TRAFFIC SIGNAL CLASSIFICATION USING CNN

SREE SANKAR S | https://github.com/sreesankar711

OBJECTIVES

Implementation: To implement a CNN model that can detect and recognize traffic signals present in images.

Accurate Classification: The CNN-based traffic signal classifier should be able to accurately classify different traffic signals, even in complex and challenging scenarios such as low light, adverse weather conditions, and heavy traffic.

Real-Time Performance: The classifier should be able to process traffic signals in real time without significant delays. This is essential for ensuring that the traffic signals are detected and classified in time to inform drivers and ensure safe driving.

Robustness: The classifier should be robust to variations in lighting conditions, occlusion, and other challenging scenarios. This is important to ensure the classifier can accurately classify traffic signals in various real-world situations.

LITERATURE SURVEY

These are some of the alternative algorithms proposed for the problem

Using computer vision feature extraction methods

Scale Invariant Feature Transform

This was one of the early approaches that helped scientists implement different algorithms. A technique like Histogram Oriented Gradients was initially popularized for detecting pedestrians. In this method, gradients of color images are computed along with different normalized, weighted histograms.**[4]** Scale Invariant Feature Transform (SIFT) technique is used for classification, and the sliding window approach is used to perform both classification and detection tasks simultaneously.

Using machine learning

Linear Discriminate Analysis

[5] This approach is based on Linear Discriminant Analysis that performs dimensionality reduction and improves class separability. The tests were performed on the German Traffic Sign Recognition Benchmark, using a Multi-Layer Perceptron as a classification tool. LDA classification and k-Nearest Neighbors were also used for comparison. This achieved a maximum accuracy of 96.5%.

Using Deep Learning

Rotation Invariant Binary Pattern Based Feature

[6] Here, a typical Hough transformation is adopted to implement coarse-grained location of the candidate regions of traffic signs. Second, a RIBP (Rotation Invariant Binary Pattern) based feature in the affine and Gaussian space is used to reduce the time of traffic sign detection and achieve robust traffic sign detection in terms of scale, rotation, and illumination. Third, the techniques of ANN (Artificial Neutral Network) based feature dimension reduction and classification are designed to reduce the traffic sign recognition time. A classification accuracy of 98.62% was obtained.

BACKGROUND

Convolutional Neural Networks (CNNs): CNNs are a type of deep neural network that is widely used for image recognition and classification. They work by extracting features from an image at different levels of abstraction through a series of convolutional layers. CNNs are very effective in traffic signal classification because they can learn and generalize complex patterns from images.

Image Preprocessing: Image preprocessing involves several techniques, such as image resizing and normalization. These techniques prepare the images for classification by reducing noise and enhancing their features.

Data Augmentation: Data augmentation involves applying transformations such as rotation, translation, and flipping to the training images to increase the diversity of the dataset. This technique helps to improve the generalization ability of the CNN and reduces the risk of overfitting.

Optimization Algorithms: Optimization algorithms such as Adam (Adaptive Moment Estimation) and Stochastic Gradient Descent (SGD) are used to optimize the weights of

the CNN during training. These algorithms help to minimize the loss function and improve the accuracy of the classifier.

Activation Functions: Activation functions such as ReLU (Rectified Linear Unit) and Sigmoid introduce non-linearity into the CNN. These functions enable the CNN to learn complex patterns and improve the accuracy of the classifier.

METHODOLOGY

Model Selection

The model selection for traffic signal classification involves choosing a suitable neural network architecture to classify traffic signals accurately. In this case, a Convolutional Neural Network (CNN) architecture has been selected, which is commonly used for image classification tasks.

The model is a more complex and modern variant of the LeNet-5 architecture, with more layers, larger filter sizes, and modern techniques like dropout regularization.**[3]** LeNet-5 is a classic convolutional neural network architecture proposed by Yann LeCun et al. in 1998. The model was designed for handwritten digit recognition and comprised several convolution, pooling, and fully connected layers.

Here, the number of filters used in the convolutional layers of the model is much higher than in LeNet-5. Also, our model includes dropout layers, which are used for regularization and to prevent overfitting. LeNet-5 uses a "tanh" activation function in the fully connected layers, whereas this model uses "relu".

Another notable difference is the number of neurons in the fully connected layers. The model used here has significantly more neurons in the fully connected layers than LeNet-5, which allows it to learn more complex and abstract features from the input data.

Procedure

Data collection: The German Traffic Sign Recognition Benchmark (GTSRB) dataset is used here. It covers various traffic signals, angles, and lighting conditions. It also has .csv files containing paths to training and testing datasets and their corresponding labels. Both are added to their corresponding numpy arrays, which are more memory-efficient as they allow for contiguous storage of data in memory, making it faster to access and manipulate data.

Data preprocessing: The images are added to the numpy array by resizing them to a consistent size (32 x 32). Later, the pixel values are normalized before the data is fed into the model.

Model selection: An appropriate model architecture is selected for the classification task. Here we used a custom CNN model built from scratch.

Model training: The selected model was trained on the preprocessed data using categorical cross-entropy as the loss function and the Adam optimizer. In addition, the ModelCheckpoint callback is used to save the best model weights during training, which are stored in the 'model.h5' file. This ensures that the model with the best performance on the validation set is saved for later use.

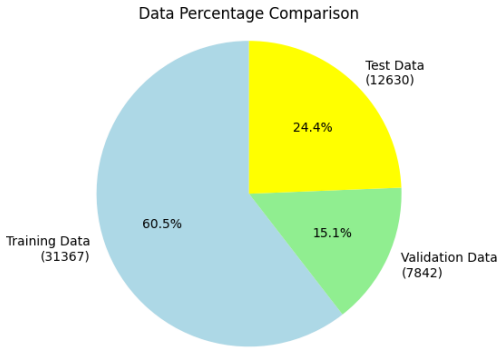
Model evaluation: The model is evaluated on the test data set to ensure it can adequately classify the traffic signals and to fine-tune the model later. Metrics such as accuracy, precision, recall, and F1 score are used to evaluate the model, and also the confusion matrix is plotted. Images from the Meta directory are also used to test, and the predicted results and the confidence of prediction are also plotted.

EXPERIMENTAL SETUP

Dataset: [2] German Traffic Sign Recognition Benchmark (GTSRB) dataset is used. It is a publicly available dataset created from 10 hours of video of driving on different roads in Germany. The dataset contains a total of 51,840 images of the 43 classes.

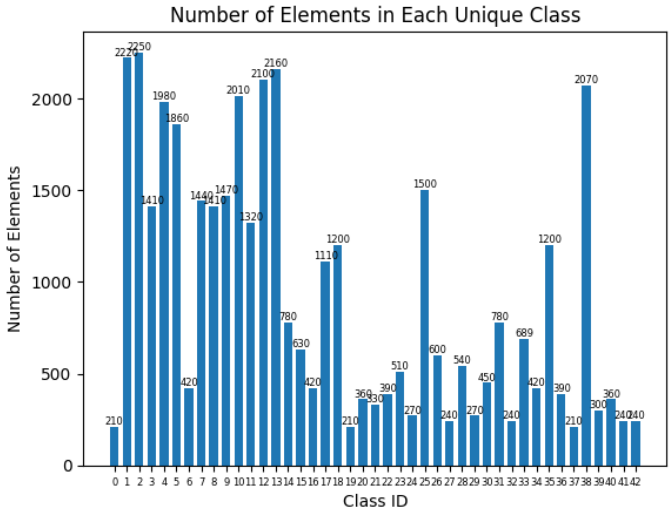
Images from different classes in GTSRB

The dataset also contains .csv files containing paths to training and testing datasets and their corresponding labels. Images are of different sizes, and the total dataset is divided into training and testing data. A total of 39,209 images are present as training data and 12,630 images as test data. The training dataset is further divided into training and validation set on an 80:20 split such that there are 31367 training images and 7842 validation images



Pie chart showing data distribution

The primary challenge associated with the German Traffic Sign Recognition Benchmark (GTSRB) is its biased dataset. Biased datasets do not accurately represent the true population and can lead to models that are trained on such datasets to be discriminatory towards certain groups. Data augmentation can help to some extent in reducing the impact of bias in a dataset. By creating additional variations of the existing data, it can help to increase the diversity of the dataset and reduce over-reliance on a particular subset of the data.



German Traffic Sign Recognition Benchmark (GTSRB) suffers from bias due to imbalanced class distribution

Training and Testing Strategies

Data Preparation: The data is read from the disk and is converted into numpy arrays. Different numpy arrays are used for Training and Testing datasets along with their labels. Each image is resized into 32*32 for efficient training and better model performance. A total of 31367 images as training data, 7842 as validation data, and 12,630 as test data are used.

Preprocessing: For better performance and to prevent overfitting, the training dataset is augmented five-fold by replicating the available with data rotation ± 15, shear range of 0.1, width shift range of 0.1, and height shift range of 0.1, which increases the training dataset size leading to better performance, and regularization thus avoiding the overfitting problem. Augmenting the existing training dataset will increase the dataset size to 31367 x 5 = 156835 images.

Network architecture: The network architecture implemented here is a Convolutional Neural Network designed for image classification tasks. It consists of multiple convolutional layers with increasing filter sizes followed by max-pooling layers for downsampling. The convolutional layers help to extract relevant features from the input images.

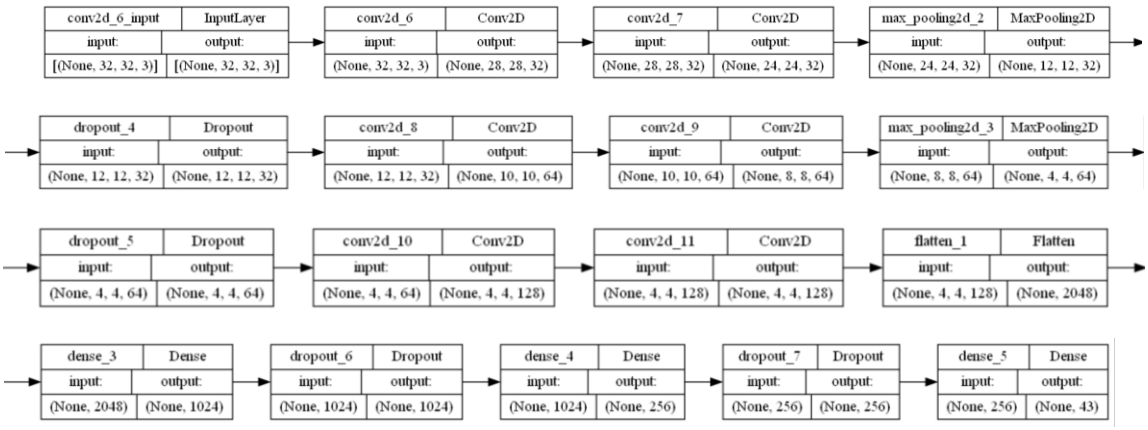
The first layer in the network has 32 filters of size 5x5, followed by a second layer with the same number of filters and sizes. This is followed by a max pooling layer that reduces the spatial dimension of the feature maps by a factor of two. A dropout layer with a rate of 0.15 is added after the max pooling layer to prevent overfitting.

The subsequent convolutional layers have 64 filters of size 3x3, followed by another max pooling layer and a dropout layer with a rate of 0.15. The third set of convolutional layers has 128 filters of size 3x3 with padding set to "same". Padding ensures that the output feature maps have the exact spatial dimensions as the input feature maps. A dropout layer also follows these convolutional layers.

Finally, the output from the convolutional layers is flattened and passed through several fully connected layers with increasing numbers of neurons. Using fully connected layers helps to learn more complex relationships between the extracted features. The last layer uses a softmax activation function to output the predicted probabilities for each of the 43 classes.

Using dropout layers at various stages of the network helps prevent overfitting and improve generalization performance. Additionally, the use of "relu" activation function in the

convolutional layers helps to avoid the vanishing gradient problem, which can occur with other activation functions such as "tanh" and "sigmoid".



Convolutional neural network architecture

Training: The training process is essential for optimizing the model weights to achieve better performance on the task. The number of epochs, batch size, and choice of optimizer are crucial hyperparameters that must be tuned carefully to achieve optimal performance.

The model is trained for 50 epochs, which means that the model will go through the entire training data 50 times. During each epoch, the model will update its weights based on the gradients computed by the optimizer. The optimizer used in this case is Adam.

During training, the model is evaluated on the validation set. This helps monitor the model's performance on unseen data and detect overfitting. The "callbacks" argument saves the best-performing model in a file called "model.h5".

Testing: After training the traffic signal classifier, the next step is to evaluate the model's performance on the test dataset. This is typically done by making predictions on the test dataset and comparing the predicted labels with the ground truth labels.

Several metrics were used to evaluate the performance of the classifier. These metrics included precision, recall, F1 score, and accuracy. The precision score measures the proportion of correctly identified positive instances among all instances predicted as positive. The recall score measures the proportion of correctly identified positive instances among all actual positive instances. The F1 score is the harmonic mean of precision and

recall. Finally, the accuracy score measures the proportion of correctly classified instances among all instances.

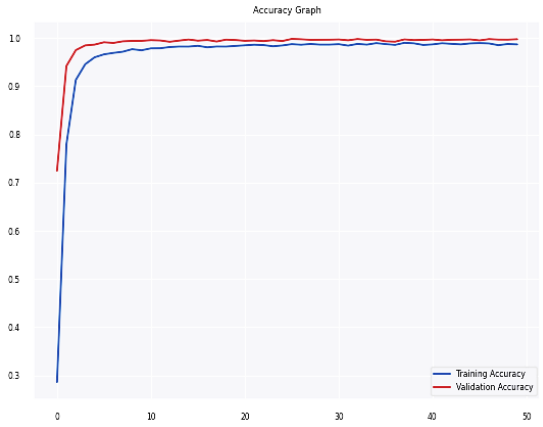
In addition to these metrics, a confusion matrix was also generated to provide further insights into the classifier's performance. The confusion matrix is a table that summarizes the test data's predicted and actual class labels. It can be used to determine which classes were frequently misclassified and identify potential areas for improvement.

To gain a better understanding of the model's confidence in its predictions, confidence plots were generated. Confidence plots show the confidence scores of the classifier for each test image. They can identify images for which the classifier is uncertain or has low confidence.

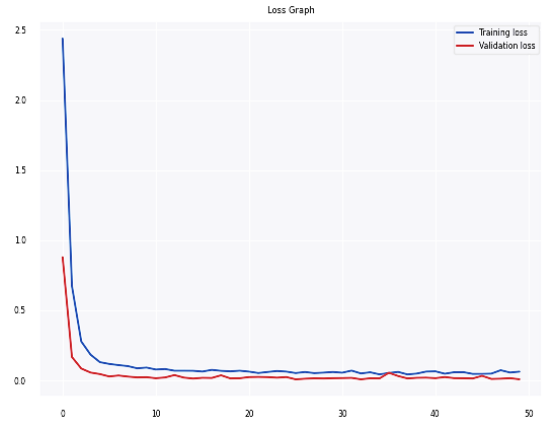
RESULT

The model was trained for 50 epochs. Comparing with the other works, our method has good performance on the GTSRB dataset.

Accuracy of the model is evaluated using the testing dataset of 12630 images. It can be calculated as the ratio of the correctly identified traffic signs to the total number of traffic signs. The traffic signal classifier achieved an overall accuracy of 98.46%. The precision score, indicating the proportion of correctly classified positive instances out of all predicted positive instances, was 97.39%. The recall score, representing the proportion of correctly classified positive instances out of all actual positive instances, was 97.96%. The F1 score, which is the harmonic mean of precision and recall, was 97.63%. These results suggest that the classifier performed well in correctly identifying traffic signals in the test dataset.

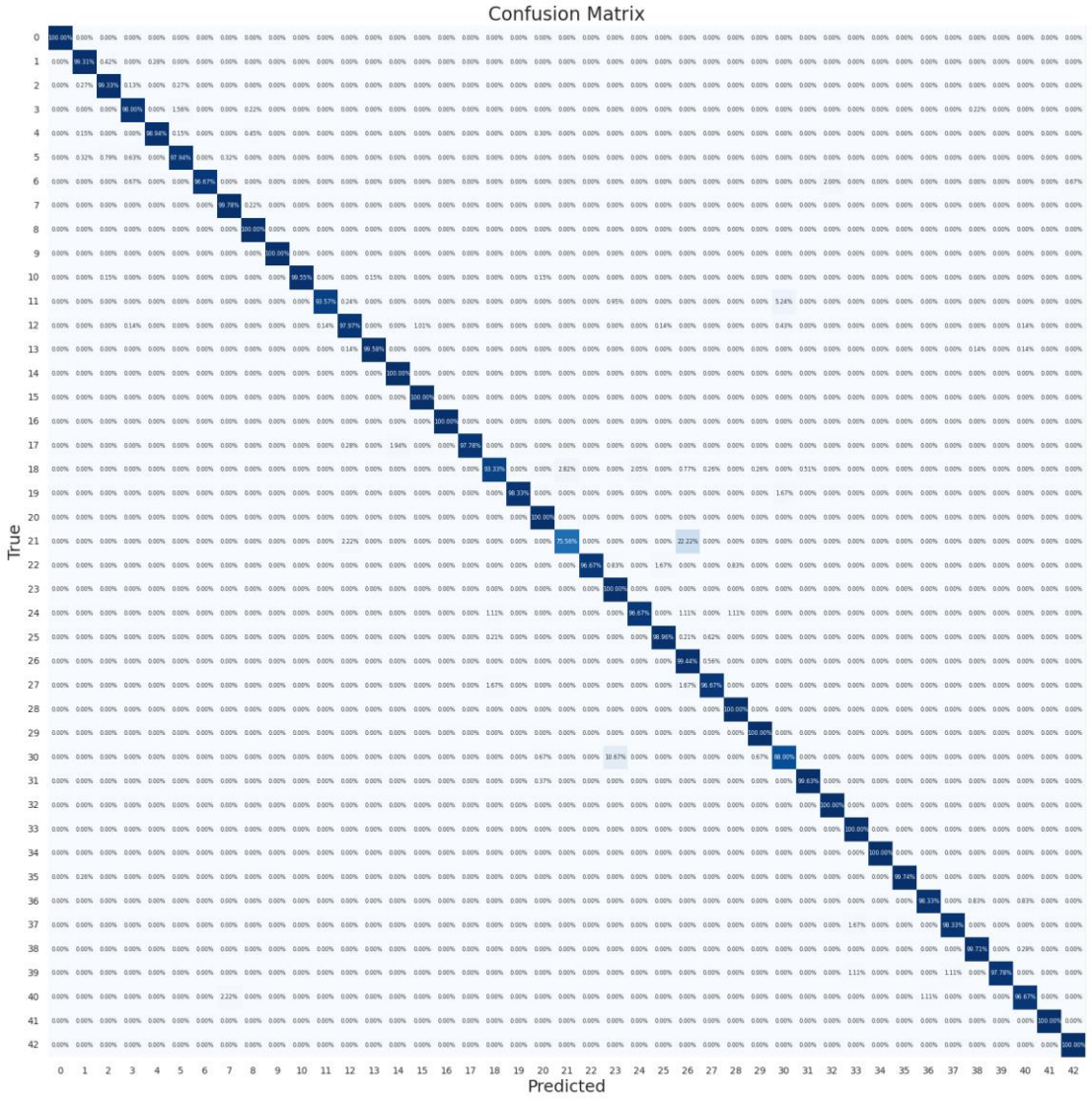


The training and validation accuracies over epochs.



The training and validation loss over epochs

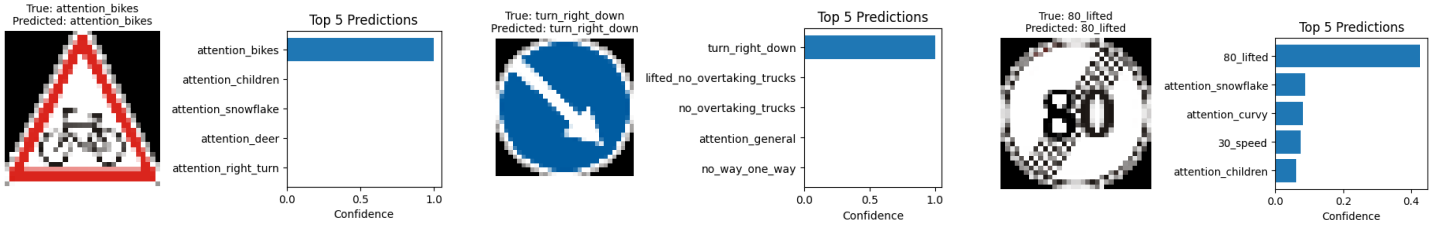
The confusion matrix also provides further insights into the classifier's performance. The confusion matrix summarizes the predicted and actual class labels of the test data. It can be used to determine which classes were frequently misclassified and identify potential areas for improvement.



The intensity of color blue through the diagonal represents that most of the classes were predicted correctly.

Using Images from the Meta directory, the model was used to predict the class labels. It was also used to plot the confidence in the prediction.

Predicted vs True Class labels



(a) (b) (c)

Confidence in prediction of class label

Here we can see that the model predicts the class labels of the images with high confidence.

By reducing the number of epochs during model training, the model may need more time to learn the complex patterns in the data. This phenomenon, known as underfitting, can result in lower accuracy scores and increased incorrect predictions. A smaller number of epochs can lead to insufficient learning by the model, which may compromise its ability to make accurate predictions. When the epoch is set very high, the model has more time to learn the patterns in the data, which may result in higher accuracy scores. However, setting the epoch too high may lead to overfitting the model, where the model becomes too complex and starts to learn the noise in the data, resulting in lower accuracy on new, unseen data. In addition, a high number of epochs can also lead to longer training times and the risk of the model getting stuck in local minima.

By setting the number of epochs to 10 and keeping all other parameters the same, we obtained an Accuracy score of 96.12%, a Precision score of 94.90%, a Recall score of 94.91%, and an F1 score of 94.67%. This suggests that the model was not able to fully capture the patterns in the data, resulting in a slightly lower performance

Using a simpler deep learning model comprising of two sets of convolutional layers with 32 and 64 filters, respectively, each followed by maxpooling and dropout layers, and two dense layers with 256 and 43 neurons, respectively, also followed by a dropout layer with a rate of 0.5, we were able to achieve an accuracy of 97.61% on the given dataset. So we can see that a simpler model can only capture some of the underlying patterns in the data. At the same time, a highly complex model may overfit the training data, performing well on it but poorly on new, unseen data. It is also computationally expensive, making it impractical for real-world applications. Therefore, the key is to find the right balance between model complexity and performance for a given task

CONCLUSION

In conclusion, the implementation of the traffic sign classification model based on the German Traffic Sign Recognition Benchmark dataset has demonstrated high accuracy and performance. By using convolutional neural networks and data augmentation techniques, the model was able to classify traffic signs with an accuracy score of 98.46%. These results show the effectiveness of the proposed model in tackling the challenge of traffic sign detection. The model can be further improved by incorporating more advanced techniques, such as transfer learning and ensembling, and can be applied to real-world scenarios for enhancing road safety and traffic management.

REFERENCES

[1] Y. Wu, Y. Liu, J. Li, H. Liu and X. Hu, "Traffic sign detection based on convolutional neural networks," The 2013 International Joint Conference on Neural Networks (IJCNN), Dallas, TX, USA, 2013

https://ieeexplore.ieee.org/document/6706811

[2] J. Stallkamp, M. Schlipsing, J. Salmen and C. Igel, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," The 2011 International Joint Conference on Neural Networks, San Jose, CA, USA, 2011, pp. 1453-1460, doi: 10.1109/IJCNN.2011.6033395.

https://ieeexplore.ieee.org/document/6033395

[3] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

https://ieeexplore.ieee.org/document/726791

[4] Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* **60**, 91–110 (2004).

https://doi.org/10.1023/B:VISI.0000029664.99615.94

[5] Gonzalez-Reyna, Sheila & Avina-Cervantes, Juan Gabriel & Ledesma, Sergio & Cruz, Ivan & Garcia-Hernandez, Ma. (2013). Traffic Sign Recognition Based on Linear Discriminant Analysis. 185-193. 10.1007/978-3-642-45111-9_16.

https://www.researchgate.net/publication/261098957_Traffic_Sign_Recognition_Based_on_Linear_Discriminant_Analysis

[6] Shouyi Yin, Peng Ouyang and Leibo Liu et al. Fast Traffic Sign Recognition with a Rotation Invariant Binary Pattern Based Feature. *Sensors (Basel, Switzerland).* 2015. Vol. 15(1):2161-2180. DOI: 10.3390/s150102161

https://www.scienceopen.com/document/read?vid=e6fcbb32-3482-46c0-b860-8db7161c6650