifying the linear discriminant function that best separates the classes, and using this function to project new observations onto the separation line. As illustrated on Figure 4, before applying LDA, data points (blue and red dots representing different classes) are not well-separated in their original multidimensional space, leading to potential classification difficulties. After applying LDA, the data points are projected onto a new axis that maximizes class separability, represented by a decision boundary (black line), facilitating more accurate classification.

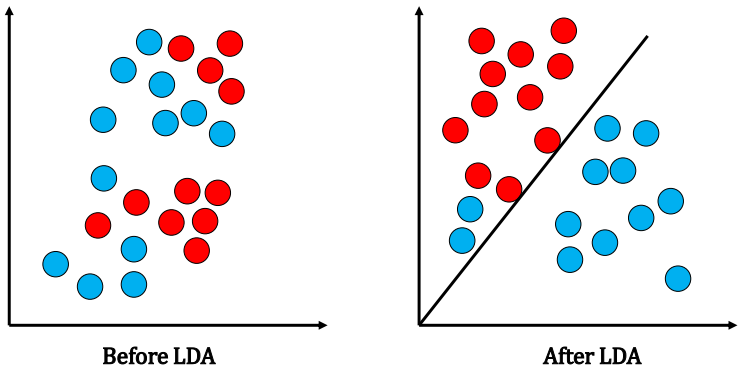


Figure 4. Linear Discriminant Analysis (LDA) process showing data distribution before (**left**) and after (**right**) LDA. Post-LDA, data is projected to maximize class separation, enhancing classification by preserving discriminatory information.

3.2.4. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have revolutionized the field of image classification, setting new benchmarks for accuracy and performance [13]. This transformation was clearly demonstrated in the results of the ImageNet competition showcased on Figure 5, where CNNs consistently outperformed all other methods. CNNs, with their Deep Learning architecture, excel at automatically detecting complex features and patterns in images, which has enabled them to achieve superior results compared to traditional image processing techniques.

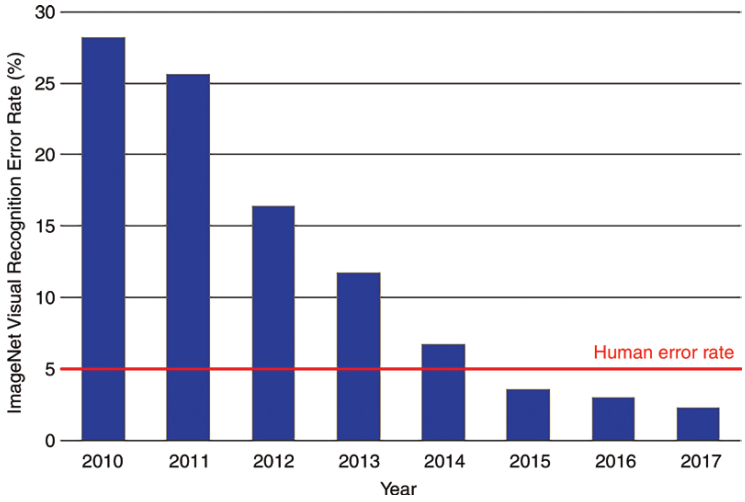


Figure 5. ImageNet error rate trends: Since 2012, CNNs have dominated, reaching superhuman performance in image classification.

Even though CNN techniques have evolved significantly over the years, our study focuses on a basic CNN architecture. While global image classification requires more complex and deeper models, for classification on GTSRB, a basic architecture can achieve almost perfect accuracy. As shown in Figure 6, the Convolutional Neural Network (CNN) used in our study applies convolutional layers to detect patterns and features like edges and textures, with ReLU activation introducing non-linearity. The network includes three convolutional layers followed by max-pooling layers, which reduce the spatial dimensions of the feature maps, retaining essential information and enhancing robustness to spatial variations. The first convolutional layer has 32 filters of size 3×3, while the subsequent two layers have 64 filters each. Input images with three channels (RGB) are processed through these layers, then the feature maps are flattened into a one-dimensional vector, which is input into two fully connected layers with 64 neurons each. These layers perform high-level reasoning and classification using weighted connections. Finally, the output layer, with a SoftMax activation function, produces a probability distribution across the 43 target classes (traffic signs). The CNN model was selected for its simplicity and effectiveness, with hyperparameters such as learning rate, batch size, and number of epochs optimized using a grid search approach.

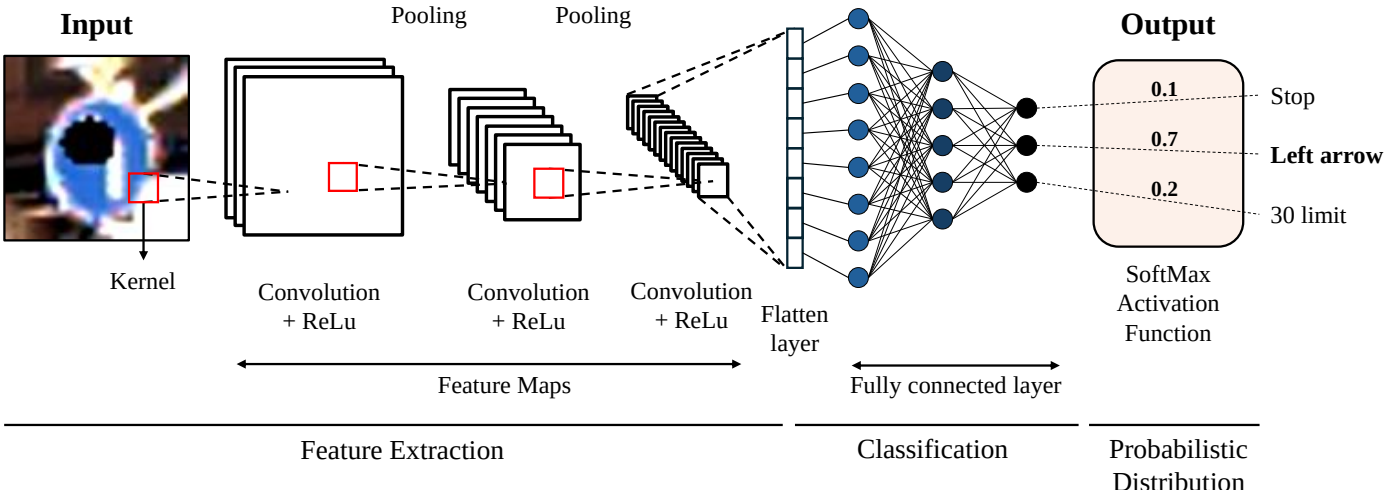


Figure 6. Architecture of a Convolutional Neural Network (CNN) for image classification, illustrating the stages from input images through convolution, pooling, and fully connected layers to the final probabilistic distribution and classification output.

3.3. Transformations

In this section, we describe the data alterations applied to simulate real-world traffic sign conditions, such as environmental wear and vandalism, and their impact on model performance.

3.3.1. Database Alteration

In actual driving scenarios, traffic signs may be subjected to alteration from two primary sources:

1. Application of stickers or paint;
2. Environmental wear, such as fading colors or loss of detail.

To simulate these forms of alteration in our database, we employed a two-step approach: initially, we created a visual dashboard to explore and visually confirm the types of transformations possible. This interactive tool allowed us to manually apply various modifications to traffic sign images and immediately see the outcomes.

Leveraging the insights gained from the dashboard, we automated the process of digital alteration by implementing a specific code on the initial GTSRB database. This automation involved systematically applying the transformations we had explored visually—

such as adjusting shapes, colors, and image properties like brightness, color saturation, and contrast.

To visually showcase the diverse modifications we can apply to our database, we developed an interactive dashboard showcased on Figure 7. This platform allows users to observe and interact in real time with the changes implemented in the database. By visiting the dashboard on this link (https://arthurrubio.shinyapps.io/autonomous-driving/) (accessed: 1 October 2024), users can gain a hands-on understanding of how data alterations can influence outcomes in real-time.

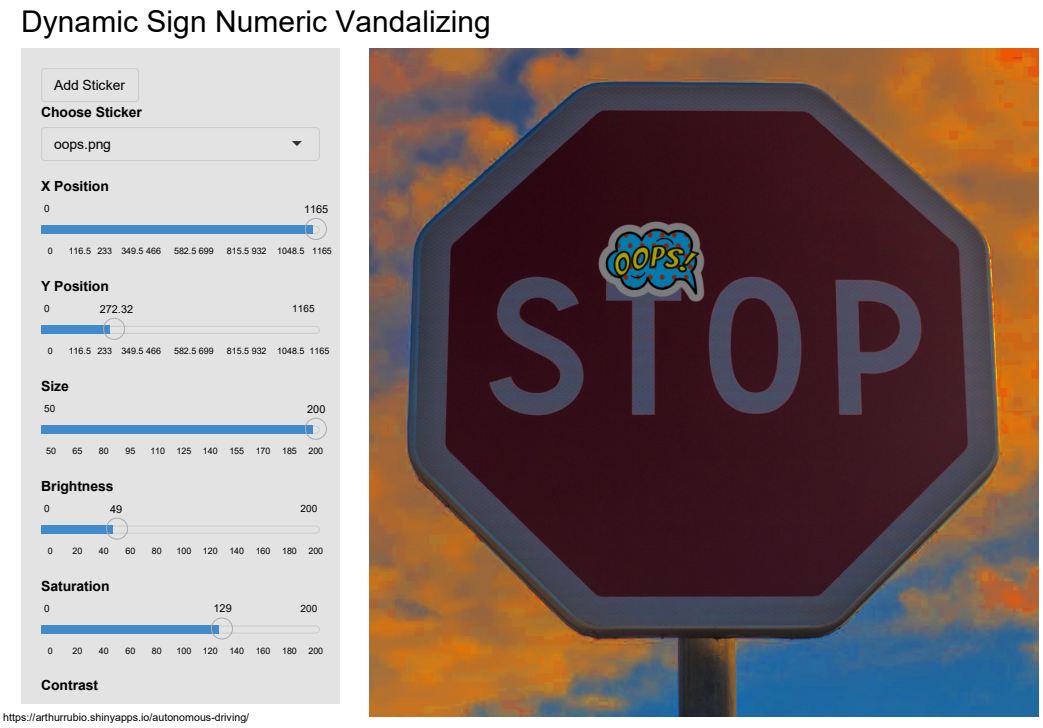


Figure 7. Example of traffic sign alteration using the dashboard.

In the design of this interactive dashboard, we aimed to create a more realistic experience by replacing the uniform obstructions with stickers. This adjustment not only enhances the visual fidelity of the simulation but also closely mimics real-world scenarios where traffic signs may be partially obscured by stickers or other materials. The interactive features of the dashboard allow users to toggle different types of visual modifications, providing a comprehensive view of how each element can affect the visibility and recognition of traffic signs in an autonomous driving context.

In addition to developing the visual dashboard, we implemented a code designed to alter the original GTSRB (German Traffic Sign Recognition Benchmark) database. This code executes a series of transformations detailed in Algorithm 1 and elaborated upon in the subsequent text. The process begins with the introduction of obstructions to the images. These obstructions are randomly selected from a set of shapes: squares, rectangles, triangles, and circles, and are uniformly colored in one of the following colors: red, blue, green, yellow, orange, purple, white, black, cyan, or gray. The obstructions cover between 1% and 10% of the panel. Subsequently, adjustments are made to the brightness, color saturation, and contrast levels of the images. Each of these properties is modified by a random factor ranging from 30% to 170% of its original value, thus ensuring that each image is uniquely altered. Values below 100% reduce brightness, saturation, or contrast, while values above 100% increase them, making the image brighter, more vibrant, or higher in contrast. This dynamic manipulation not only mimics various real-world scenarios of visual impairments but also enhances the robustness of the GTSRB database for more

effective training and evaluation of traffic sign recognition systems. You can see examples of those changes on Figure 8.

Algorithm 1 Traffic Sign Alteration Protocol.

Input:
Traffic sign image *Sign*
Output:
Altered traffic sign image *AltSign*
Load image *Sign*
Randomly select shape from {square, rectangle, triangle, circle}
Randomly select color from {red, blue, green, yellow, orange, purple, white, black, cyan, gray}
Compute random area to cover, ranging between 1% and 10% of the total sign area
Overlay the sign image with randomly selected shape and color
Modify image brightness by applying a random factor between 0.3 and 1.7
Modify color saturation by applying a random factor between 0.3 and 1.7
Modify contrast by applying a random factor between 0.3 and 1.7
return the modified image *AltSign*



Figure 8. Traffic signs samples from the altered GTSRB dataset.

3.3.2. HOG Application

The second processing phase we applied to our dataset is the calculation of the Histogram of Oriented Gradients (HOG) [14]. HOG is used to extract key features from our traffic sign images by analyzing gradient changes, which helps in classification. This process captures essential shape and texture information, making the images more recognizable for Machine Learning algorithms. The Oriented Gradient decomposition can be seen on Figure 9.

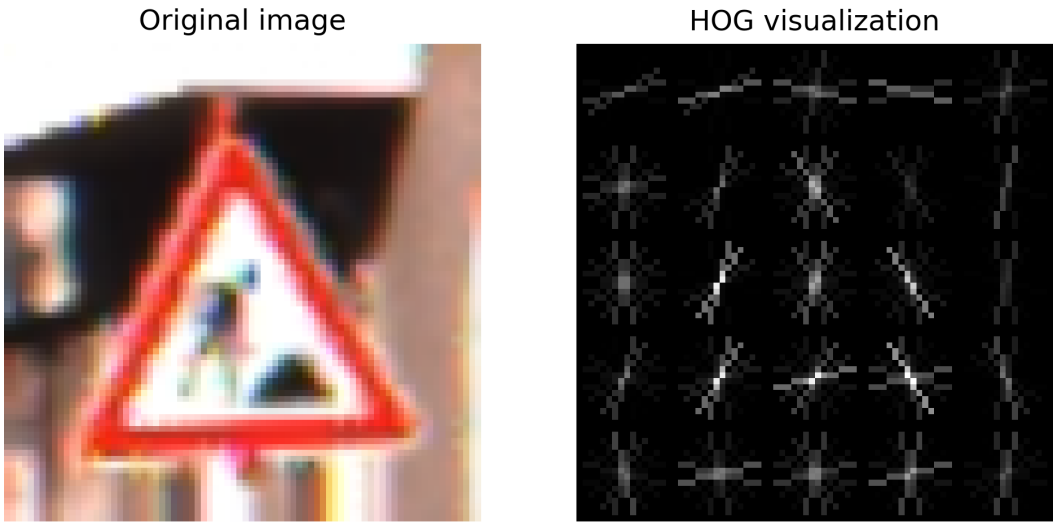


Figure 9. HOG representation (**right**) of an image from the GTSRB database (**left**).

HOG was selected among many available image features due to its effectiveness in capturing shape and texture, which are critical for traffic sign recognition. While color information is important and could enhance the recognition process, HOG's robustness

to variations in illumination and contrast makes it a reliable choice for this application. Additionally, the monochromatic nature of HOG simplifies the feature extraction process, ensuring consistency across different lighting conditions. In our experiments, HOG proved especially useful when training classical models like SVMs and decision trees, as it enhanced feature extraction and led to a significant improvement in classification accuracy, particularly in scenarios with inconsistent lighting and contrast.

The computation of HOG involves calculating the intensity gradients of the image to detect edges, dividing the image into cells, and computing an orientation gradient histogram for each cell. These histograms are then normalized within local blocks to account for variations in illumination and contrast and concatenated to form the complete HOG descriptor. For our study, we set the cell size to 14×14 pixels, block size to 2×2 cells, and the number of orientations to 9. We used the L2-Hys norm, as defined in Equation (1). The unnormalized vector containing all histograms of a single block is denoted by v, its k-norm by $||v||_k$, and ϵ is a small constant:

$$f = \frac{v}{\sqrt{\|v\|_2^2 + \epsilon^2}}. \quad (1)$$

In parallel, we converted our images to gray scale to prepare the image dataset for HOG computation, with gray scale simplifying the feature extraction process. We then computed the HOG descriptors for each image of the GTSRB dataset. The inclusion of HOG enhanced the performance of classical models by providing a more detailed representation of image gradients, improving their ability to classify traffic signs accurately. This impact was especially noticeable in tests where illumination and contrast varied significantly, as HOG's normalization process helped to maintain robustness across different conditions. The extracted HOG features were used for both the training and testing phases of our model evaluation.

3.4. Evaluation

This section explains the evaluation methodology, focusing on model calibration and the experimental protocol to assess performance under different conditions.

3.4.1. Model Calibration

Before evaluating our models on the datasets, we conducted hyperparameter fine-tuning. The extent of this tuning was adapted to the available computational resources. To achieve optimal tuning, we utilized Bayesian hyperparameter optimization via the BayesSearch library. Bayesian optimization presents several advantages over traditional methods like GridSearch [15]. Rather than confining the search to a predefined discrete space, Bayesian optimization explores a predefined continuous space, thereby enhancing efficiency and performance.

Bayesian optimization operates as follows [16]: initially, the model tests hyperparameter values randomly selected from the search space. Based on these initial tests, it employs a Gaussian Process (GP) [17] probabilistic model to estimate performance as a function of the hyperparameters, subsequently selecting the next hyperparameters to test. At each iteration, the model undergoes evaluation using cross-validation, continuously refining the probabilistic model. The optimization process proceeds until either a specified number of iterations (n-iter attribute) is reached or the probabilistic model converges.

This method confers significant advantages, including more efficient search processes, the capability to handle complex and high-dimensional spaces, and improved utilization of computational resources by concentrating on the most promising regions of the hyperparameter space.

For both the SVM and LDA models, Bayesian optimization was performed whenever the training dataset changed, including variations in the proportion of altered data or the number of images per class. Hyperparameters were specifically tailored to each dataset

configuration. This method was feasible due to the minimal impact of hyperparameter tuning on computational time.

Gaussian Process:

A Gaussian Process (GP) is a probabilistic model used for regression tasks, defined by its mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ [17].

The mean function $m(\mathbf{x})$ is given by Equation (2):

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{2}$$

The covariance function $k(\mathbf{x}, \mathbf{x}')$ is defined in Equation (3):

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \tag{3}$$

Assuming the mean function is zero, the GP prior is written as shown in Equation (4):

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}')) \tag{4}$$

Given n observations $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, where $y_i = f(\mathbf{x}_i) + \epsilon_i$ and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, the joint distribution of the observed values $\mathbf{y}$ and the function values at new test points $\mathbf{f}_*$ is described by Equation (5):

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K} + \sigma^2\mathbf{I} & \mathbf{K}_*^\top \\ \mathbf{K}_* & \mathbf{K}_{**} \end{bmatrix}\right) \tag{5}$$

Here, $\mathbf{K}$ is the covariance matrix of the training points, $\mathbf{K}_*$ is the covariance matrix between training points and test points, and $\mathbf{K}_{**}$ is the covariance matrix of the test points.

The GP can predict the function values at new points, $\mathbf{f}_*$, with a Gaussian distribution as shown in Equation (6):

$$\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_* \sim \mathcal{N}(\bar{}_*, \mathbf{\Sigma}_*) \tag{6}$$

The mean $\bar{}_*$ and covariance $\mathbf{\Sigma}_*$ of this distribution are given by Equations (7) and (8), respectively:

$$\bar{}_* = \mathbf{K}_*(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y} \tag{7}$$

$$\mathbf{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_*(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{K}_*^\top \tag{8}$$

These formulas help the GP provide both predictions and the uncertainties associated with them, which are important for selecting the next points to evaluate in Bayesian optimization.

Support Vector Machine:

For the SVM model, we optimized three key hyperparameters [18]: the kernel type, gamma, and the penalty parameter C. Initially, we evaluated which kernel type was most suitable for our problem, testing both the Radial Basis Function (RBF) and linear kernels. Through several iterations, we determined that the RBF kernel produced the best results. With the kernel type established, we then optimized the gamma and C parameters using Bayesian optimization.

For the gamma parameter, which is specific to the RBF kernel, we selected a range of values from 10^{-6} to 10. This parameter controls the width of the RBF kernel, thus influencing its impact on the decision boundary. Concurrently, we optimized the C parameter, which regulates the trade-off between achieving a low error on the training data and minimizing the norm of the weights. For the C parameter, we explored values ranging from 0.1 to 1000.

By systematically optimizing these hyperparameters, we ensured that the SVM model was finely tuned to our specific dataset characteristics, thereby enhancing its classification performance.

Linear Discriminant Analysis:

For the LDA model, there was only one hyperparameter to optimize: the shrinkage. This hyperparameter is useful for enhancing the numerical stability of the model and preventing over fitting. Here, we searched for the optimal value between 0 and 1.

Random Forest:

For the Random Forest model, the number of hyperparameters to test and the computational time did not allow for hyperparameter optimization as performed for the SVM and LDA models. Here, calibration was conducted only once for each change in the test dataset (i.e., for each change in the proportion of the altered database). This calibration was complex as it required finding a balance between computational time and reduction of the randomness of Bayesian search.

For optimization, we focused on three hyperparameters: tree depth, the number of estimators, and the minimum purity decrease. The first two were primarily responsible for the complexity of optimization. To ensure reasonable computational time for Bayesian search, a fixed number of iterations needed to be set. However, setting too few iterations risked poor convergence of the model, not allowing access to the best values for these influential hyperparameters. In our case, we observed variations of up to 20% in accuracy for training the same model with the same amount of data. However, too many iterations would result in excessively long computation times. After several tests on the number of iterations, 20 iterations seemed to be a good compromise in terms of time and reduction of randomness.

Once the number of iterations was fixed, we had to define the different search spaces. For the minimum purity decrease, we searched for values between 0 and 0.2. For tree depth, we searched integers between 1 and 20, and for the number of estimators, we searched between 10 and 10,000 estimators. For all our searches, we found that the optimal number of estimators chosen was the upper bound of the specified space. The same was true for tree depth. Since these two parameters have a significant influence on computation time, we could not explore how far we could increase the upper bound of the search space. Here again, we tried to achieve the best compromise between computation time and model performance. After several attempts, we found that the combination of hyperparameters—21 for maximum depth and 1000 estimators—was an optimal solution for our problem. Therefore, we fixed these two parameters for all our tests. Finally, repeated tests on different datasets allowed us to see that the influence of purity decrease was negligible compared to the other hyperparameters. Therefore, we chose to set it aside. For all our tests, the forest was thus calibrated with the following two hyperparameters: a maximum depth of 21 and 1000 estimators. The summary of the hyperparameter search spaces can be seen on Table 3.

Table 3. Summary table of hyperparameter search spaces.

Model	Hyperparameter	Range of Study
SVM	kernel	Rbf, linear
SVM	gamma	$[10^{-6}, 10]$
SVM	C	$[0.1, 10^{3}]$
Random Forest	max depth	[1, 21]
Random Forest	min impurity decrease	[0, 0.2]
Random Forest	n estimator	$[10, 10^{4}]$
LDA	Shrinkage	[0, 0.9]

3.4.2. Experimental Protocol

To conduct our dual sensitivity study, we first needed to choose an evaluation criterion. Here, we selected accuracy, which is defined in the following equation:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \quad (9)$$

where:

- TP (True Positives) is the number of correctly predicted positive instances.
- TN (True Negatives) is the number of correctly predicted negative instances.
- FP (False Positives) is the number of incorrect positive predictions.
- FN (False Negatives) is the number of incorrect negative predictions.

The choice of this metric was mainly motivated by the context. In the context of autonomous driving, it is crucial to minimize prediction errors as much as possible. Studying accuracy is therefore the best way to mitigate risks effectively. Additionally, this metric is one of the metrics used for model optimization in terms of pure performance, so it is interesting to use it for behavioral studies of the models.

Once the metric of interest was fixed, we established the protocol to observe the behavior of the models concerning the number of input images and alteration levels. First, we needed to find a reproducible way to create a dataset corresponding to the desired level of alteration and the number of images per class.

The first approach was to create a dataset with the correct proportion of altered images and randomly select the appropriate number of images per class from this dataset. However, with this approach, there is no guarantee that the selected sample of images presents the correct proportion of altered images. Therefore, we would need to repeat our measurements numerous times on several samples to approach the exact proportions of altered images in the subset of data. However, limited time and computational power made it impossible to implement this idea.

The second approach, the one we chose, involved creating an entirely altered auxiliary dataset upfront. To create the dataset with the desired features, we relied on a function calculating the number of altered images to be selected from the fully altered database, based on the number of images per class and the desired proportion of altered images. Once the altered images were selected, we complemented them with regular images. Using this method, we constructed a dataset with the exact proportion of degraded images desired, as outlined in Algorithm 2.

Algorithm 2 Dataset Construction with Desired Proportion of Altered Images.

Input:
Original GTSRB dataset $orig_{data}$
Outputs:
Fully altered GTSRB dataset alt_{data}
Dataset with desired proportion of altered images $train_{data}$

1: Load Original GTSRB dataset $orig_{Data}$
2: Completely alter the GTSRB dataset
3: Select number of images to input for each class num_{img}
4: Compute number of altered images for each class: $\text{num}_{\text{alt}} = \text{num}_{\text{img}} \times \text{alt}_{\text{proportion}}$
5: Select altered images from fully altered database
6: Construct dataset with desired proportion of altered images alt_{data}

Once this method for constructing the datasets was established, we could set up the protocol for sensitivity testing. First, we worked on the non-altered dataset with 20 images per class as input. By applying the protocol described earlier, we constructed the dataset for our tests. Once this dataset was created, we applied the HOG descriptors. After all transformations were performed, we split the dataset into training and test subsets, using 80% of the data for training and 20% for testing, and conducted cross-validation. For our study we chose to use ten folds for traditional Machine Learning models and five for Deep Learning models. Once the calculations were conducted, we obtained the average of the k-fold. When varying the number of class samples for accuracy testing, the test subset remained consistent to ensure comparability of results.

4. Results

The Results section presents the outcomes of the experiments conducted, focusing on model performance under varying conditions. Key metrics such as accuracy and sensitivity to alterations in the dataset are examined, providing a comprehensive analysis of how each model performs in traffic sign recognition tasks.

4.1. Impact of Image Quantity on Model Sensitivity

This section explores how varying the number of input images per class affects the sensitivity and performance of the models in traffic sign recognition.

First, we examined the sensitivity of the models to the number of images per class as input for a fixed level of alteration. The primary objective here is to highlight the existence of a threshold beyond which adding new images as input is no longer beneficial to the accuracy of the model.

To study the presence and behavior of this threshold, we start by defining the criterion for its attainment. In our case, we consider the plateau reached if the variations in accuracy due to the addition of new images do not exceed 1% accuracy. This threshold is based on prior studies [19] that consider minor fluctuations in accuracy (typically below 1%) as insignificant, particularly in the context of image classification where the model has reached a level of stable performance.

For this initial plateau study, the alteration of the dataset does not come into play. We choose to fix it at 30% for all our models. To visualize the results, we performed k-fold averaging and took measurements every 20 images from 20 to 140 images, and then at 200 and 300 images per class. We then extracted the values obtained for these levels of the number of images per class as input. These values were plotted to create the accuracy curve as a function of the number of images per class as input.

Firstly, let us examine the performance of the SVM model. As illustrated in Figure 10, we can observe that the best performance is reached for approximately 120 images per class as input. Beyond this point, the model's performance no longer significantly improves. Additionally, we can see that the convergence towards this performance is relatively rapid. For a low number of images per class, an increase of 20 images can lead to an increase of up to 3% in accuracy.

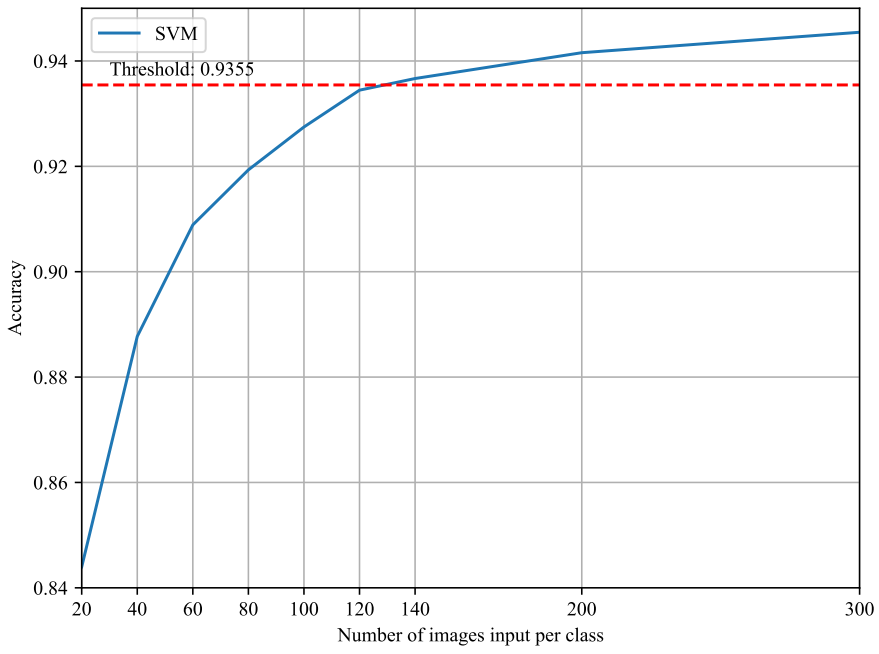


Figure 10. SVM accuracy performance as a function of the number of input images per class (blue). Threshold set at 99% of the final achieved accuracy (red).

Now that we have identified the threshold and made an initial estimation of the number of input images required to reach it, we need to verify if this observation holds true for the other models. To achieve this, we plot the same curve, this time with the four models overlaid on the same graph.

On Figure 11 we can observe that, at equal levels of alteration, all four models reach their threshold with a number of images between 100 and 120. Notably, there is no significant difference in performance between the Deep Learning models.

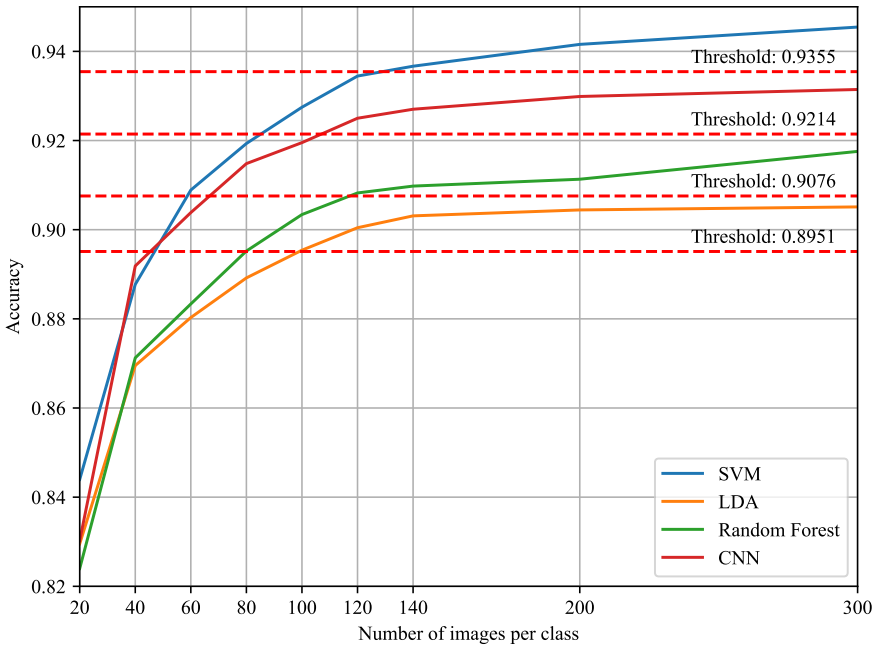


Figure 11. Evolution of model accuracy based on the number of input images for 30% alteration rate: SVM (blue), LDA (yellow), Random Forest (green), CNN (red).

4.2. Performance Sensitivity to Dataset Alteration

In this section, we investigate how different levels of dataset alteration impact the accuracy and robustness of the models under study.

Now that we have clarified the behavior of the plateau with fixed alteration, we will explore the dataset's sensitivity to alteration.

Currently, we lack information on how dataset alteration influences the number of input images required to reach the accuracy threshold. Additionally, we do not know if dataset alteration impacts the value of the accuracy threshold itself. To investigate the models' behavior in response to varying levels of dataset alteration, we fix the number of images per class at 20 to ensure that this quantity does not interfere with our observations.

To visualize the evolution of the values, we plot the different accuracy levels for various degrees of dataset alteration, keeping the number of input images constant.

In Figure 12, we observe that the accuracy decreases as the proportion of altered images in our dataset increases, regardless of the model. This result is logical because the introduction of altered images adds heterogeneity to the dataset. The inclusion of singularities, particularly through color overlay alterations, reduces certainty about the features that define a particular category.

Additionally, we see a notable difference between Deep Learning models and traditional Machine Learning models. While traditional models exhibit an almost linear decrease in performance, Deep Learning models experience a more gradual decline as alteration increases. However, the initial performance loss is more significant for Deep Learning models than for traditional models. In the context of autonomous driving, this characteristic can be somewhat disadvantageous for Deep Learning. Given that the true

proportion of altered images in real-world scenarios is likely closer to around 10%, these factors suggest that Deep Learning models, although seemingly advantageous, may not be the most optimized solution for this particular issue.

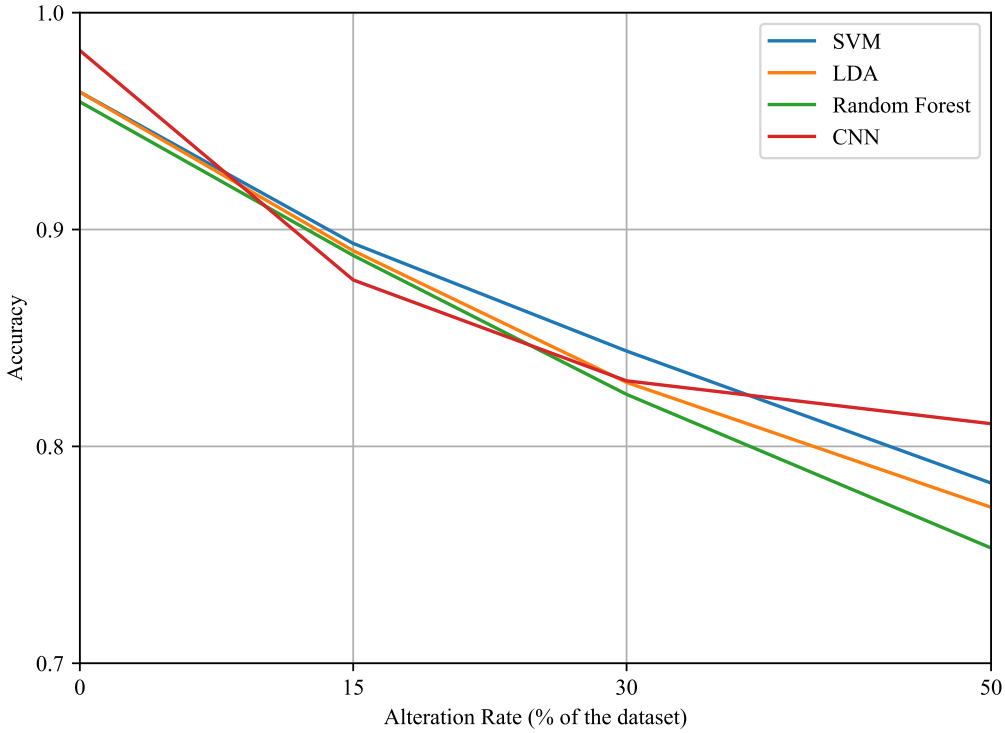


Figure 12. Accuracy variation with the percentage of dataset alteration for 20 images per class: SVM (blue), LDA (yellow), Random Forest (green), CNN (red).

4.3. Impact of Alteration on Sensitivity to Image Quantity

This section combines the effects of image alteration and input quantity variation, analyzing their joint influence on model performance.

Now that we have separately examined the concepts of plateaus and performance alteration, we will focus on the combined effects of image alteration and the variation in the number of images per class as input.

Using the values obtained during the tests, we will plot heatmaps for each model studied. Through this representation, we will be able to identify two main elements:

- The behavior of the plateau and the evolution of its positioning depending on the proportion of altered images.
- The number of images required by the model to achieve performance identical to that obtained with a lower percentage of alteration.

We will present the different heatmaps for each of the models.

In the SVM model heatmap shown in Figure 13, several observations can be made. Firstly, for an unaltered dataset, the accuracy threshold is reached with 40 input images. For other levels of alteration, the thresholds are at about 80, 100, and 120 images per class. Therefore, for SVM models, the introduction of alteration significantly increases the number of images required to reach the accuracy threshold. However, this observation holds true primarily for the transition from a clean dataset to an altered one. When transitioning from an altered dataset to one with even more alterations, the additional number of images required to reach the threshold decreases sharply, from 20 images to 60 images.

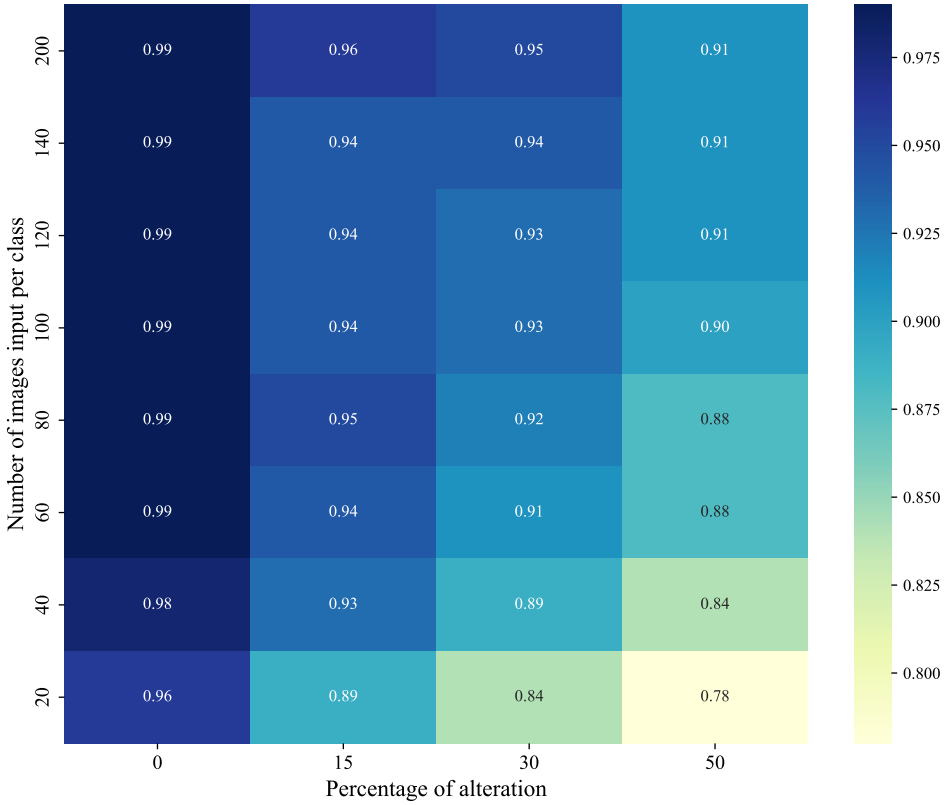


Figure 13. SVM accuracy heatmap by dataset alteration percentage and number of input images per class.

Regarding the model's ability to regain performance with lesser alteration, it is observed that for low levels of alteration (e.g., 15%), it is necessary to provide ten times more images for training. Furthermore, for a training dataset with a high level of alteration, it is impossible to reach the initial performance level. Increasing the proportion of altered signs results in a lower accuracy threshold. Excessive alteration thus appears to limit the model's performance, regardless of the number of images per class provided for training.

For the LDA model, as shown in Figure 14, we observe that the accuracy threshold behavior is notably similar to that of the SVM model. However, two key differences are evident. Firstly, the accuracy threshold for altered datasets requires fewer images to be reached compared to the SVM model. Secondly, while the introduction of alteration leads to a performance drop similar to that seen with the SVM model, the decline in performance for the LDA model is so significant that it becomes impossible to achieve performance levels comparable to those obtained with an unaltered dataset.

The observations made for the LDA model also apply to the Random Forest model, as shown in Figure 15. The plateau behavior is similar, and it remains impossible to regain the performance achieved with non-altered datasets for both training and testing. However, the Random Forest model requires slightly fewer images to reach the accuracy threshold. This reduces the additional number of images needed when transitioning from a clean dataset to an altered one. Consequently, the difference in performance evolution between clean/altered and altered/altered transitions is less pronounced compared to the other models.

For the Deep Learning model, as illustrated in Figure 16, the threshold behavior closely mirrors that of classical Machine Learning models. The number of images required to reach the accuracy threshold is comparable to those of the Random Forest model. However, a notable peculiarity is observed: at a higher level of alteration, specifically 50%, the number of images needed to regain satisfactory results appears to be lower than for less altered datasets.

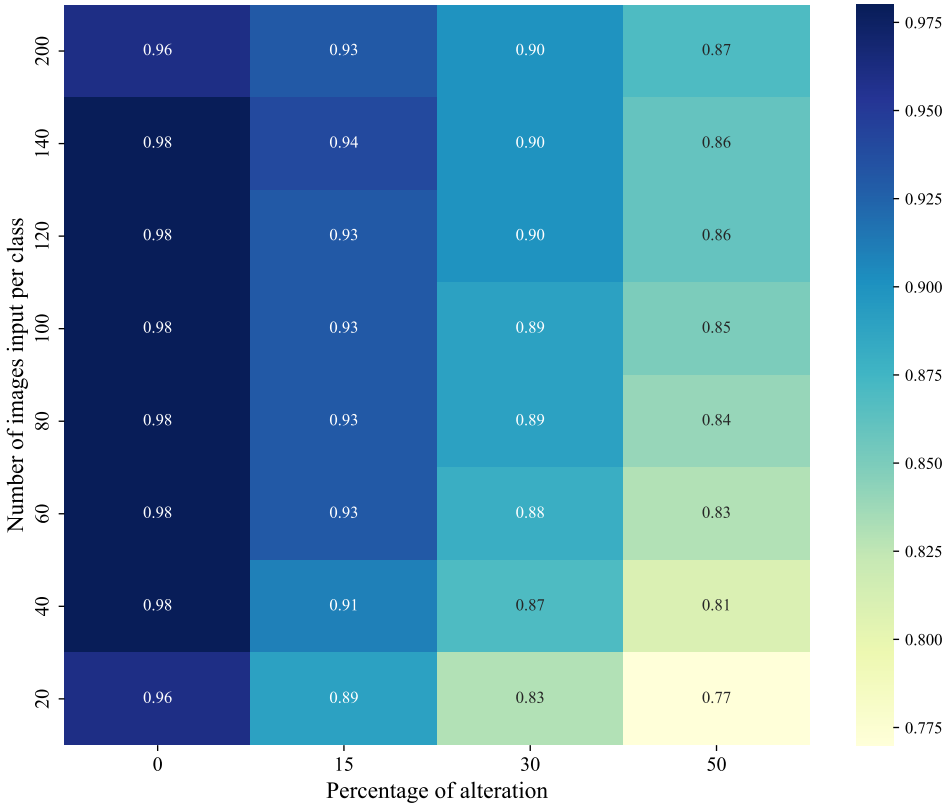


Figure 14. LDA accuracy heatmap by dataset alteration percentage and number of input images per class.

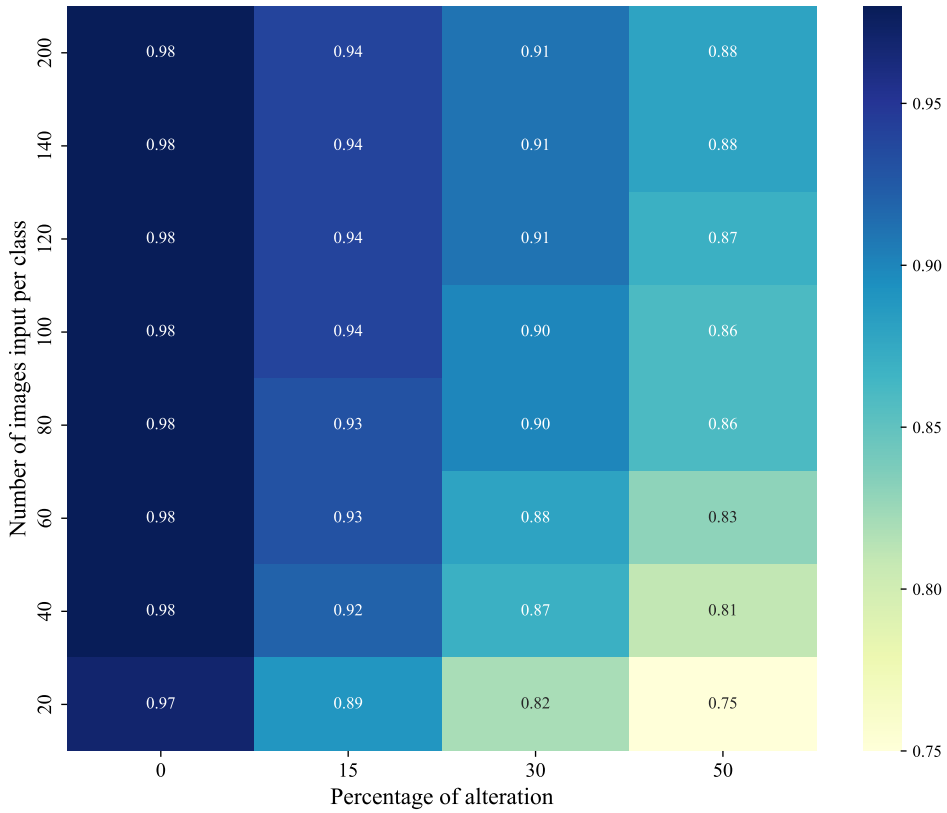


Figure 15. Random Forest accuracy heatmap by dataset alteration percentage and number of input images per class.

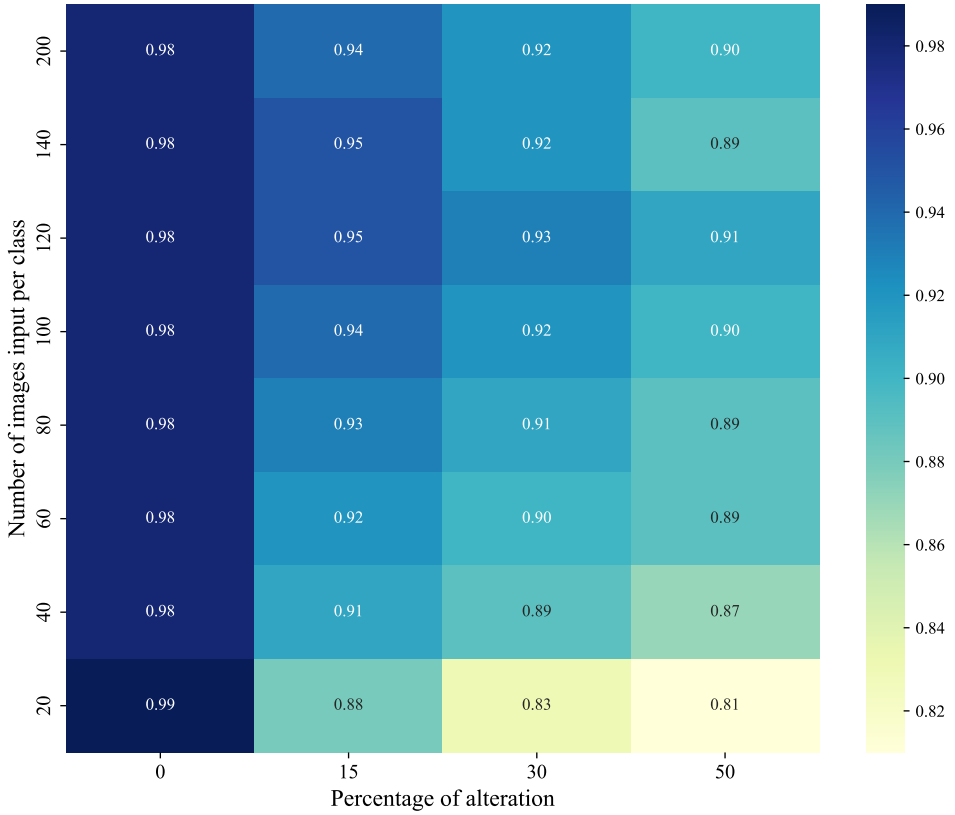


Figure 16. CNN accuracy heatmap by dataset alteration percentage and number of input images per class.

5. Discussion

In our study, we have elucidated several aspects regarding the sensitivity to the number of images in the dataset and the sensitivity to dataset alterations.

Firstly, we identified a plateau where adding new images to the model training data no longer improves performance. For most of the models studied, this plateau is reached between 20 and 140 images per class. The position of this plateau is heavily dependent on the level of dataset alteration. A more altered dataset requires a higher number of images per class to reach the plateau. Additionally, we observed that this maximum value deteriorates as the dataset becomes more altered.

Regarding sensitivity to dataset alteration, we noted that for most models, even a slight alteration of the dataset leads to a performance gap that cannot be bridged even by adding a large number of images for training. This inability can be explained by the impact of alteration on the value of the accuracy plateau. If alteration changes the number of images required to reach the plateau, it also affects its value, sometimes significantly, making it impossible to return to the accuracy values obtained for a less altered dataset. Furthermore, for most classical Machine Learning models, the performance decrease appears to be linear, which is not the case for Deep Learning models.

When comparing classical Machine Learning models with Deep Learning models, we consistently observed similar behaviors. Both model types struggle to regain performance after dataset alteration, but the way accuracy declines with increasing alteration differs. Deep Learning models, despite their advanced architecture, exhibit a faster drop in performance at lower alteration levels, which might make them less suited for real-world scenarios where such alterations are common. As such, Deep Learning models may not always be the optimal choice depending on the problem's context. From a resource-saving perspective, traditional models can provide acceptable performance while being less computationally demanding, making them valuable in scenarios with hardware constraints.

Our study concludes that even with a significant increase in the number of images, the original performance levels cannot be restored once the dataset is altered. Additionally, the presence of a plateau at 140 images per class, regardless of the level of dataset degradation, is a significant finding. The difference in behavior between CNNs and classical models, with CNNs showing less robustness at lower alteration levels but better performance at higher alteration percentages, is also noteworthy. Finally, the similar performance of different models, despite some being more resource-intensive, highlights the potential for optimizing computational resources while maintaining effective results.

6. Conclusions

There are several avenues for further exploration and refinement. Firstly, our study was conducted with limited resources, which may have led to some inaccuracies. A more rigorous hyperparameter search for all models, particularly Random Forests, would be beneficial with more computational power. Exploring the asymptotic behavior of the models using the entire dataset would provide deeper insights into real-world performance and the impact of class imbalance. Furthermore, a more in-depth study of the nature of errors, especially in panel detection, could yield valuable findings since not all errors are equally critical. A detailed classifier analysis could provide a broader view of how alterations and data volume affect performance.

Additionally, it would be valuable to test other models to see if the conclusions of this study are generalizable across different architectures. This could also include experimentation with different image descriptors, offering new perspectives. Overall, there are numerous opportunities for further research in this area.

The key takeaway from this study is that defining the problem thoroughly is crucial for selecting the most appropriate method. While Deep Learning is often at the forefront of image recognition, simpler models can offer significant advantages under specific conditions. Traditional Machine Learning models, in particular, can be highly efficient in resource-constrained environments, making them ideal for applications such as embedded systems or real-time traffic sign recognition. Furthermore, combining classical and Deep Learning models in a hybrid approach could optimize both performance and resource use, with simpler models handling less complex tasks and CNNs reserved for more demanding scenarios.

Future studies could also explore simulating factors such as degradation over time and extreme viewing angles, which are common in real-world traffic scenarios. This would further test model robustness under more challenging conditions. Additionally, improving preprocessing techniques to mitigate the impact of these alterations would make traditional models more adaptable to real-world applications. By expanding the range of scenarios and refining preprocessing, future traffic sign recognition systems could better balance efficiency and accuracy, providing scalable solutions that adapt to increasing complexity.

Author Contributions: Conceptualization, A.R.; methodology, A.R., G.D. and S.C.; software, A.R., G.D. and S.C.; validation, B.M. and N.S.-C.; formal analysis, B.M. and N.S.-C.; investigation, A.R., G.D. and S.C.; resources, A.R., G.D. and S.C.; data curation, A.R.; writing—original draft preparation, A.R.; writing—review and editing, A.R. and B.M.; visualization, A.R., G.D. and S.C.; supervision, B.M. and N.S.-C.; project administration, B.M. and N.S.-C.; funding acquisition, B.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The majority of the data presented in this research article is available on the https://benchmark.ini.rub.de/gtsrb_news.html (GTSRB Benchmark website, accessed: 1 October 2024). Any additional data, if required, can be obtained by contacting the corresponding author upon request.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GTSRB	German Traffic Sign Recognition Benchmark
ML	Machine Learning
DL	Deep Learning
SVM	Support Vector Machine
LDA	Linear Discriminant Analysis
CNN	Convolutional Neural Network
HOG	Histogram of Oriented Gradients
IDSIA	Dalle Molle Institute for Artificial Intelligence Studies
RBF	Radial Basis Function
GP	Gaussian Process
EI	Expected Improvement

References

1. Stallkamp, J.; Schlipsing, M.; Salmen, J.; Igel, C. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Netw.* **2012**, *32*, 323–332. [CrossRef] [PubMed]
2. Zaklouta, F.; Stanciulescu, B.; Hamdoun, O. Traffic sign classification using K-d trees and Random Forests. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), San Jose, CA, USA, 31 July–5 August 2011; pp. 2151–2155.
3. Cireşan, D.; Meier, U.; Masci, J.; Schmidhuber, J. Multi-Column Deep Neural Network for Traffic Sign Classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, 16–21 June 2012; pp. 3642–3649.
4. Sermanet, P.; LeCun, Y. Traffic sign recognition with multi-scale Convolutional Networks. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), San Jose, CA, USA, 31 July–5 August 2011; pp. 2809–2813.
5. Fu, M.-Y.; Huang, Y.-S. A survey of traffic sign recognition. In Proceedings of the IEEE International Conference on Wavelet Analysis and Pattern Recognition, Qingdao, China, 11–14 July 2010; pp. 119–124.
6. Stallkamp, J.; Schlipsing, M.; Salmen, J.; Igel, C. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), San Jose, CA, USA, 31 July–5 August 2011; pp. 1453–1460.
7. Gao, X.W.; Podladchikova, L.; Shaposhnikov, D.; Hong, K.; Shevtsova, N. Recognition of traffic signs based on their colour and shape features extracted using human vision models. *J. Vis. Commun. Image Represent.* **2006**, *17*, 675–685. [CrossRef]
8. Bahlmann, C.; Zhu, Y.; Ramesh, V.; Pellkofer, M.; Koehler, T. A system for traffic sign detection, tracking, and recognition using color, shape, and motion information. In Proceedings of the IEEE Proceedings. Intelligent Vehicles Symposium, Las Vegas, NV, USA, 6–8 June 2005; pp. 255–260.
9. Broggi, A.; Cerri, P.; Medici, P.; Porta, P.P.; Ghisio, G. Real Time Road Signs Recognition. In Proceedings of the IEEE Proceedings. Intelligent Vehicles Symposium, Istanbul, Turkey, 13–15 June 2007; pp. 981–986.
10. Noble, W. What is a support vector machine? *Nat. Biotechnol.* **2006**, *24*, 1565–1567. [CrossRef] [PubMed]
11. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
12. Izenman, A.J. Linear Discriminant Analysis. In *Modern Multivariate Statistical Techniques*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 237–280.
13. Alzubaidi, L.; Zhang, J.; Humaidi, A.J.; Al-Dujaili, A.; Duan, Y.; Al-Shamma, O.; Santamaría, J.; Fadhel, M.A.; Al-Amidie, M.; Farhan, L. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *J. Big Data* **2021**, *8*, 53. [CrossRef] [PubMed]
14. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005; pp. 886–893.
15. Alibrahim, H.; Ludwig, S. A. Hyperparameter Optimization: Comparing Genetic Algorithm against Grid Search and Bayesian Optimization. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC), Kraków, Poland, 28 June–1 July 2021; pp. 1551–1559.
16. Shahriari, B.; Swersky, K.; Wang, Z.; Adams, R.P.; de Freitas, N. Taking the Human out of the Loop: A Review of Bayesian Optimization. *Proc. IEEE* **2016**, *8*, 148–175. [CrossRef]
17. Rasmussen, C.E. Gaussian Processes in Machine Learning. In *Advanced Lectures on Machine Learning*; Bousquet, O., von Luxburg, U., Rätsch, G., Eds.; Springer: Berlin, Germany, 2003.

18. Wainer, J.; Fonseca, P. How to tune the RBF SVM hyperparameters? An empirical evaluation of 18 search algorithms. *Artif. Intell. Rev.* **2021**, *54*, 4771–4797. [CrossRef]
19. Linjordet, T.; Balog, K. Impact of Training Dataset Size on Neural Answer Selection Models. In *Advances in Information Retrieval*; Azzopardi, L., Stein, B., Fuhr, N., Mayr, P., Hauff, C., Hiemstra, D., Eds.; Springer: Cologne, Germany, 2019.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.