# The German Traffic Sign Benchmark (GTSB) Traffic Sign Classification

Daniel Ferreira Mec. Number: 102442
*DETI*
*Universidade de Aveiro*
danielmartinsferreira@ua.pt

Guilherme Antunes Mec. Number: 103600
*DETI*
*Universidade de Aveiro*
guilherme.antunes@ua.pt

***Abstract*—Traffic sign recognition is essential for advanced driver assistance systems and autonomous vehicles. This paper investigates the German Traffic Sign Benchmark (GTSB) dataset and proposes a modified LeNet-5 convolutional neural network for traffic sign classification. We address dataset challenges, such as class imbalance, through data augmentation techniques and introduce modifications to the LeNet-5 architecture to enhance generalization and prevent overfitting. Our model demonstrates competitive performance in terms of accuracy and computational efficiency when compared to state-of-the-art approaches, offering valuable insights for the development of improved traffic sign recognition systems and potential applications in other image classification tasks.**

***Index Terms*—Traffic sign recognition German Traffic Sign Benchmark (GTSB) Convolutional neural networks (CNN) LeNet-5 architecture Data augmentation Model generalization Overfitting prevention Autonomous vehicles Advanced driver assistance systems (ADAS) Image classification**

## I. Introduction

This paper provides an in-depth exploration of the methods, approaches, and algorithms employed in the inaugural TAA project at Universidade de Aveiro, centered around Topics in Automated Learning. The project involved implementing machine learning techniques, acquired during the course or through self-learning, to address one of several problems suggested by the course instructor, Pétia Georgieva.

Our group opted to address the German Traffic Sign Benchmark (GTSB) challenge, which required developing an algorithm capable of accurately recognizing traffic signs. This issue captured our attention due to its widespread relevance and importance in one of the most prominent applications of machine learning today - the development of self-driving, AI-enabled, autonomous vehicles. The current excitement around this subject acted as both inspiration and motivation, pushing us to dedicate our best efforts to find an optimal solution.

Moreover, this project aligns with prior research indicating that autonomous vehicles could potentially prevent up to 90% of avoidable accidents [1]. This serves as a testament to the importance of research in the autonomous driving domain and the potential influence it may have on society.

## II. Data-set

### A. Data-set Specifications

The German Traffic Sign Benchmark (GTSB) dataset, accessible via this link, is a collection of 51,883 images of various dimensions, categorized into three distinct sets: Meta, Test, and Train. These images represent 43 unique traffic signs, serving as classes for image classification.


Fig. 1. All available classes represented in the Meta Set.

**Meta Set**: This set serves as a reference for the appearance of traffic signs in other sets. It contains one example of each traffic sign, with images named numerically from 0 to 42 based on their class.

**Test Set**: Comprising 12,631 mixed images of different signs, this set is utilized to evaluate the model's accuracy. It encompasses a wide range of image quality, such as different lighting conditions, blurriness, rotation, and focus/scale, ensuring a diverse and realistic dataset that reflects the dynamic real-life visibility conditions.

**Train Set**: This set consists of 39,209 images, organized into 43 folders according to their class. Like the Test set, the Train set exhibits significant variations in image quality, providing an extensive range of examples for training the model across varying difficulty levels. Supplementary Data: Alongside the three primary sets, the dataset includes three CSV files containing information about each image, such as width, height, relative path, and class. These files provide additional context for the images in the Meta, Test, and Train sets

Fig. 2. Example of an entry in the Test and Train Sets.

### B. Addressing class imbalance in the GTSB dataset

A significant challenge in the German Traffic Sign Benchmark (GTSB) dataset is the class imbalance. The distribution of the 43 traffic sign classes in the dataset is not uniform: some classes contain as many as almost 10 times the number of samples as others in the Train set. This imbalance in distribution should be addressed to prevent any bias arising in training the model on this dataset.

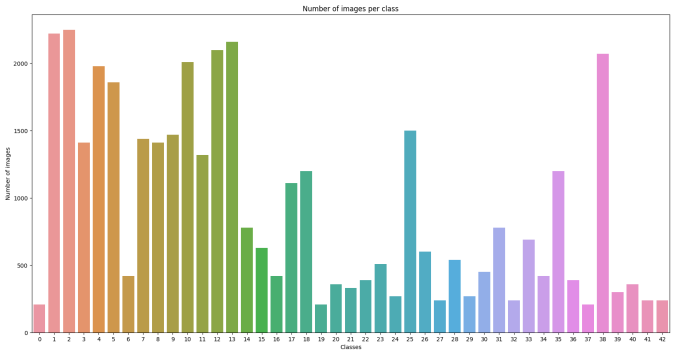


Fig. 3. Distribution of images per class in Train set.

Class imbalance can adversely affect the model's generalization capabilities, as it is less likely to perform well on underrepresented classes. Consequently, it is crucial to address this issue while training the model to ensure a more balanced performance across all classes.

There are several techniques to mitigate the class imbalance problem:

**Data Augmentation**: Generating new samples by applying various transformations to the original images, such as rotation, scaling, flipping, or changing brightness and contrast, can increase the number of samples for underrepresented classes. This approach helps balance the dataset while providing the model with more diverse training examples. [2]

**Resampling**: This technique involves either oversampling the minority classes, undersampling the majority classes, or a combination of both. By balancing the number of samples per class, the model can learn to better recognize underrepresented classes.

**Cost-sensitive Learning**: Assigning different weights or costs to the classes during model training can help address class imbalance. In this approach, higher weights or costs are assigned to the minority classes, effectively penalizing the model more for misclassifying them. This encourages the model to pay more attention to underrepresented classes.

**Transfer Learning**: Utilizing pre-trained models on similar tasks or datasets can help improve the model's performance on minority classes. The pre-trained model can serve as a starting point, with further fine-tuning performed using the GTSB dataset.

By incorporating these strategies in the model training process, it is possible to address class imbalance in the GTSB dataset and achieve a more balanced performance across all traffic sign classes. By changing the images to a single channel colour gradient, we hoped to address this issue.

## III. Convolutional Neural Networks

### A. Convolutional Neural Networks and complex problems

Convolutional Neural Networks (CNNs) have become essential tools for addressing complex problems, particularly in the domain of image classification and computer vision. In contrast to traditional feedforward neural networks, CNNs are specifically designed to handle the unique challenges posed by high-dimensional data, such as images, by exploiting the spatial structure and local patterns present in the data.

The necessity of using CNNs for complex problems, such as traffic sign recognition in the GTSB dataset, arises from their inherent advantages over traditional neural networks [3]:

**Local Connectivity**: In a CNN, each neuron in a convolutional layer is only connected to a small local region of neurons in the previous layer, effectively capturing local features and patterns in the input image. This local connectivity enables the model to learn spatial hierarchies, allowing it to identify more complex features as the network goes deeper.

**Parameter Sharing**: CNNs employ shared weights and biases across the entire convolutional layer, enabling the detection of the same feature or pattern at different locations in the input image. This weight sharing results in a significant reduction in the number of parameters, making CNNs more computationally efficient and less prone to overfitting compared to traditional neural networks.

**Translation Invariance**: Due to the shared weights and local connectivity, CNNs are inherently robust to small translations and distortions in the input images. This makes them particularly suitable for complex problems where objects of interest, such as traffic signs, may appear in various positions, scales, and orientations.

**Pooling Layers**: CNNs often include pooling layers that perform downsampling operations, reducing the spatial dimensions of the feature maps and aggregating local information. This results in a more compact representation, reducing computational complexity and improving the model's ability to capture higher-level abstract features.

Given the high-dimensional nature of the GTSB dataset and the varying appearance of traffic signs, employing a CNN is crucial to effectively learn the hierarchical structure of the data and achieve robust performance in classifying traffic signs. The unique architecture of CNNs, with their local connectivity, parameter sharing, translation invariance, and pooling layers, enables them to outperform traditional neural networks when handling complex image classification tasks.

### *B. CNN Structure Manifested in LeNet-5*

The structure of a typical Convolutional Neural Network (CNN) consists of several key components, including convolutional blocks, flattening convolutional layers, fully connected layers, and softmax layers. LeNet-5, a pioneering CNN architecture developed by Yann LeCun et al., follows this structure and incorporates these components to effectively address image classification tasks. Here, we provide a theoretical explanation of each component and its practical implications in LeNet-5 [4]:

- **Convolutional Block**: A convolutional block consists of one or more convolutional layers followed by a max-pooling (subsampling) layer. This structure helps in learning the local features in the input images and reducing the spatial dimensions.
  - **Convolutional Layer**: The convolutional layer is the core building block of a CNN. It applies a set of filters (kernels) to the input image or feature maps, detecting local features and patterns. In LeNet-5, there are two convolutional layers, each responsible for learning different levels of abstraction in the input data.
  - **MaxPool Layer**: The max-pooling layer, also known as a subsampling layer, is used to reduce the spatial dimensions of the feature maps, while retaining important information. It aggregates local features by selecting the maximum value within a defined neighborhood. LeNet-5 employs max-pooling layers after each of the convolutional layers, resulting in a more compact and translation-invariant representation.
- **Normalization**: Normalization is an essential preprocessing step in CNNs, which ensures that the input values are on a similar scale, thus improving the stability and convergence of the learning process. Typically, normalization is applied after the max-pooling layer to scale the feature map values within a specific range, such as [0, 1] or [-1, 1]. This helps to mitigate the effect of large input variations, allowing the model to learn more effectively and generalize better to unseen data. In LeNet-5, normalization can be incorporated after each max-pooling layer to enhance the training process and improve the model's performance. [5]
- **Flattening Convolutional Layer**: After passing through the convolutional blocks, the feature maps are flattened into a one-dimensional vector. This step is essential for connecting the convolutional layers to the subsequent fully connected layers. In LeNet-5, the flattening layer is applied after the second max-pooling layer, reshaping the feature maps into a single vector.
- **Fully Connected Layer**: Fully connected layers are utilized to learn non-linear combinations of the features extracted from the convolutional layers. These layers are responsible for integrating the local features learned by the convolutional blocks to make final predictions. LeNet-5 incorporates two fully connected layers after the flattening layer, allowing the model to learn higher-level abstractions and relationships between the features.
- **Softmax Layer**: The softmax layer is used in the output layer of the CNN to convert the output of the previous fully connected layer into class probabilities. This layer employs the softmax activation function, which normalizes the output values to ensure they sum up to one, representing the probability distribution across the classes. In LeNet-5, the softmax layer is applied after the final fully connected layer, producing the final classification probabilities for the 43 traffic sign classes in the GTSB dataset.

By combining these components, LeNet-5 constructs a powerful CNN architecture capable of effectively learning hierarchical features from the input images and classifying traffic signs with high accuracy.

### *C. Final implemented structure*

The implemented CNN model based on the LeNet-5 architecture consists of the following structure:

The model is built using a sequential architecture, enabling the stacking of layers in a linear manner. The input layer is a 2D convolutional layer with a ReLU activation function, designed to process grayscale images. Additional 2D convolutional layers are added, featuring ReLU activation functions and increasing filter depth. All convolutional layers use a kernel size of (3,3). Max-pooling layers are incorporated to reduce the spatial dimensions of the feature maps, using a pool size of (2, 2) and stride of 1. Batch normalization layers are used to normalize the output along the channel axis, enhancing the stability and convergence of the learning process. A flattening layer reshapes the feature maps into a one-dimensional vector, preparing the input for subsequent fully connected layers. Fully connected layers with ReLU activation functions are employed to learn non-linear combinations of the extracted features. A dropout layer is included to reduce overfitting by

randomly dropping out neurons during training. The output layer is a fully connected layer with a softmax activation function, which produces class probabilities for the traffic sign classification task. This implementation adheres to the general structure of a CNN manifested in LeNet-5 described in the previous section, incorporating key components such as convolutional layers, max-pooling layers, batch normalization, dropout, and fully connected layers to effectively learn and classify traffic signs from the GTSB dataset.

## IV. Configurations, Processing and Adjustments

### A. Data preprocessing

In evaluating if our dataset required any notable preprocessing, [7] we established that an easy to implement and possibly usefull solution would be to reduce the dataset's complexity by transforming the images into a single color channel. Our mindset behind this was two-fold:

**Reducing computational load:** By converting the images from the original RGB format to grayscale, the number of color channels is reduced from three to one. This reduction significantly decreases the amount of data that the network needs to process. As a result, the computational load is lowered, leading to faster training times and reduced memory requirements.

**Minimizing overfitting:** Simplifying the images by focusing on a single color channel allows the model to concentrate on learning the most relevant features for traffic sign classification, such as shapes, edges, and textures. By discarding color information, the model is less likely to overfit to the training dataset. This process encourages the network to generalize better to unseen data, ultimately improving its performance on real-world traffic sign recognition tasks. Converting the images to a single color channel during the data preprocessing stage is a practical approach to enhance the model's training efficiency and performance while mitigating the risk of overfitting.

We also resized every image to a 30x30 format, thus allowing for the use of models that required square matrices.

### B. Cross validation

Holdout cross-validation was performed by splitting the train set into two distinct subsets: 70% of the data was used for training the model, and the remaining 30% was reserved for validation. This approach allows for the assessment of the model's performance on unseen data, which helps to evaluate its generalization capabilities. The holdout method is computationally efficient and straightforward to implement. However, it is important to note that the performance estimates obtained from holdout cross-validation can be sensitive to the specific choice of samples in the training and validation sets. In practice, multiple runs of holdout cross-validation with different random splits were used to obtain more reliable performance estimates.

### C. Configurations

In the context of our experiments with the Convolutional Neural Network (CNN) based on LeNet-5, the following choices were made as the baseline for our tests:

**Categorical Cross-Entropy:** As the problem involves multi-class classification (43 traffic sign classes), categorical cross-entropy was chosen as the loss function. Categorical cross-entropy measures the difference between the predicted probability distribution and the true probability distribution across all classes, making it suitable for multi-class problems.

**ReLU:** Rectified Linear Unit (ReLU) was used as the activation function in the convolutional and fully connected layers. ReLU is computationally efficient and helps mitigate the vanishing gradient problem, which can occur when training deep neural networks. It introduces non-linearity, allowing the model to learn complex patterns.

**Adam:** The Adam (Adaptive Moment Estimation) optimizer was chosen for model training. Adam is an adaptive learning rate optimization algorithm that combines the advantages of two popular optimization methods, AdaGrad and RMSProp. It is known for its efficiency and robustness, making it a popular choice for training deep learning models.

**30 epochs:** The model was trained for 30 epochs, meaning the entire training dataset was passed through the network 30 times. This number of epochs was chosen as a balance between computational time and the model's ability to learn the data patterns effectively.

**Learning Rate 0.001:** A learning rate of 0.001 was used in the Adam optimizer. The learning rate is a crucial hyperparameter that determines the step size during the optimization process. A smaller learning rate allows for more precise convergence to the optimal solution but may require more iterations, while a larger learning rate can lead to faster convergence but may risk overshooting the optimal solution.

**Batch Size 32:** The model was trained using a batch size of 32, which means that 32 samples were used to compute the gradient update in each iteration. Smaller batch sizes can provide a more accurate estimate of the gradient, while larger batch sizes can be more computationally efficient. A batch size of 32 is a common choice, as it balances the trade-off between computational efficiency and gradient accuracy. These choices serve as the baseline for your tests with the LeNet-5-based CNN, providing a starting point for evaluating the model's performance and allowing for further experimentation and optimization of hyperparameters to improve the model's accuracy on the GTSB dataset.

As previously mentioned in Section III. C. , our model has 2 convolutional blocks, a number we found that struck a balance between reliability and feasibility of execution.

## V. Support Vector Machines

### A. SVM as a point of comparison

Besides our final model, we experimented with a Support Vector Machine (SVM) based one, in order to have a point of comparison, tinkering with its different parameters to see how far we could take it.

SVM is a powerful supervised machine learning algorithm used for classification and regression tasks. The core idea behind SVM is to find the optimal hyperplane that maximizes

the margin between different classes. In the case of non-linearly separable data, SVM employs the kernel trick to project the data into a higher-dimensional space where a linear hyperplane can separate the classes. [8]

In the context of the traffic sign classification problem, using SVM as a point of comparison to the Convolutional Neural Network (CNN) model is beneficial for several reasons:

**Baseline performance:** SVM can serve as a baseline model to evaluate the performance of the CNN. By comparing the classification results of the CNN and SVM, one can determine whether the CNN's more complex architecture leads to a significant improvement in the classification task.

**Simplicity:** SVM is a relatively simple algorithm compared to a CNN. It requires less computational resources and is easier to implement. This simplicity makes it an appropriate point of comparison to assess whether the additional complexity of the CNN is necessary for this specific task.

**Effectiveness on small datasets:** SVMs are known to be effective on small and medium-sized datasets, which might be the case for some traffic sign classification tasks. By comparing the performance of the SVM and the CNN on the GTSB dataset, it becomes possible to determine if the CNN's ability to capture complex patterns and features outweighs the SVM's simplicity and effectiveness on smaller datasets. In summary, employing SVM as a point of comparison provides valuable insights into the necessity and effectiveness of the CNN architecture in the traffic sign classification problem. It helps to justify the use of CNNs and to assess whether their added complexity leads to improved performance over simpler algorithms such as SVM.

### *B. Usage of LBP and HOG features*

Local Binary Patterns (LBP) and Histogram of Oriented Gradients (HOG) are feature extraction techniques that capture different aspects of an image's structure. Both methods have been widely used in various computer vision tasks, such as object recognition and classification. In the context of the traffic sign classification problem, implementing LBP and HOG features provides several advantages: [9] [10] [11]

**Complementary information:** LBP and HOG capture distinct characteristics of an image. LBP is a texture descriptor that encodes local patterns and structures by comparing the intensity of a pixel with its neighbors. On the other hand, HOG is a gradient-based descriptor that captures the distribution of edge directions and edge intensities within an image.

**Robustness:** Both LBP and HOG features are known for their robustness against various image transformations, such as changes in illumination, scale, and rotation. Incorporating these features makes the classification model more resilient to variations in the input data, which is particularly important for real-world traffic sign recognition tasks, where lighting conditions and viewpoints can change rapidly.

**Reduced computational complexity:** By extracting LBP and HOG features from the images before feeding them into the classifier, the dimensionality of the input data is significantly reduced. This reduction not only decreases the computational load but also helps to mitigate the risk of overfitting by providing the classifier with a more compact and discriminative representation of the images.

In conclusion, the implementation of LBP and HOG features in the traffic sign classification problem allows the model to exploit the complementary information provided by each technique, enhances the model's robustness to variations in the input data, and reduces the computational complexity of the classification task. We use each exclusively and compare their performance.

### *C. Usage of linear, RBF, sigmoid, poly, and precomputed kernels*

In the context of Support Vector Machine (SVM) classification, the kernel function is a crucial component that determines the mapping of input data into a higher-dimensional space. Different kernel functions capture different types of relationships among data points, potentially leading to varying levels of performance depending on the nature of the data. The following kernel functions are commonly used in SVMs: [12]

**Linear:** The linear kernel is the simplest kernel function, representing a linear relationship between data points. It does not involve any transformation of the input data, making it computationally efficient and suitable for large-scale problems. However, its simplicity may limit its performance in cases where the data is not linearly separable.

**Radial Basis Function (RBF):** The RBF kernel is a popular choice for non-linear classification problems. It can model complex relationships among data points by projecting the input data into an infinite-dimensional space. The RBF kernel is particularly effective in capturing local patterns and has the advantage of having only one hyperparameter to tune (the Gaussian width).

**Sigmoid:** The sigmoid kernel is inspired by the activation function used in neural networks. It is capable of modeling non-linear relationships but can be sensitive to the choice of hyperparameters. In practice, the sigmoid kernel may not always perform as well as the RBF kernel for some problems.

**Polynomial:** The polynomial kernel can model a wide range of relationships, from linear to high-degree polynomial. It is parameterized by the degree of the polynomial and a scaling factor. While the polynomial kernel can be effective in capturing complex patterns, it may be computationally expensive for high-degree polynomials and sensitive to the choice of hyperparameters.

**Precomputed:** The precomputed kernel allows users to precompute the kernel matrix before training the SVM. This option is useful when a custom kernel function is used or when the kernel matrix can be computed more efficiently outside of the SVM. By experimenting with different kernel functions, one can identify the most suitable kernel for the traffic sign classification problem. The choice of kernel can have a significant impact on the classification performance, generalization capability, and computational efficiency of the SVM. By comparing the performance of various kernels, it becomes possible to determine the best kernel function for this

specific task and gain insights into the underlying structure of the traffic sign data.

### D. SVMs in the context of the GTSB problem

All the previous attributes were combined with each other with a C value of 100. We present only the statistically relevant results.

## VI. Results, comparisons and conclusions

### A. LeNet-5 and derivatives

Starting off with our most performant model, it served as a term of comparison which we sought to beat, but with **98.69%**, it simply was outstandingly good.

The following shows class accuracy histograms for the previously stated model, showing that the inital configuration stated in Section IV. C. is indeed optimal.

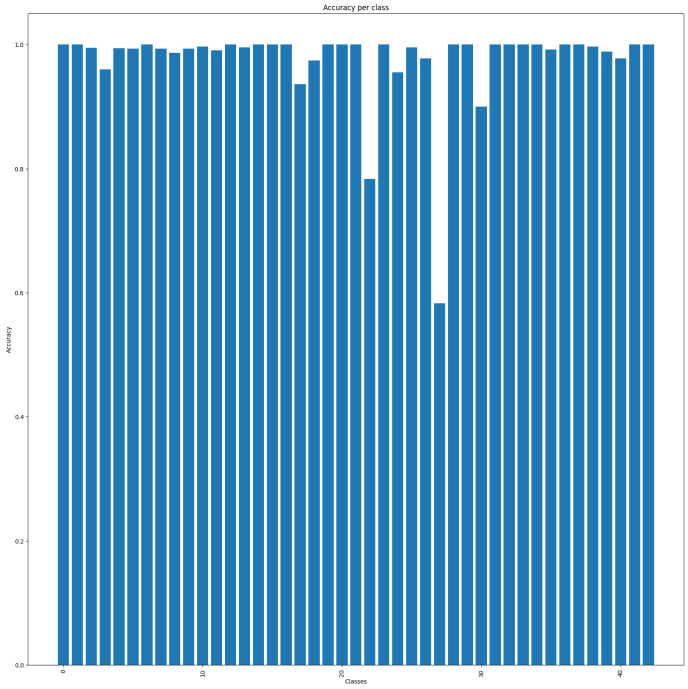


Fig. 4. LeNet 5 - Single Channel Colour - LR:0.001 EPOCHs:30

As can be observed, this model performs extremely well, with some exceptions related to the underrepresentation of certain classes in the data set.

Increasing the number of epochs beyond a certain point may not lead to improved performance as we see in Fig. 5 because the model may have already stabilized or reached a plateau. When the model stabilizes, it means that the model has already learned the underlying patterns and structures in the training data and further training does not significantly improve the model's ability to generalize to new data.

Continuing to train the model beyond this point of stabilization can lead to overfitting, where the model starts to learn the noise present in the training data, resulting in poor generalization to new, unseen data. Overfitting reduces the model's predictive performance and may even lead to a

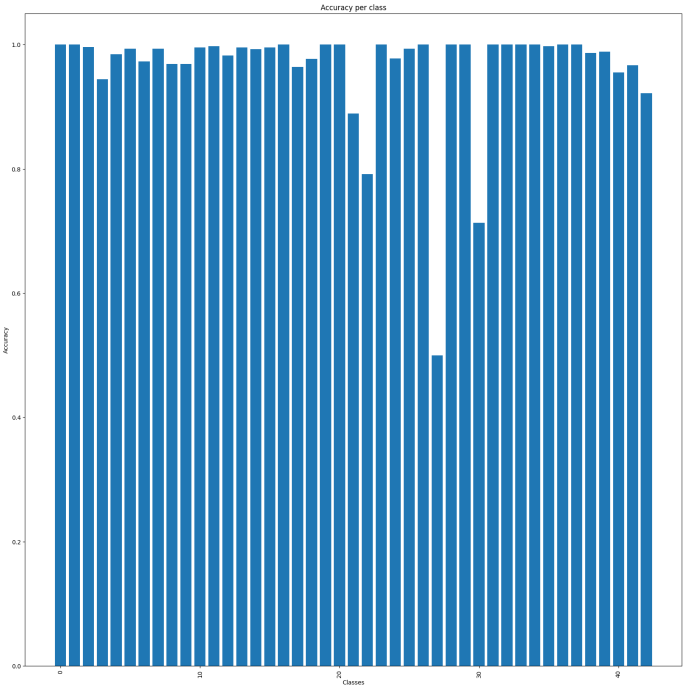


Fig. 5. LeNet 5 - Single Channel Colour - LR:0.001 EPOCHs:50

decrease in validation accuracy as we see for some classes in Fig. 5.

Monitoring the validation loss and accuracy during training can help in determining when the model has stabilized. If the validation loss stops decreasing and validation accuracy plateaus or starts to degrade, it is an indication that the model has reached its optimal performance, and further training may not be beneficial. In such cases, using techniques like early stopping can help prevent overfitting and ensure that the model maintains its generalization capabilities.

In Fig.6 we can see that this stablization has already happened at 30 EPOCHs.

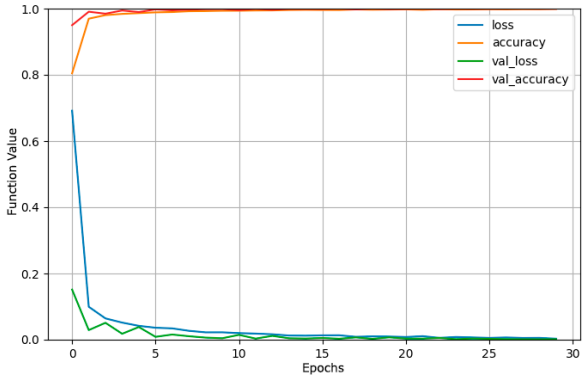


Fig. 6. Cost Function Evolution Over EPOCHs

### B. SVM and derivatives

With linear SVM models not producing significant data and linear ones precomputed ones requiring square matrices through and through, when, do to convolution and zero padding techniques, our data is molded throughout the network, we were left to compare the other 3 kernel types with HOG and LBP features. RBF was the most performant, followed closely by polynomial and trailed by the sigmoid kernel.

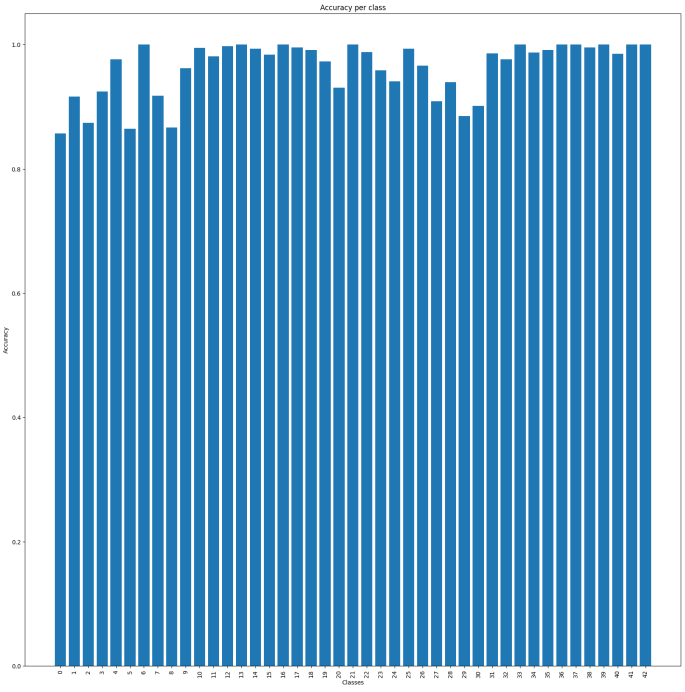


Fig. 7. RBF SVM with HOG Features

We can conclude with certainty that the use LBP features hinders massively the performance of a model in the context of this problem, as all models were heavily punished as result of its use as can be seen in Figures 10, 11 and 12, compares to their HOG counterparts.

This leads us to conclude that RBF with HOG features is the ideal case scenario for a SVM model in this context.

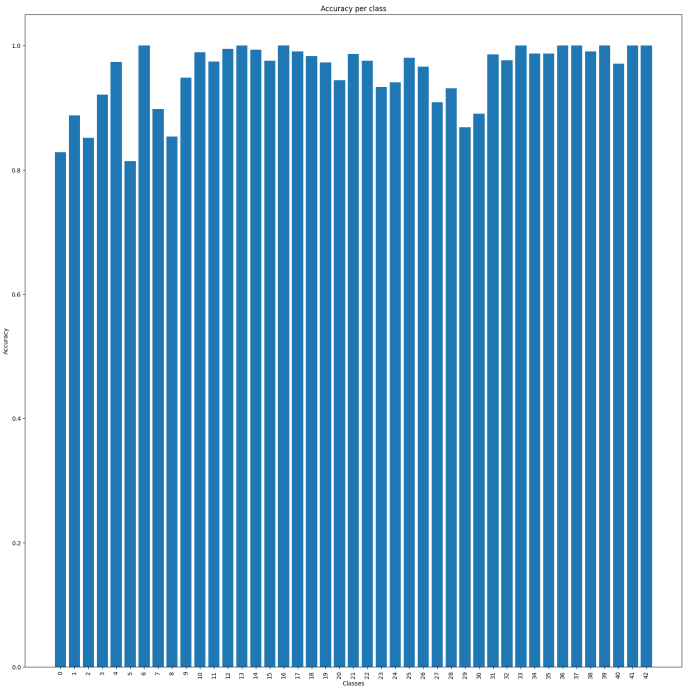


Fig. 8. Polynomial SVM with HOG Features

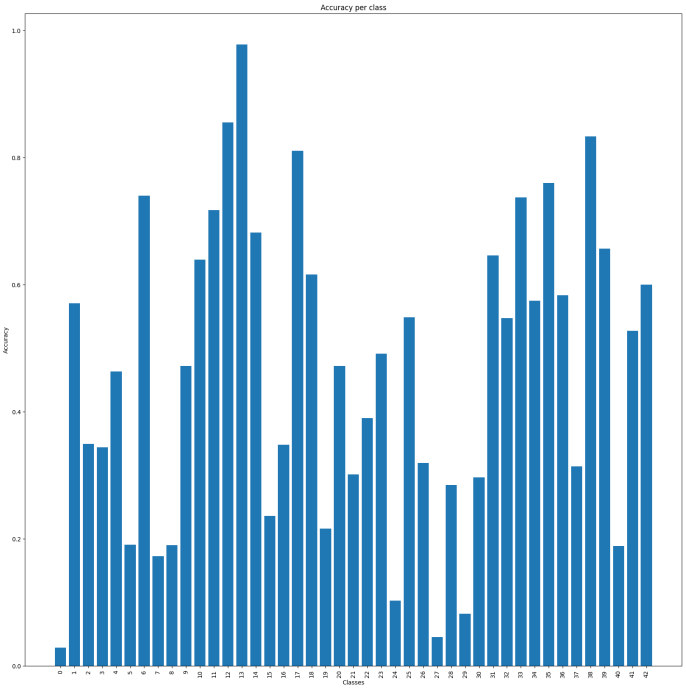


Fig. 9. Sigmoid SVM with HOG Features

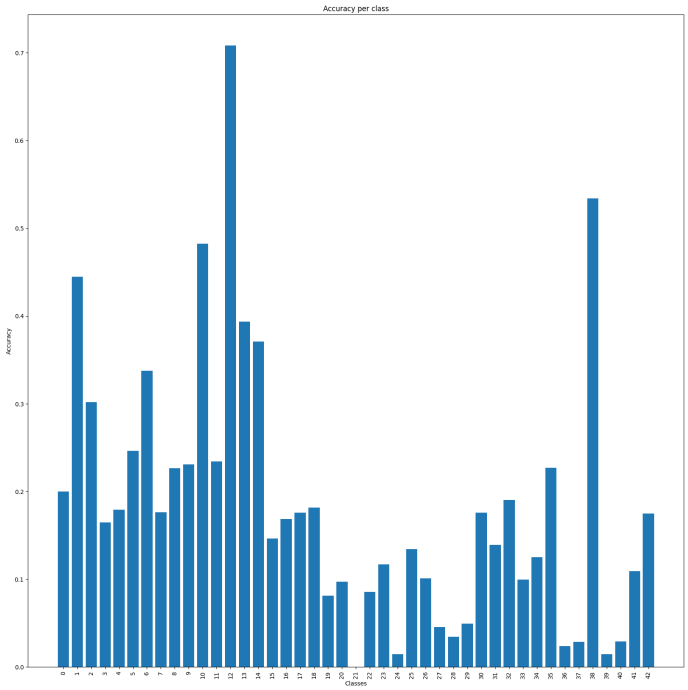


Fig. 10. RBF SVM with LBP Features

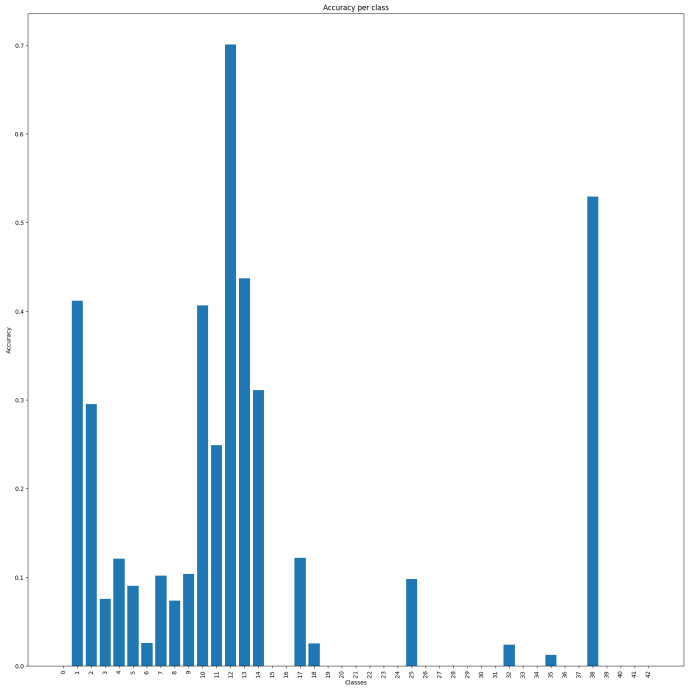


Fig. 11. Polynomial SVM with LBP Features

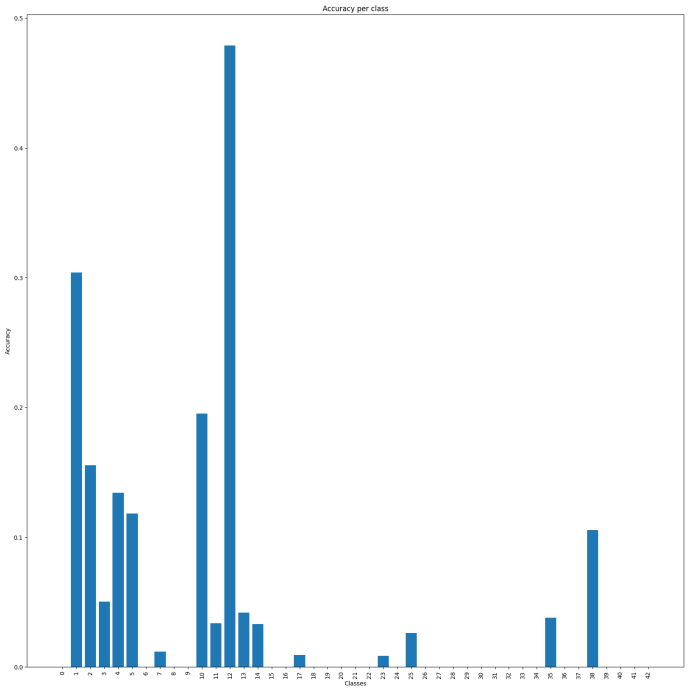


Fig. 12. Sigmoid SVM with LBP Features

### *C. Top model comparison*

With all this baggage, we can compare our most accurate models using their respective confusion matrices in order to more adequately rate their performance. Confirming our supposition that our original model is indeed superior.

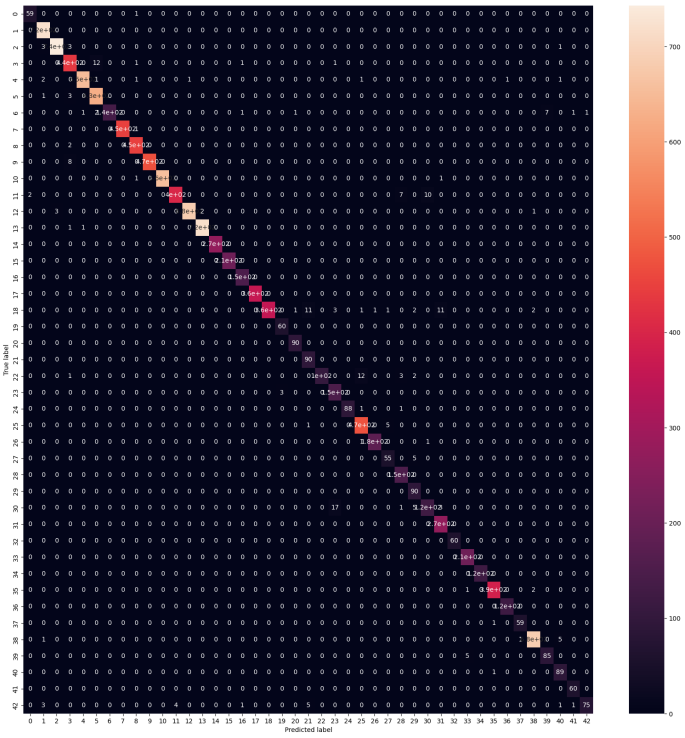


Fig. 13. Confusion Matrix for Optimal Grayscale LeNet-5 Model

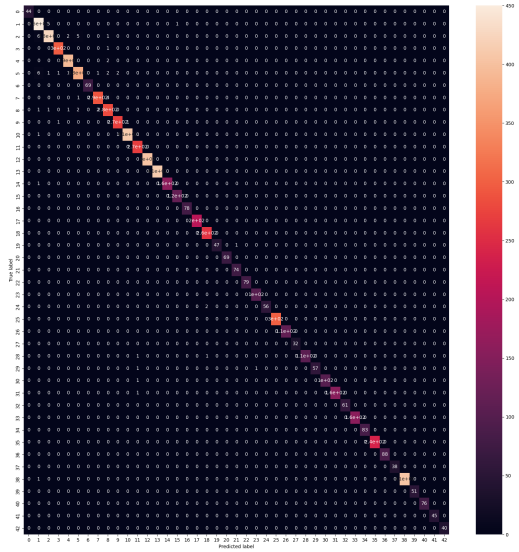


Fig. 14. Confusion Matrix an SVM RBF with HOG Features Model

## VII. Conclusions and Future Work

In conclusion, the use of Convolutional Neural Networks (CNNs), specifically the LeNet-5 architecture, has proven to be an effective approach for traffic sign classification on the German Traffic Sign Benchmark (GTSB) dataset. The model achieved an impressive accuracy of 98.69%, demonstrating its ability to recognize and categorize various traffic signs with high precision. This success highlights the potential of CNNs in tackling complex image classification tasks, particularly in the domain of autonomous vehicles and advanced driver-assistance systems.

Despite the remarkable performance, there is always room for improvement. In the future, the model could be further refined by exploring alternative architectures, fine-tuning hyperparameters, or employing advanced regularization techniques such as dropout and batch normalization in different configurations. Additionally, data augmentation methods could be employed to increase the dataset's diversity, making the model more robust to variations in lighting conditions, occlusion, and other real-world challenges. [6]

Moreover, exploring the potential of transfer learning, where pre-trained models on large-scale datasets are fine-tuned for the specific task of traffic sign classification, may lead to even higher accuracy levels and improved generalization capabilities. Finally, incorporating the model into a real-time system and evaluating its performance in real-world scenarios would provide valuable insights into its practical applicability and potential challenges.

In summary, the current study demonstrates the power and effectiveness of CNNs in traffic sign classification, achieving a high accuracy rate of 98.69%. With further research and refinements, this approach could lead to even more accurate and robust traffic sign recognition systems, contributing significantly to the advancement of intelligent transportation systems and autonomous vehicles.

## References

[1] Scanlon, J. M., Kusano, K. D., Engström, J., and Victor, T. (n.d.). Collision Avoidance Effectiveness of an Automated Driving System Using a Human Driver Behavior Reference Model in Reconstructed Fatal Collisions. Waymo, LLC.

[2] Temraz, M., and Keane, M. T. (2021). Solving the Class Imbalance Problem Using a Counterfactual Method for Data Augmentation. arXiv preprint arXiv:2111.03516.

[3] O'Shea, K., and Nash, R. (2015). An Introduction to Convolutional Neural Networks. arXiv preprint arXiv:1511.08458.

[4] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE, 86, 2278-2324. doi:10.1109/5.726791

[5] Barney Kim, Traffic Sign Recognition, 2019

[6] Shekhar Bavanari, Traffic Sign Detection using Deep Learning, 2019

[7] Famili, A. et al. 'Data Preprocessing and Intelligent Data Analysis'. 1 Jan. 1997 : 3 – 23.

[8] Evgeniou, Theodoros & Pontil, Massimiliano. (2001). Support Vector Machines: Theory and Applications. 2049. 249-257. 10.1007/3-540-44673-7-12.

[9] Yang Z. and Pun-Cheng L. S. C. 2018 Vehicle detection in intelligent transportation systems and its applications under varying environments: A review Image and Vision Computing 69 143-154

[10] Cao L., Jiang Q., Cheng M. and Wang C. 2016 Robust vehicle detection by combining deep features with exemplar classification Neurocomputing 215 225-231

[11] Hadi R. A., Sulong G. and George L. E. 2014 Vehicle Detection and Tracking Techniques : A Concise Review Signal & Image Processing : An International Journal 5 1-12
[12] Apostolidis-Afentoulis, Vasileios. (2015). SVM Classification with Linear and RBF kernels. 10.13140/RG.2.1.3351.4083.