MACHINE LEARNING MODEL FOR TRAFFIC SIGN RECOGNITION

Prestone Jeremiah Simiyu

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Information Technology of Masinde Muliro University of Science and Technology

October 2025

PLAGIARISM STATEMENT

Student Declaration

1. I hereby declare that I know that the incorporation of materials from other works or a paraphrase of such material without acknowledging will be treated as plagiarism according to the Rules and Regulations of Masinde Muliro University of Science and Technology.
2. I understand that this thesis must be my work.
3. I know that plagiarism is academic dishonesty and wrong, and that if I commit any act of plagiarism, my thesis can be assigned a failing grade ("F").
4. I further understand that I may be suspended or expelled from the University for Academic Dishonesty.

Signature: _____ Date: _____

Student's name: Prestone Jeremiah Simiyu

Registration Number: SIT/H/01-70251/2022

Supervisors' Declaration

We hereby approve the examination of this thesis. This thesis has been subjected to a plagiarism test and its similarity index is not above 20%.

Signature: _____ Date: _____

Dr. Raphael Angulu

Department of Computer Science

School of Computing and Informatics

Masinde Muliro University of Science and Technology

Signature: _____ Date: _____

Dr. Daniel Otanga

Department of Information Technology

School of Computing and Informatics

Masinde Muliro University of Science and Technology

COPYRIGHT

DEDICATION

To Maurice Simiyu; “Father, hopefully your vehicle will be safer”

DECLARATION

This thesis is my original work prepared with no other than the indicated sources and support and has not been presented elsewhere for any other award.

Signature…….. Date……

Prestone Jeremiah Simiyu

SIT/H/01-70251/2022

The undersigned certify that they have read and hereby recommend for acceptance of Masinde Muliro University of Science and Technology proposal entitled ***"Machine Learning Model for Traffic Sign Recognition"***

Signature …….. Date……

Dr. Raphael Angulu

Department of Computer Science

School of Computing and Informatics

Masinde Muliro University of Science and Technology

Signature …….. Date……

Dr. Daniel Otanga

Department of Information Technology

School of Computing and Informatics

Masinde Muliro University of Science and Technology

ACKNOWLEDGEMENT

Glory to God for the good health and protection throughout out the journey of this research. My supervisor Dr. Raphael Angulu and Dr. Otanga Daniel played a constant and important role of supporting and guiding this research. My dear parents Juliet and Maurice Simiyu without your guidance, I would not have been able to find my way to this point; thank you. My wife Ladyisar Simiyu, siblings Cynthia and Lydia and my Kids Avianne and Alpha Simiyu I will forever be grateful for your support and prayers.

ABSTRACT

Autonomous (driverless) cars are increasingly becoming popular, hence calling for robust Traffic Sign Recognition (TSR) systems to ensure road safety. A report by World Health Organization Road Safety Report of 2018, shows that failure to distinguish and recognize traffic signs is among the leading causes of accidents. Existing TSR systems are adversely affected by environmental conditions, partial occlusion of traffic sign, illumination, colour deterioration because of their exposure to different rays including Ultra-Violet (UV), physical deformation, variations in pictogram designs and weather conditions among others. The study was guided by the following main objective; to develop a robust model using machine learning for recognition of traffic signs. Deductive research approaches, was used to achieve the following specific objectives: to analyses existing techniques in TSR, to design an advanced feature extraction technique for robust TSR and develop a machine-learning model for recognition of traffic signs. The new Local Directional Histogram of Oriented Gradient (LD-HOG) feature extractor is the main contribution of this work. A more resilient and discriminative descriptor is produced by combining the directional resilience and noise invariance of the Local Directional Pattern (LDP) with the potent gradient magnitude representation of HOG. The German Traffic Sign Detection Benchmark (GTSRB) dataset, was used to extract features. With an average F1-score of 96.5% using an SVM classifier, LD-HOG outperformed HOG by 4.2% and LDP by 7.8%. By helping to create more accurate and dependable advanced driver-assistance systems, the study will benefit a variety of stakeholders and road users, including drivers, passengers, and legislators., and policy makers.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

Equation 2.16 - Final normalization

Equation 2.17 - Resulting transformation after normalization

Equation 2.18 - Generalized dilated convolution operator $*_l$

Equation 2.19 - Dilated convolutions

Equation 2.20 - Dilation filters

Equation 2.21 - Activity of a neuron

Equation 2.22 - ReLu in AlexNet

Equation 2.23 - ResNet

Equation 2.24 - Residual unit

Equation 2.25 - Feature Extraction starting at (l) up to (L) layers

Equation 2.26 - Loss function of the first factor

Equation 2.27 - Faster R-CNN

Equation 2.28 - Bounding-box regression for t_x

Equation 2.29 - Bounding-box regression for t_x^*

Equation 2.30 - Interpolation weights along the x and y axes

Equation 2.31 - Bilinear interpolation for w and h for equation 2.1

OPERATIONAL DEFINATION OF TERMS

Machine Learning Model – are computer algorithms that can automatically learn from data and improve their performance on a specific task without being explicitly programmed.

Feature Extracting Technique – A subfield of machine learning that involves the automatic extraction of useful features or patterns from raw data.

Traffic Sign Recognition - Refers to the task of categorizing images or videos of Traffic Signs into predefined classes based on the structures presented in the Image.

ABBREVIATIONS AND ACRONYMS

ADAS	Advanced Driving Assistance Systems
ANN	Artificial Neural Network / Approximate Nearest Neighbor
API	Application Programming Interface
ARA	Aspect Ratio Adjustment
BOW	Bag-of-Words
CAS	Content-Aware Scaling
CBIR	Content-Based Image Retrieval
CNNs	Convolutional Neural Networks
DoG	Difference-of-Gaussian
DPS	Directorate of Postgraduate Studies
DT	Decision Tree
DT-CWT	Dual-Tree Complex Wavelet Transform
ECOC	Error-Correcting Output Code
FEC	Feature Extraction and Classification

FCN	Fully Convolutional Network
fMRI	Functional Magnetic Resonance Imaging
GLCM	Gray Level Co-occurrence Matrix
HOG	Histogram of Oriented Gradients
HSV	Hue, Saturation, and Value
KNN	K-Nearest Neighbors
LBP	Local Binary Patterns
LD-HOG	Local Directional Histogram of Oriented Gradient
LDA	Linear Discriminant Analysis
LRN	Local Response Normalization
LSTM	Long Short-Term Memory
MMUST	Masinde Muliro University of Science and Technology
MSERs	Maximally Stable Extremal Regions
NNI	Nearest Neighbor Interpolation
NRF	National Research Fund of Kenya

PCA	Principal Component Analysis
ReLU	Rectified Linear Unit
RF	Random Forest
RGB	Red, Green, and Blue
RI-LBP	Rotation-Invariant Local Binary Patterns
RNNs	Recurrent Neural Networks
ROI	Region of Interest
SGD	Stochastic Gradient Descent
SIFT	Scale-Invariant Feature Transform
SRC	Sparse Representation Classification
SVM	Support Vector Machine
TS	Traffic Signs
TSD	Traffic Sign Detection
TSR	Traffic Sign Recognition
URF	University Research Fund

WaDe Wave-based Detector

WHO World Health Organization

YCbCr Y' (luma), Cb (blue chroma), and Cr (red chroma)

CHAPTER ONE

INTRODUCTION

1.0 Overview

This section begins with background discussion of the study before presenting the problem statement. The purpose of the study, research objectives and research questions, and the study's justification is documented. The scope of the study is discussed, and finally, the assumptions and limitations of the study are explained.

1.1 Background of the Study

Road safety is still a big issue globally, but it affects developing nations more than developed ones. In many developing countries, road safety depends on drivers following traffic regulation, while enforcement is done by government through traffic department. Several challenges affect the enforcement of traffic regulation in developing countries including lack of resources and human labor. These challenges have led to road safety research receiving increasing global attention due to the role it plays in protecting human life. According to World Health Organization (WHO), 1.35 million people die annually due to accidents caused on road while 20 to 50 million end up with injuries or disabilities [1]. [1] notes that a significant proportion of the accidents are caused by drivers failing to adhere to traffic regulations, drivers' distraction, and the inability to detect and recognize Traffic Signs (TS). Ensuring driver compliance with traffic signs can help prevent a substantial number of road accidents.

In response to these challenges, a lot of research has been done targeting several areas including design and development of Advanced Driving Assistance Systems (ADAS) that

automate tasks like Traffic Sign Recognition (TSR) and lane departure. TSR has a major impact in the robustness and effectiveness of ADAS.

1.1.1 Traffic Signs Recognition

Every road in the world has traffic signs whose main goal is to pass information to the road users on how to safely navigate. Traffic signs are usually placed along the road to either regulate, warn or guide road users. TSR plays a critical role of ensuring road safety and efficient traffic flow. They are normally universal accepted and may vary depending on the road system one is using. Therefore, their visibility, legibility and placement are critical especially in complex traffic scenes. The Vienna Convention code on traffic signs is an example of traffic system that has been adopted by many countries in the world. In this code, traffic sign can be divided into three; regulatory signs, warning signs and informational signs. Figure 1 has examples of these traffic sign categories.

a) Regulatory sign b) Warning sign c) Informational sign

Figure 1.1: Classification of traffic signs

Regulatory sign instructs the driver on traffic rules that must be obeyed including speed limit while warning signs communicates on the state of the road ahead including potential dangers such as sharp corners or narrow road. Information signs provide directional or important information on the road like bus stage area.

Notably, the Vienna Convention promotes uniformity in the design and appearance of TS in terms of color and shape with the aim of supporting international understanding and road safety. Many countries in the world have adopted the Vienna Conventions code making it possible to localize TS using color and shape.

Traffic sign recognition (TSR) research has been attracting enormous attention and is considered an important element of ADAS and autonomous vehicles [2]. Autonomous vehicle's ability to promptly and accurately detect and recognize a traffic sign is considered as a matter of human life [3]. The process of Traffic Sign Recognition has three stages; localization of TS, detection of TS and classification of TS. Notably, if any of these stages returns a wrong result, then the overall output will be false and may lead to chaotropic incidents including death. TS discriminative features (color and shape) can be localized therefore enabling detection of TS. Nonetheless, localization and detection should be done correctly with a higher speed in a real-world situation. It could be catastrophic to detect a TS after the vehicle has already passed it. Classification places the TS to a specific class. In the Vienna convention, TS can be classified into 43 classes with Class A being warning signs, B regulatory and C informative [2]. Table 1 summaries Vienna Convention traffic signs.

Table 1.1: Vienna Conventions traffic sign classes

Class	Code	Meaning
Danger Warning Signs (A)	A1	Dangerous curve
Danger Warning Signs (A)	A14	Pedestrian crossing ahead
Danger Warning Signs (A)	A20	Wild animals crossing
Regulatory Signs - Priority (B1)	B1	Stop
Regulatory Signs - Priority (B1)	B2	Yield (Give way)
Regulatory Signs - Prohibitory (B2)	B5	No entry
Regulatory Signs - Prohibitory (B2)	B13	Speed limit
Regulatory Signs - Mandatory (B3)	B15	Turn left
Regulatory Signs - Mandatory (B3)	B18	Roundabout
Informative Signs (C)	C1	Direction to city/route
Informative Signs (C)	C5	Parking

Classification is done using an algorithm know as a classifier. Some commonly used classifiers include Support Vector Machine, Random Forest, Decision Tree and Convolution Neural Network. The choice of a classifier depends on the task and resources (time and space) available.

A number of Traffic Sign Recognition (TSR) systems [4] [5] [6] [7] have been developed and tested on various traffic sign datasets. Although there are variations in traffic signs that make them distinctive, TSR systems are affected by a number of factors among them variations in pictogram designs, illumination, rotation, partial road sign occlusion, variety of colors, color deterioration due to exposure to sunlight, deformation, weather conditions, environmental conditions among others [8]. A robust system for TSR should be able to automatically identify and detect traffic signs from an image or video frame, and correctly classify the traffic sign [3].

1.2 Statement of the problem

Traffic Sign Recognition techniques developed over the years have been either colour-based or shape-based. One method utilized in TSR is the application of handcrafted feature extractors. Notably, colour-based handcrafted feature extractors are adversely affected by noise, occlusion, variety of colours, and colour deterioration due to exposure to sunlight, deformation, and illumination among other factors while Shape-based are affected by rotation, scale and occlusion. The other method for TSR (deep learning) performs better than hand-crafted techniques, but are significantly slow and require a lot of computational resources. These factors make TSR a challenging task. There was need to develop TSR system with a robust feature extractor that balance accuracy and computational resources leading to a better outcome.

1.3 General Objective of the study

The study's primary goal was to design, develop and evaluate a robust machine learning model for traffic sign recognition.

1.4 Objective of the study

The study achieved these objectives:

i. To analyze existing traffic sign recognition techniques for their strengths, weaknesses, and performance
ii. To design robust feature descriptor for extraction of discriminative traffic sign features
iii. To develop a machine learning model for traffic sign recognition
iv. To test and validate the performance of the developed model.

1.5 Research Questions

The study answered these research questions:

i. What are the existing traffic sign recognition approaches and techniques?
ii. How could existing traffic sign recognition approaches and techniques be enhanced for extraction of suitable features for improved recognition results?
iii. How should machine learning models be designed to improve trade-off between speed and accuracy in traffic sign recognition?
iv. How can the performance of the model be tested and validated?

1.6 Justification of the Study

Machine learning models are popular for TSR because they can learn by applying large amount of data and adapt to new situations. Traffic Signs are dynamic, and may contain a various objects, colors, and other factors. Despite the fact that traditional approaches can be used to recognize traffic signs, they may struggle to capture variability. Notably, with

the ability of training on vast data set and proper adjustments, machine learning models can be a solution to TSR with different variation and environment. Additionally, these models have the ability of not only recognizing but also classifying new tasks with high accuracy, even when the task contain variations or noise that was missing in the training data. Also, machine learning models can generalize across different Traffic Signs and environments, making them more versatile and adaptable to different applications. Machine learning models provide an alternative to Traffic Sign Recognition that has the potential of increasing accuracy, proficiency, and robustness.

1.7 Significance of the Study

Numerous applications pertaining to mobility, safety, and transportation could be impacted by machine learning model for traffic sign recognition. Accurate and efficient Traffic Sign Recognition can help improve the performance of various systems and applications, including autonomous driving. Machine learning models for TSR can enable autonomous vehicles to move freely and obey various traffic signs as well as scenarios in real-time. By accurately detecting and classifying different objects and structures within a Traffic Sign, autonomous vehicles can be able to ensure safe and efficient driving through correct decision making. Machine learning models for TSR can monitor and analyze traffic flow, congestion, and safety in real-time. Traffic management systems can come up with good decision that will help in optimizing traffic flow. Machine learning models for TSR has the ability of analyzing and interpreting different Traffic Signs and environments to inform urban planning decisions.

1.8 Research Scope

The study was limited to traffic sign, machine learning and image classification. Statistically, the study used the widely applied GTSRB dataset that had been collected from different roads.

1.9 Limitation of the Study

Financial and time constraints limited the study. The study was done in three academic years as per the university requirements. Also, the study required financial support in buying of necessary equipment's for the carrying out different experiments. However, the researcher multitasked on different areas of the study to ensure that it's completed on time. Additionally, the researcher funded the research.

1.10 Assumptions of the study

This study made the following assumptions:

i) The GTSRB dataset that study applied had an honest reflection of real world.

ii) The Application Programming Interface (API) and libraries that were used for testing produced real results.

1.11 Study Contribution

The idea and creation of the Local Directional Histogram of Oriented Gradient (LD-HOG) feature extractor constitute this work's main theoretical contribution. The unique approach of computing gradient responses along the 45° and 135° diagonals in addition to the conventional horizontal and vertical axes is the theoretical novelty. LD-HOG creates a more thorough and discriminative feature representation by combining these multi-

directional gradients using a maximum pooling technique into a single orientation binning and normalization procedure. By overcoming the shortcomings of both HOG and LDP this technique creates a new theoretical framework for building reliable handcrafted features in computer vision.

CHAPTER TWO

LITERATURE REVIEW

2.1 Overview

The chapter covered the following section: Image classification, artificial neural network, convolutional neural network, traffic sign classification, traffic sign analysis, traffic sign recognition models and frameworks, theoretical framework, knowledge gap and conceptual framework.

2.2 Image classification

Computer vision, specifically Image classification plays a major role in technology especially artificial intelligence. [9] notes that computer vision main task is to ensure that models created can examine, understand, and analyze data and information straight from the users. [9] further note that computer vision application is widely spread to different sectors including robotics, driverless vehicles, health and medicine and many more. Image classification is one of the tasks performed in computer vision [9]. According to [10], image classification can be defined as a branch of computer vision that can provides output that are usually categories for input that can be an image or video frames based on the image contained. [10], further notes that the main task of image classification is to assist machines tasked with creating models to understand and analyze the contents of images and videos. Also, despite image classification being one of the main tasks in computer vision, its applications are important and they include video and image retrieval, video indexing, and surveillance.

Traffic Sign is an image that can be classified. [11], [12] and [13] notes that there are different models created for identifying as well as classifying traffic signs. These models apply different techniques in classifying traffic signs as discussed in section 2.3 and 2.4.

2.3 Artificial Neural Networks

A human brain has a neural network made up of connected and interacting nerve cells, an Artificial Neural Network (ANN) mimic this network. The brain network works by transmitting electrical signals using an axon terminal from the output called synapses to the input referred to as dendrites of a neuron [14]. [15], adds that each neuron must be activated in order to receive the electrical signal through an activation threshold. The total signal entering the input layer must be greater than the activation threshold in order for the neuron to fire, activate, and transmit. Notably, ANN mimics the human brain which is one of the complex networks with over 10^{10} neurons with each single neuron having a connection to approximately 10^4 similar ones according to [15]. ANN is a simulation of the human brain though on a lower scale. [14], notes that in order to create a full ANN, the activation function that would be compared to a weight form different neuron combination is modeled with a numeric value. Additionally, a non-linear activation function is associated with each neuron. [15] goes on to explain that the network is used as a non-linear function since certain neurons are classified as input or output neurons. One advantage of ANN is that it can contain an arbitrary number of input and output neurons, making it a very flexible operational structure that can be used for a wide range of tasks in a variety of ways, according to [14] and [15].

2.3.1 Deep Feed-Forward Network

The deep feed-forward network has a set of modeling techniques that have shown great success in the present computer vision problems of pattern recognition, classification, regression, etc. [16] observes that information only moves in one direction across this network, suggesting that it does not have a feedback mechanism. [17] goes on to say that these neurons are layered so that each input and output layer is distinct, and there may be a few hidden levels as well. This provides a systematic approach to deciding when to activate the output layer given the input and the weights and biases of each intermediate layer.

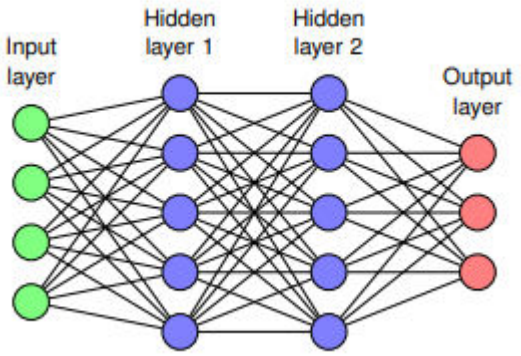


Figure 2.1: Feed forward neural network

A feed-forward neural network with completely linked layers is shown in Figure 2.5. According to [17], matrix-vector multiplication in conjunction with element-wise activation functions can be used to quantitatively depict the process of feeding and transmitting information across such a network. For a network-like figure 2.5, [16] and [17] provide a condensed illustration of how to obtain the output vector a_1 from the first

hidden layer. If x is the input structured in an N-dimensional vector and σ_1 is the activation function, then equation 2.6 gives the value of a_1. The network weight matrix is W_1, w_{ij} links input neuron j with hidden neuron i, and b_1 is a vector containing the biases of neurons in the first hidden layer.

$$a_1 = \sigma_1(W_1 x + b_1), \tag{2.1}$$

It serves as the second hidden layer's input. [17], use equation 2.7 to generalize the equation for layer n output.

$$a_n = \sigma_n(W_n a_{n-1} + b_n). \tag{2.2}$$

A deep feed-forward neural network is essentially a multi-layer feed-forward network. Moreover, a hierarchical representation of the input features is produced by the information flow. This allows for greater non-linear mapping because deep feed-forward has a high modeling capacity.

2.3.1.1 Activation Functions

In order for ANNs to perform plotting for non-linear input data, activation functions play an important role. The activation functions are always applied to each element of the neurons that form the hidden layer [18]. According to, [19], [20] and [21], the commonly applied activation functions were Sigmoid, Hyperbolic Tangent, Rectified linear Unit (ReLU) and SoftMax for the last layer.

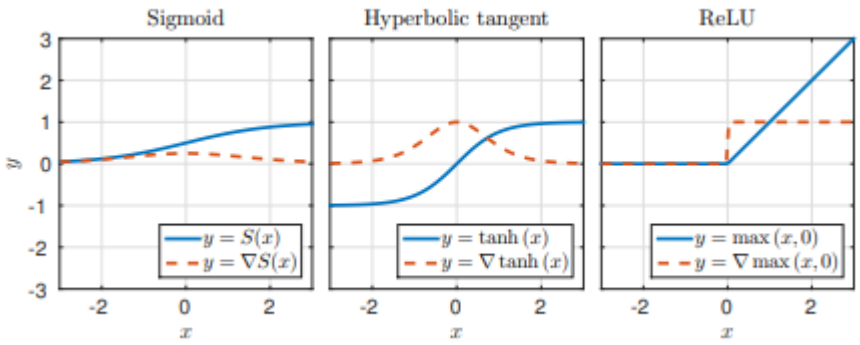


Figure 2.2: Common activation functions

According to [19] the sigmoid activation function can be represented using the following formula

$$S(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

[19] observes that this function has a linear relationship near the origin and accepts values between 0 and 1. Additionally, this function has the "squashing" property, which causes big input values that are either positive or negative to approach 1 or 0, respectively. [20] states that the following equation can be used to represent hyperbolic tangent.

$$tanh\ tanh\ (x) = \frac{sin\ h(x)}{cos\ h(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.4}$$

[19], and [20] notes that this activation function is similar to sigmoid but $tanh\ tanh\ (x)$ can take values from -1 to 1. The ReLU is also a common activation function applied to deep neural networks represented by the following equation:

$$f(x) = ma\,x(x, 0) \tag{2.5}$$

[21] notes that, unlike the sigmoid activation function, ReLU has the benefit of not saturating for large inputs. Furthermore, since sigmoid functions accept values between 0 and 1, they are less likely to be "on" or "off" than ReLU, which accepts values between 0 and ∞. The SoftMax function is a normalizing function that is commonly used as an activation function for the final layer of a neural network, as [22] notes. An example of an equation to convert an input vector x with values x_i into the corresponding output element y_i is given in [22]:

$$fy_i = \frac{e^{x_i}}{\sum_{J=1}^{N} e^{x_j}} \tag{2.6}$$

[22], further notes that the elements of the output vector must sum to 1. Notably, SoftMax should be applied when the output of the ANN should be different case probabilities including classification problem. Table 2.1 summaries activation function

Table 2.1 Summary of activation functions

Activation Function	Strength	Weakness	Big O notation
Sigmoid	Smooth gradient, suitable for backpropagation algorithm	Susceptible to vanishing gradient problem	O (1)
Hyperbolic Tangent	Zero centered output, which can help in training convergence	Susceptible to vanishing gradient problem	O (1)
Rectified Linear Unit	Avoids vanishing gradient problem	Not Zero centered, can cause dying ReLu problem	O (1)
Soft-max	Outputs probabilities, suitable for classification	Sensitive to outliers and high dimensionality	O (n)

From table 2.1, soft-max had the worst, worst case time complexity of O (n) while the other three had a constant time (O (1)). Notably, sigmoid was best suitable for binary problem while soft-max was for multi-class classification. Hyperbolic Tangent was mainly applied to recurrent neural networks and ReLu performed well when applied to hidden layers of deep neural network. Additionally, sigmoid worked well with back propagation algorithm while ReLu avoided vanishing gradient problem. Hyperbolic was Zero centered.

Sigmoid and hyperbolic were susceptible to vanishing gradient problem, soft-max was sensitive to outliers while ReLu could have dying ReLu problem.

From reviewed literature, it was clear that feed forward network with activation could classify traffic sign, however the accuracy and recall rate were low. To solve these a mechanism to send back information from the output to the input while monitoring the accuracy was important. These mechanisms include back propagation, cost function and gradient descent.

2.3.1.2 Back propagation Algorithm

Supervised learning is one common technique that is used to make the mapping of input to output of an ANN useful. [23], notes that in supervised learning, one must have predetermined desired outputs for a specific input. [23], further applies an example of images for rock scissors and paper to explain supervised learning. In his discussion, [23] notes that the desired output y along with the actual output $\hat{y}$ are passed to a differentiable cost function C which is usually minimized by adjusting the weight and biases of the network. If Θ is the set of all parameters in the network, [24], notes that the objective when training using supervised learning is given be the following equation:

$$\theta min \frac{1}{N} \sum_{i=1}^{N} \quad C(Y_i, \hat{y}_i | \theta), \tag{2.7}$$

Where $\hat{y}_1 \ldots \hat{y}_N$ and $y_1 \ldots y_N$ represents the output and labels of the network for a training data $x_1 \ldots x_N$. [24] adds that until the network reaches a satisfactory level of accuracy when exposed to validation or test data, the process of transferring data through the network, evaluating the cost, and readjusting weights and bias is carried out frequently. Notably, back-propagation is a technique used in feed-forward networks to modify various

parameters to attain the intended outcomes [23]. Gradient descent of the cost function with respect to the parameters is applied using a chain rule in backpropagation.

2.3.1.3 Stochastic Gradient Descent

While a single sample of input data, desired output, and actual output suffices to calculate the gradients for all network parameters, it is customary to incorporate many samples and average the resultant gradients [25]. The term "mini-batch" typically denotes this grouping of samples. Mini-batches provide numerous applications in network training. The average of gradients is the optimal method to estimate the gradients of the minimization problem, derived from the complete training set. Secondly, leveraging the extensive parallelism offered by contemporary CPUs and GPUs generally results in more efficient computation of gradients for several inputs simultaneously. The technique of randomly selecting samples from the training set, calculating the gradients, and subsequently adjusting the parameters in line with equation 2.13 is referred to as stochastic gradient descent, or SGD, as noted in [25].

$$\ominus \leftarrow \ominus \; - \frac{\propto}{m} \sum_{i=1}^{m} \quad \frac{\partial C\left(y_{i,}, \hat{y}\, \ominus\right)}{\partial\, \ominus}, \tag{2.8}$$

Where α represents the rate at which the model learns and the size of the mini batches is represented by m.

[25] points out that the network parameters are updated using epochs, which are the number of times all training samples have been used. Epochs are frequently used to measure the quantity of training that has been done. If N training samples are available, one epoch is completed after using N/m mini-batches for training. If the input data is changed in any

manner before being delivered to the network, it is impossible to specify the measure of one epoch in the original meaning because this could lead to an excessive number of training samples being used [25]. Under such circumstances, the number of mini-batches used is directly proportional to the training.

2.3.1.4 Vanishing Gradient problem

To train deep neural networks using back propagation, gradients need to be propagated over several layers. The gradient pertaining to a specific layer is multiplied by the activation function gradients for every layer that has previously been passed through the chain rule. Since the gradients of the activation functions were typically restricted to the range (−1, 1), the resulting gradient tends to zero since it is a product of values with magnitudes lower than one [25]. In reality, learning occurs more slowly and inefficiently when layers close to the network's input layer learn more slowly than layers close to its output. The vanishing gradient problem is the name given to this issue. The use of ReLUs in deep neural networks is one method that has greatly aided in solving the vanishing gradient problem [26].

2.3.2 Convolutional Neural Network

Convolutional neural networks (CNNs), a type of feed-forward neural network, possess a distinct architecture that enables them to analyze spatially organized input, such as 2D or 3D images represented as tensors. The architecture draws inspiration from the mechanisms of biological visual perception [27]. Similar to other artificial neural networks, these networks consist of neurons with adjustable weights and biases. Upon receiving input, each neuron does a dot product prior to possibly executing an activation function. Deep learning refers to CNN research due to the architecture's characteristic of being "deep," consisting

of multiple layers. The network establishes a mapping function that correlates the pixels of an image with the desired output [27]. An RGB image, with three channels that denote the intensity values of red, green, and blue, is intended to serve as the input for a general convolutional neural network (CNN). Subsequent layers of the CNN may comprise additional channels, referred to as feature maps.

Until the desired output size is attained, the number of feature maps usually rises across a CNN's layers, while their spatial dimension falls. This structure's general idea is that the input image's representation gets increasingly abstract as the layers increase in number [28]. A picture's later layers reveal less about the "where" and more on the "what" and "how" of items and things. A feature vector at a CNN's single layer is comparable to a feature map and is composed of the components of all feature maps at a certain geographic region. The CNN structure is illustrated in Figure 2.4 [28].

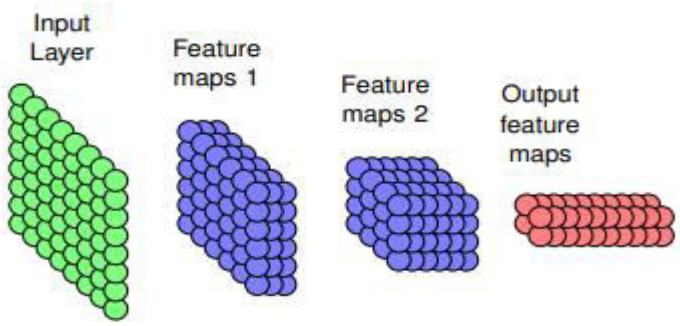


Figure 2.3: Structure of feature maps in a CNN.

Figure 2.7 depicts an 8 by 8 one channel that is grayscale picture as the CNN model input as well as the feature maps that will act on the image.

2.3.3 Components of Convolutional Neural Network

In addition to the fully connected layer described in section 2.2, many types of layers and connections can be employed to construct deep convolutional networks. Certain elements are frequently employed to emulate fully connected layers, alter spatial dimensions, minimize the size of intermediate layers, and additional functions [28]. This section elucidates the essential components of the CNN framework.

2.3.3.1 Convolution Kernel

A convolution kernel defines the discrete 2D convolution action shown in figure 2.7. For size$k \times k$, K by k. The convolution kernel is applied to every pixel in a $N \times M$ input picture (tensor) X, multiplying and adding the values of the surrounding pixels. The output is $a(N-k+1) \times (M-k-1)$ image Y. Equation 2.14 below is used to determine the output at pixel location i, j.

$$Y_{i,j} = \sum_{i'=1}^{k} \sum_{j'=1}^{k} k_{i',j'} X_i - \frac{k+1}{2} + i', j - \frac{k+1}{2} + j' \tag{2.9}$$

Further explained as

$$Y = k * X \tag{2.10}$$

A 3 x 3 kernel convolution operation and the relationship between several input activations and a single output activation are shown in Figure 2.8 [29].

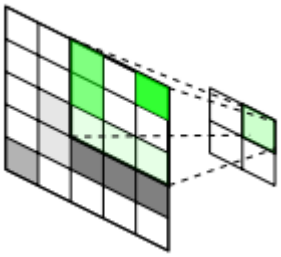


Figure 2.4: An example of the convolution process with a 3 × 3 kernel

As with the fully connected layer, the later layer applies a bias bi across the entire feature map by summing the output of all convolutions related to feature map i [29] on kernel. The discrete 2D convolution action seen in figure 2.7 is defined by a convolution. The convolution operation between two layers in a CNN is defined by a kernel k_{ij} for each pair of feature mappings in the two layers. The feature map Yi in the next layer is found using the following equation, assuming that $X_j$is a feature map in a layer with $N \geq j$ feature maps:

$$Y_i = \sigma_i\left(\sum_{J=1}^{N} \quad k_{ij} * X_j + b_i 1\right), \tag{2.11}$$

$$Y_i = \sigma_i\left(\sum_{J=1}^{N} \quad k_{ij} * X_j + b_i 1\right) \tag{2.12}$$

where σi is an element-wise activation function and i is an all-ones matrix. The biases b_i and the components of all kernels k_{ij} are learnable parameters that are changed throughout the network training procedure [29].

[29] states that the size of the feature maps is also reduced by the convolution method. Nonetheless, numerous techniques exist to alleviate or circumvent this effect. A prevalent

method for preserving spatial dimensions is zero-padding, which entails surrounding the feature maps with a border of $(k1)/2$ zeros prior to the convolution process, effectively counteracting spatial reduction. Nevertheless, a stride can be incorporated into the convolution to further reduce the spatial dimensions of activations in the intermediate layer. The stride advances the convolution kernel by s steps between each "multiply and sum" operation, hence diminishing the size of the feature maps by a factor of s.

Convolutions are generally performed without stride, indicating that $s = 1$. [30] indicates that the essential concept of the receptive field is established when a CNN is augmented with a convolutional layer. The receptive field of a CNN measures the extent of information from the input image that the feature vectors of a particular layer may utilize. If the initial component of a CNN is a convolutional layer with kernel width k_i, then its receptive field measures $ki \times ki$. Every element in the second layer obtained information from the $ki \times k_i$ nearest pixels in the input image. Based on this pattern, n convolutional layers with kernel sizes

$$\left(1 + \sum_{i=1}^{n} (k_i - 1)\right) \times \left(1 + \sum_{i=1}^{n} (k_i - 1)\right) \tag{2.13}$$

2.3.3.2 Up-convolution or backwards convolution

Up-convolution, a process akin to convolution but inverted, can be applied to up sampling. As illustrated in figure 2.9, it is achieved by reversing the convolution approach, as per [30]. Therefore, if up sampling by f is necessary, it can be represented as a convolution with a fractional input stride of $(1)/f$. This kind of layer can connect many output activations to a single input activation. Learnable filter settings in this up-convolution layer

may correspond to object reconstruction bases. Consequently, [30] notes that repeatedly down-sampling and then up-sampling to produce dense predictions can be used to build an end-to-end learning mechanism, and that dense pixel-level predictions have been successfully made using this technique.

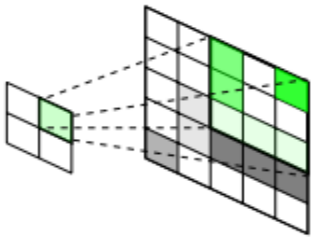


Figure 2.5: An example of the up-convolution process

2.3.3.3 Pooling layers

In order to decrease the spatial dimensionality of feature maps as they traverse the network, non-learnable pooling layers are used. Similar to convolution layers, they are paired with a stride s and a kernel of size $sk_i \times k_i$. The max-pooling layer and the average-pooling layer are the two kinds of pooling layers that are frequently employed, per [29, 30]. The max-pooling layer, depicted in figure 2.10, discards the data from the non-max neurons by performing $amax(\quad)$ operations on the feature map elements at each kernel point.

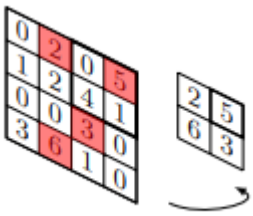


Figure 2.6: Maximum pooling process

The output activation is determined by the largest value recorded among the input activations within a specified window. However, as noted in [29], the average-pooling layer calculates an average at each kernel location by performing a standard convolution with all kernel values assigned to $\frac{1}{k2}$. [29, 30] indicates that to achieve dimensional reduction of components without necessitating zero-padding, the stride of the pooling layers is typically set at s = k, as illustrated in figure 2.11.

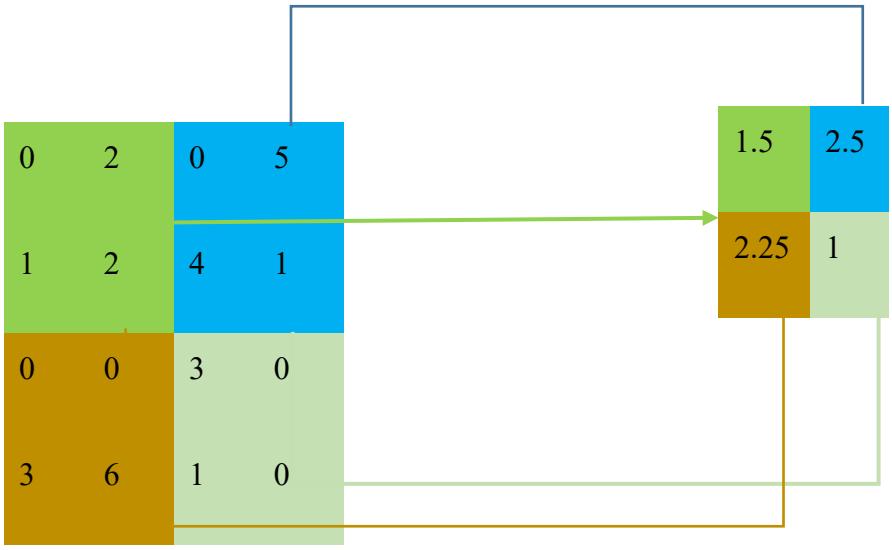


Figure 2.7: Average pooling operation

Notably, pooling layers provide a practical way to decrease the spatial dimensions of feature maps while simultaneously expanding the receptive field in a CNN. The effect of a pooling layer with stride s is to increase the receptive field of a factor. Nevertheless, [30] notes that stride can be added to a convolutional layer to get this pooling layer effect.

2.3.3.4 Un-pooling

A reversing process called un-pooling aims to restore the initial size of activations that pooling has destroyed. Figure 2.9 [30] illustrates a form of un-pooling specifically designed to reverse the max-pooling process, which preserves the maximum activations selected during pooling and employs them to reconstruct the original activations by repositioning the recorded values to their initial places. The index of the highest value selected during down sampling is always stored in the maximum function used for pooling. As a result, the maximum value is returned and put in its original location after un pooling, while zero

is used to replace the other values. This method works well for parameter-free up sampling and object structure recovery.

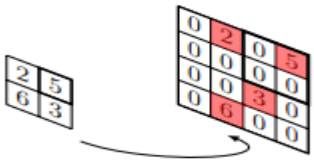


Figure 2.8: Max-un-pooling process

The recorded maximum activations from a corresponding max-pooling layer are utilized to extract the maximum input activations. All other values within a window are assigned a value of zero subsequent to the recovery of the maximum value.

2.3.3.5 1 × 1 Convolutions

Remarkably, the 1×1 convolutional kernels and the $k \times k$ kernels are utilized in a same fashion. Philosophically, $k \times k$ convolutions are typically regarded as edge or feature detectors, whereas 1×1 kernels solely serve to amalgamate the activations of individual feature vectors [31]. This can be interpreted as implementing the influence of a fully connected layer on each feature vector. The convolution operation's independence from the spatial dimensions of its input makes the transformation of fully connected layers into 1 × 1 convolutions an effective method for generalizing the network across varying input sizes [30–31].

2.3.3.6 Batch Normalization

Because the distribution of inputs for each intermediate layer varies during training, deep neural network training can be challenging in practice. Changes in the characteristics of

the preceding layers are the reason for this shift [32]. Internal covariate shift is the term used to describe this issue. The vanishing gradient problem is also caused by the activations' tendency to migrate into the saturated regimes when the input distribution changes. As the network depth increases, this effect gets worse. To address this, a normalization technique is designed that normalizes the inputs to the activation layers over the mini-batch [32]. A brief overview and detailed explanation of the batch normalizing transform algorithm are provided below.

Think of a mini-batch with n inputs, where $B = x1, x2 \dots, xn$. The batch normalization's linear transformation can be shown by $\{y1,\ 2 \dots, yn\}$ [32-33] if the normalized values are $x\hat{}1, x\hat{}2 \dots, x\hat{}n$. Then, the batch normalizing transform is given by:

$$BN_{\gamma,\beta} : x_1, x_2 \dots, x_n \rightarrow y_1, y_2 \dots, y_n \tag{2.14}$$

where the learnable parameters γ and β indicate the transformation's scaling and shifting [33]. Thus, the variance and mini-batch mean, which are provided by:

$$\mu B \leftarrow \frac{1}{n}\sum_{i=1}^{n} x_i \quad \sigma_B^2 \leftarrow \frac{1}{n}\sum_{i=1}^{n} (xi - \mu B)^2 \tag{2.15}$$

Thus, the normalization looks like

$$x\hat{}i \leftarrow \frac{xi - \mu B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{2.16}$$

Thus, the outcome is presented by:

$$yi \leftarrow \gamma x\hat{}i + \beta. \tag{2.17}$$

2.3.3.7 Inception module

An architectural enhancement that attempts to deepen the network while maintaining efficiency is part of the inception module that was introduced in. Convolutional building blocks are used, and a deep architecture is constructed by closely examining the correlation statistics of the preceding layer [34].

i). A layer of 1x1 convolutions designed to cover the whole spatial range of the output from the layer before it;

ii). A 3 x 3 convolutional layer designed to group small sets of units with strong spatial correlation; and

iii). A layer of 5x5 convolutions is used as an initialization block or module to cluster bigger groups of units with high spatial correlation.

Due to the differing levels of correlation in feature mappings during the initial phases of the network, all three varieties of convolutional layers are typically integrated into the later inception module [34]. The 1×1 and 3×3 convolution layers are superior design choices as the network's depth escalates. This architecture offers the advantage of utilizing 1×1 convolution layers prior to the more computationally intensive 3×3 or 5×5 convolution layers, hence greatly reducing the feature map dimensions for the larger spatial convolution layers. This approach circumvents the computational bottleneck associated with the use of Inception modules. The inception module, a crucial element of several inception topologies, is illustrated in Figure 2.13.

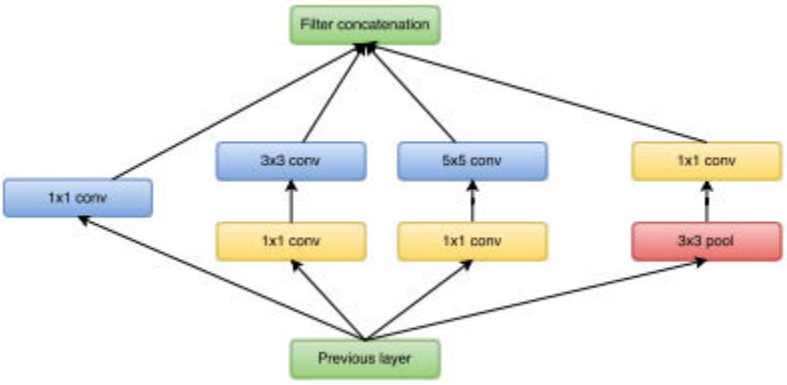


Figure 2.9: Inception module.

2.3.3.8 Residual learning

Researchers were inspired to attempt building even deeper architectures by the success of deep convolution networks. However, the vanishing gradient problem is responsible for the learning's tendency to saturate with depth [35]. The deterioration issue seen in very deep architectures is intended to be resolved by the architecture presented by deep residual learning. In order for the few stacked layers in between to learn a residual mapping instead of the original one, skip connections between several convolutional layers are meant to be inserted. More precisely, if a few stacked layers were to fit a desired mapping, say $H(x)$, the stacked layers would try to fit another mapping $F(x) \coloneqq Hx - x$ through the revised architecture. Consequently, $F(x) + x$ is the new formulation of the original mapping. The architecture is based on the notion that optimizing a residual mapping is simpler than optimizing the original, unreferenced mapping [35]. The fundamental component of a residual learning architecture is depicted in Figure 2.14. Essentially layered on top of each other, this building block of residual layers with skip connections enables the creation of extremely deep structures.

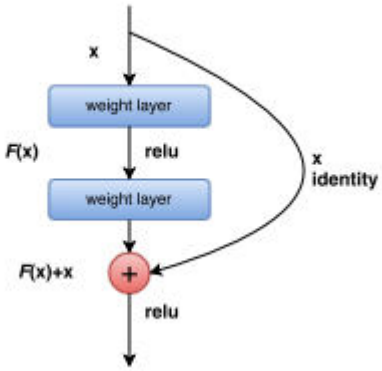


Figure 2.10: Residual learning scheme.

2.3.3.9 Tiling

Tiling is a non-learnable upsampling method that entails reshaping and combining feature maps. The technique pertains to a singular parameter, t, and necessitates that t^2 be a divisor of the quantity of feature mappings in the layer input [36]. Given m feature maps of size N × M, reshaping is applied to all feature vectors, rearranging them into m/t^2 matrices of size t × t. After concatenating the matrices from each feature vector, m/t^2 feature maps of size tN × tM are generated. The tiling technique offers a notably advantageous interpretation when applied immediately following a 1 × 1 convolution layer. The convolution kernel weights are not uniformly distributed across feature maps, resulting in an interpretation of applying t^2 independent fully linked layers to each spatial position, with the output organized in a t × t grid.

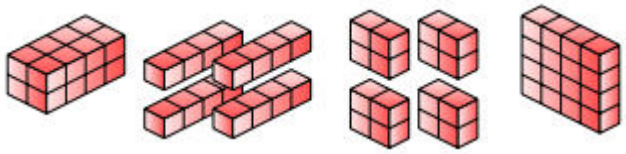


Figure 2.11: Tiling operation

A tensor that extends in the feature dimension is converted to tile across in the spatial dimension. Four 2 × 2 feature maps are put through a size 2 tiling operation in the example for figure 2.15.

2.3.3.10 Dilated Convolutions

Dilated convolutions are altered convolutions that enable an expansion of a network's receptive field size while preserving the spatial dimensions of the intermediate feature maps. The formulations for dilated convolution and general discrete convolution are delineated as follows [37], wherein dilated convolution filters are constructed by augmenting the general convolution operator $*$. Let $\Omega r = -r, r2 \cap Z2$ and let $F\backslash: Z^2 \rightarrow R$ be a discrete function. Additionally, let k be a discrete filter with the size $(2r + 1)^2$ and defined as k: $\Omega r \rightarrow R$. With this explanation, the discrete convolution operator $*$ and the more generalized dilated convolution operator $*_l$ can be given by:

$$(F * k)(p) = \sum_{s+t=p} F(s)k(t) \tag{2.18}$$

$$(F *_l k)(p) = \sum_{s+lt=p} F(s)k(t) \tag{2.19}$$

where the l-dilated convolution process was denoted by $*_l$. The expression changes to the typical 1-dilated discrete convolution operation when i = 1 [37]. As seen in figure 2.12, the dilation filters allow for an exponential rise in the receptive field, which is indicated by:

$$F_{i+1} = F_{i*_{2^i}} k_i \tag{2.20}$$

In image segmentation, when multi-scale contextual information must aggregate without erasing spatial information, this trick is particularly helpful. The dilated convolutions, as shown in figure 2.12, are a practical alternative to the concept of down-sampling feature maps using strides in convolution layers or consecutive pooling [37].

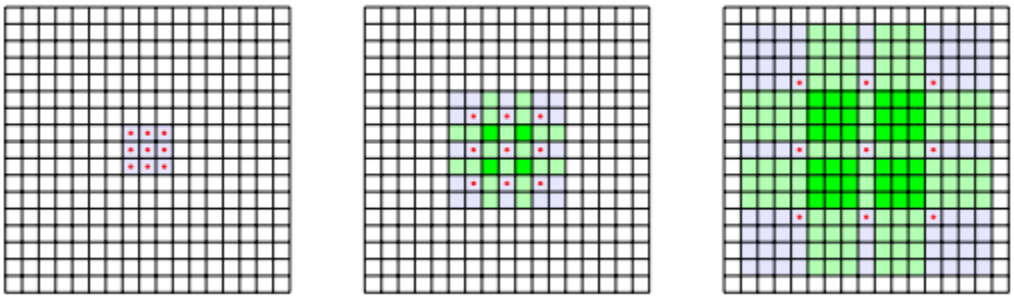


Figure 2.12: 3 × 3 kernel increase dilation.

The first filter has a 3x3 receptive field with 1-dilated convolutions; the second has a 7x7 receptive field with 2-dilated convolutions for each element; and the third has a 15 x 15 effective receptive field with 4-dilated convolutions. Whereas the blue area displays the receptive field, the green areas display areas where the elements overlap. The receptive field can extend exponentially because to the architecture.

2.3.4 Classification, Detection and Localization

Deep learning techniques for object categorization, localization, detection, and segmentation were made popular by the ImageNet challenges. Given a training set of more than 5 million annotated photos, the classification task in this multi-task challenge is to

assign images to one of 1000 different classes [38] and [39]. In 2012, the CNN-based methodology and architecture outlined in [38] produced state-of-the-art classification results, revolutionizing computer vision techniques. The remaining tasks entail localization and detection, which call for providing bounding box coordinates in the picture pixel grid that indicate the location of the classified object. Therefore, CNNs are initially demonstrated to be promising for applications involving categorization. Soon after, though, they undergo modifications to enable identification and localization. In this part, we provide a brief overview of the body of theory that clarifies some of the ideas involved in using CNNs for localization and detection [39]. These ideas form the core of the thesis goal and will serve as the foundation for the methodology.

2.3.4.1 Detection using Classification

Many methods rely on independently identifying spatial portions of the image as a first step between classification and localization, offering a "detection through classification" approach [39]. To minimize the number of regions that are sent for classification, the detector typically labels the spatial regions as "Object" or "Background." A greater level of categorization of the geographical regions can be obtained directly by combining the classifier and detector, though, assuming the computing cost of the classification is not an issue [39]. By allocating the number of feature mappings in the final layer to the number of classes that need to be differentiated, a CNN can be used to accomplish a detection through classification strategy. After that, each feature map can be linked to a certain class.

The feature vectors generated by employing a SoftMax function as the activation function correspond to the probability distribution of the classes. Each prediction generated by such a CNN would possess a receptive field centered at the identical point and a matching spatial

position within the image. The receptive field of the predictions is delineated by many pixels due to the stride of the layers in a CNN. The effective stride of the network is determined by multiplying the strides of its tiers.

The following layers of the network typically consist of a sequence of 1 x 1 convolutions to facilitate separate predictions for each feature vector from the preceding layer. The initial layers of the CNN are designated as feature extractors, and the subsequent layers are termed classifiers. Viewing the CNN pipeline as two distinct networks is highly beneficial when addressing transfer learning. Transfer learning involves pre-training a network architecture on extensive, generalized data sets, such as the ImageNet data set, and subsequently replacing the network's final layers to train on a distinct, usually smaller data set. This technique relies on the premise that a pre-trained feature extractor can derive feature vectors from an input image that possess a high degree of conceptual information [86]. The approach regularizes the network and is advantageous when the dataset of interest is limited.

2.3.4.2 Bounding Boxes

The detection technique outlined previously indicates the object's location inside the image, but it does not provide information regarding its size. A prevalent technique for providing more information regarding the geometry of an object is bounding box regression [88]. The concept involves connecting a smaller network to the output of the feature extractor, which operates concurrently with the classifier, as depicted in figure 2.14.

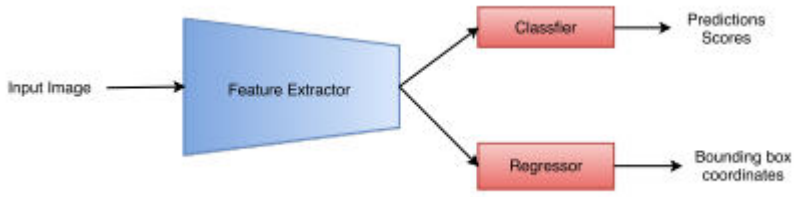


Figure 2.13: Network architecture used to carry out classification-based detection

The configurations of the bounding box regressor and the classifier are analogous. Nonetheless, as a bounding box necessitates a pair of coordinates relative to the image pixels, the quantity of output feature maps is established at 4 [40]. Consequently, each element in the output feature vectors can be linked to one of the bounding box coordinates. The classifier's output and the bounding box regressor can be integrated to ascertain the class and position of items within the image. Due to the one-to-one correlation between the geographical predictions of the classifier and the output coordinates of the regressor, the results for locations classed as "No class" can be disregarded.

The final outcome is a compilation of spatial object detections; each associated with a bounding box. The detector may identify adjacent areas of the same class if the item is sufficiently large, resulting in many bounding boxes corresponding to the same object.

2.3.5 Convolutional Neural Network Models

There are many CNNs Models developed for various domains. This section discusses some of these models highlighting their strengths and weakness.

2.3.5.1 AlexNet

AlexNet, a deep convolutional neural network, was developed in 2012 and took first place in the ImageNet LSVRC competition. In addition to proving CNN's efficacy in complex

models, AlexNet is more significant than earlier convolutional networks because it completes training using GPU implementations. Compared to a CPU, a GPU with thousands of parallel computing cores has a better probability of delivering results in a reasonable length of time. For the first time, AlexNet's activation function is Rectified Linear Units (ReLU) rather than the more traditional Sigmoid and Tanh (hyperbolic tangent).

AlexNet, a deep convolutional neural network, was developed in 2012 and took first place in the ImageNet LSVRC competition. In addition to proving CNN's efficacy in complex models, AlexNet is more significant than earlier convolutional networks because it completes training using GPU implementations. Compared to a CPU, a GPU with thousands of parallel computing cores has a better probability of delivering results in a reasonable length of time. For the first time, AlexNet's activation function is Rectified Linear Units (ReLU) rather than the more traditional Sigmoid and Tanh (hyperbolic tangent).

The response-normalized activity b_x^i, y is determined by formula 2.26 where a^i $, y$ is the activity of a neuron, which is calculated by applying kernel i at location (x, y) and applying the ReLU nonlinearity [41].

$$b_{x,y}^{i} = a_{x,y}^{i} / \left(k + a \sum_{j=\left(0.i-\frac{n}{2}\right)}^{\left(N-1,1+\frac{n}{2}\right)} \left(a_{x,y}^{j}\right)^{2} \right) \text{^} \beta \tag{2.21}$$

Additionally, dropout and data augmentation are used by AlexNet to avoid over-fitting during training. In other words, when the amount of training data is limited, AlexNet

rapidly enhances the training data by applying several performatives to the original dataset, and dropout simultaneously suppresses the forward and backward propagation of the neural network by setting the neurons to 0 using a specified probability, while keeping the number of neurons in the input and output layers constant.

$$ReLU(x) = \{x \quad if\ x > 0\ \ 0 \quad if\ x \leq 0 \tag{2.22}$$

However, AlexNet has some weakness. Notably, AlexNet is a relatively shallow CNN architecture by modern standards, with only five convolutional layers and two fully connected layers. Some more recent models have significantly deeper architectures, which can capture more complex features and achieve even better performance. Additionally, the model has a large number of parameters (60 million), which can make it difficult to train without access to powerful computing resources. Also, the pooling layers in AlexNet can result in a loss of spatial information, which can be important for certain computer vision tasks, such as object localization and segmentation. Lastly, while AlexNet achieved state-of-the-art performance on the ILSVRC dataset, it may not generalize as well to other datasets or tasks, and newer models may achieve better performance on certain benchmarks.

2.3.5.2 GoogleNet

Convolutional neural networks have been used to create ImageNet champions since AlexNet's inception in 2012, and the layers are becoming increasingly complex. CNN thus emerged as the primary algorithm in the domain of picture classification and recognition. GoogLeNet was victorious in the 2014 ImageNet Challenge (ILSVRC14). Despite having just 22 layers, GoogLeNet performs better than AlexNet in a number of areas. With far

fewer parameters than AlexNet, GoogLeNet has the advantage of ensuring superior performance even with constrained memory or processing power. The Inception network topology was suggested by the GoogLeNet team as the explanation for its exceptional performance with fewer parameters.

The Inception network topology was proposed by the GoogLeNet team due to its ability to yield substantial results with a reduced number of parameters. Inception replicates the recurrent neuronal buildup observed in the human brain. The Inception network is utilized to cluster sparse matrices into denser sub-matrices. This approach significantly improves the model's computational efficiency, resulting in a sparse and high-performance network. The Inception network architecture employs the Network in Network approach to enhance the efficacy of the 1×1 convolution kernel for dimensionality reduction and constraining network size.

GoogLeNet was created through four iterations, from version 1 to version 4. By integrating 1x1, 3x3, and 5x5 convolutions along with 3x3 pooling operations, Inception v1 enhances the network's width and improves the model's flexibility. Inception v2 incorporates the BN layer from Inception v1 and mitigates the internal covariate shift to standardize each layer's output to a N(0,1) Gaussian distribution. It also references how VGG reduces parameters and speeds up processing by using two 3×3 convolution kernels in the inception module rather than five×5. Inception v3 later incorporated the Factorization Machine (FM), which broke down 7x7 into a 7x7 and a 7x1 convolution and 3x3 into a 1x3 and a 3x1 convolution. The goal is to boost the network's nonlinearity and speed up the neural network training computation. The Inception block's grid size is unified in Inception v4. Additionally,

ResNet's architecture can greatly aid in speeding up network training and performance enhancement.

Nonetheless, GoogleNet has some weaknesses. Firstly, GoogleNet is a very complex architecture with 22 layers, which can make it difficult to train and optimize. Its complexity can also make it harder to interpret the learned features and understand how the model is making predictions. Secondly, due to its large number of layers and complex architecture, GoogleNet requires a significant number of computational resources to train and evaluate. This can limit its practicality in some settings, especially those with limited computing power. Thirdly, GoogleNet's depth can make it more susceptible to over-fitting on small datasets, which can lead to poor generalization performance on new data. Fourthly, the use of pooling layers in GoogleNet can result in a loss of spatial information, which can be important for certain computer vision tasks, such as object localization and segmentation. Additionally, like other deep learning models, GoogleNet is sensitive to its hyper-parameters, including learning rate, weight decay, and dropout probability. Tuning these hyper-parameters can be time-consuming and require significant computational resources. Lastly, while GoogleNet achieved state-of-the-art performance on the ILSVRC dataset, it may not be as effective on other tasks or datasets, and newer models may achieve better performance on certain benchmarks [42].

2.3.5.3 VGGNet

Based on AlexNet, VGG is a deeper convolutional neural network proposed by Oxford's Visual Geometry Group. The main contribution is to demonstrate that the network's performance will increase with its size. In order to deepen the network, VGG primarily uses the technique of adding more convolution layers. Not only does the network's learning

capacity grow with the number of network layers, but it also strengthens its categorization capabilities. The recognition has significantly improved when the depth is extended to 16 layers, or VGG-16, and to 19 layers, or VGG-19 [43]. One improvement in VGG-16 over AlexNet is the substitution of three consecutive 3x3 convolution kernels for the bigger convolution kernels in AlexNet. This results in deeper layers and fewer parameters than AlexNet, although it simply requires fewer iterations. Similar to GoogleNet and AlexNet, VGGNet has some weakness. Notably, VGGNet is a computationally expensive architecture due to its depth and the large number of parameters it has. This can make it difficult to train and optimize without access to powerful computing resources. Additionally, the use of max pooling layers in VGGNet can result in a loss of spatial information, which can be important for certain computer vision tasks, such as object localization and segmentation [44]. Lastly, VGGNet may not generalize as well to other datasets or tasks as it is primarily developed and evaluated on the ILSVRC dataset.

2.3.5.4 ResNet

Originally suggested in 2015, the deep residual network, or ResNet, won first place in the classification of the ImageNet competition. It mainly tackles the problem of the accuracy of the training set decreasing as the network deepens, caused by the gradient vanishing. The core ResNet technique is introducing an identity shortcut to directly bypass one or more layers. A shallow network can achieve lower training errors by using an identity mapping of multiple layers, even though deep networks frequently have bigger training errors than shallow networks [44]. This answer shows that the layers of the identity map are more suitable for training since residual learning requires less computations than other layers. Additionally, the following is the mathematical expression for the residual unit:

$$y_1 = h\big(x_1 + F(x_1, W_1)\big) \tag{2.23}$$

$$x_{i_{+1}} = f(y_1) \tag{2.24}$$

Where x_l and $x_{l_{+1}}$ indicate the first residual unit's input and output, respectively; It is noteworthy that each residual unit typically possesses a multi-layered structure. f denotes the ReLU activation function, h signifies the identity map, and F represents the residual function, which encapsulates the learned residual. The learning characteristics go from shallow (l) to deep (L) layers as follows:

$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i, W_i) \tag{2.25}$$

The gradient of the reverse process can be determined using chain rules:

$$\frac{\delta loss}{\delta x_1} = \frac{\delta loss}{\delta x_1} \cdot \frac{\delta x_L}{\delta x_1} = \frac{\delta loss}{\delta x_1} \cdot \left(1 + \frac{\delta}{\delta x_L} \sum_{i=l}^{L-1} F(x_i, W_i)\right) \tag{2.26}$$

When the loss function of the first factor becomes close to the gradient of L, the short-circuit mechanism can transmit the gradient without losing any data, as indicated by the parenthesis. The other residual gradient is not directly conveyed; instead, it must pass through the layer with weights.

2.3.5.5 Faster R-CNN

Faster R-CNN is an enhanced network derived from Fast R-CNN and R-CNN. Subsequent to the convolutional layer, R-CNN and Fast R-CNN employ the ROI pooling layer, while the multitask loss function enables the direct integration of bounding box regression into

the CNN architecture. Nevertheless, utilizing Selective Search for Region Proposal generation requires more time for both testing and training. Faster R-CNN [44] replaces the prior selective search by effectively generating region proposals through the Region Proposal Network (RPN) and sharing the convolutional network with the object identification network. As a result, RPN reduces the recommendations from around 2000 to 300. Moreover, the quality of suggestion boxes has markedly enhanced [45]. The RPN produces 300 proposals for each image when the complete image is input into the Faster R-CNN for feature extraction. The recommendations are subsequently mapped into the last layer of the convolutional feature map. The ROI pooling layer generates a fixed-size feature map for each ROI. Bounding box regression and classification probability are trained with the SoftMax loss function and the smooth L1 loss function [45]. A fully connected layer with dimensions of 256 or 512 is generated on the convolutional feature map with a sliding window from the RPN. Moreover, Faster R-CNN inputs these variables into the classification and regression layers.

The classification layer is used to determine if the proposal is foreground or background. The regression layer predicts the breadth and center position of the proposal. In other words, the position of the sliding window of the RPN provides generic position information of the object, whereas the regression of the frame provides a more precise location of the frame [45]. Furthermore, Faster R-CNN's two fully connected layers handle classification and regression, respectively. Similarly, it makes the most extensive use of two loss functions for fine-tuning.

In a mini-batch, if i is the index of an anchor, then pi is the probability that the item is the object; pi is the anchor i's ground-truth label; and ti is a four-element vector where the

method for the classification task places bounds. The ground truth box's parameterized coordinates, $ti\ *$, coupled with a positive anchor, are represented as follows:

$$L(\{P_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_{i} \quad L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_{i} \quad p_i^* L_{reg}(t_i, t_i^*) \tag{2.27}$$

The parameterizations of the four coordinates for the regression are as follows: w and h represent the proposal box's width and height, respectively; x and y represent the proposal box's center coordinates. The variables for the forecast box, anchor box, and ground-truth box are x, x_a, and x^ast, respectively. An anchor box's bounding-box regression to a nearby ground-truth box [93] is

$$t_x = \frac{x-x_a}{w^a} \qquad t_y = \frac{y-y_a}{h^a} \qquad t_w = log\ log\left(\frac{w}{w_a}\right) \qquad t_h = log\ log\left(\frac{h}{h_a}\right) \tag{2.28}$$

$$t_x^* = \frac{x^*-x_a}{w^a} \quad t_y^* = \frac{y^*-y_a}{h^a} \qquad t_w^* = log\ log\left(\frac{w^*}{w_a}\right) \qquad t_h^* = log\ log\left(\frac{h^*}{h_a}\right) \tag{2.29}$$

2.3.5.6 SegNet

The SegNet image segmentation model, created by the University of Cambridge, partitions an image's object regions into pixels (e.g., vehicles, roadways, pedestrians, etc.). SegNet is an encoder-decoder architecture that utilizes the VGG-16 convolutional layer as five encoders, comprising 13 convolutional layers, following the removal of the fully connected layers [46]. The decoder executes nonlinear up-sampling of the input feature map utilizing the largest pool index accepted by the corresponding encoder [46]. The penultimate encoder is paired with the second decoder, the final encoder with the first, and so on. The decoder's output is transmitted to a multistage SoftMax classifier, which executes

classification based on pixels [94]. The encoder in SegNet incrementally enhances the feature map's resolution to match the input size while maintaining the spatial resolution of the feature map through the application of batch normalization and ReLU procedures. The resultant information is forfeited. SegNet further obtains and retains boundary information within the encoder map before the upsampling process. Due to translation invariance, maximal merging and upsampling can enhance classification tasks' resilience without jeopardizing image segmentation accuracy. SegNet exhibits superior efficiency compared to many alternative segmentation methods, as it solely retains maximum pooling indices, which are subsequently utilized in the decoder network to ensure optimal performance.

For instance, DeconvNet and its semi-supervised counterpart, the decoupled platform, utilize nonlinear up-sampling in the decoder network by leveraging the maximum position of the encoder feature map. These designs operate autonomously on SegNet. Approximately 90% of the characteristics in the VGG-16 network are attributed to the fully connected layer of the encoder network. Zone suggestions and additional auxiliary tools are essential as they complicate network training significantly. Furthermore, this substantially augments the duration devoted to contemplation [46]. Rather than utilizing pooled location index data, U-Net conveys the complete signature to the matching decoder, frequently resulting in increased memory usage. Moreover, by neglecting high-resolution feature maps for end-to-end processing, FCN heightens the risk of losing edge information. The encoder profile's multiplexing in the FCN, coupled with training challenges, renders it highly memory-intensive. SegNet is primarily motivated by the utilization of scenario information, leading to designs that prioritize memory and

computational efficiency during prediction. Furthermore, quantitative analyses indicate that SegNet exhibits superior efficiency compared to alternative architectures for memory and time utilization.

2.3.5.7 MobileNet

MobileNet is a convolutional neural network architecture created by Google in 2017. It is intended to facilitate efficient inference on mobile and embedded devices with constrained processing resources. MobileNet attains this efficiency by the utilization of depth-wise separable convolutions, which distinguish between spatial filtering and channel-wise filtering operations within a convolutional layer [47]. This decreases the quantity of parameters and computational demands of the model, while preserving a similar degree of accuracy to larger, more intricate models. MobileNet is exceptionally efficient and suitable for mobile and embedded devices with constrained computational resources, rendering it ideal for real-time applications like object identification and tracking [47]. The implementation of depth-wise separable convolutions in MobileNet diminishes the parameter count and computational demands of the model, resulting in expedited training and inference durations. Ultimately, MobileNet attains a degree of accuracy equivalent to that of larger, more intricate models, rendering it an optimal selection for resource-limited settings where precision remains significant. Conversely, the efficiency of MobileNet is achieved at the expense of reduced accuracy relative to larger, more intricate models. Although MobileNet can attain a significant degree of accuracy, it may underperform compared to larger models on some applications or datasets [47]. MobileNet may be less appropriate for jobs necessitating high spatial resolution or intricate feature extraction, owing to its implementation of depth-wise

separable convolutions. The depth-wise separable convolutional operations employed in MobileNet can be more challenging to optimize and calibrate compared to regular convolutional operations, potentially complicating the attainment of optimal performance on certain tasks.

2.4 Traffic Sign Classification Techniques

Traffic Sign Classification techniques apply different machine learning techniques in training and

testing of models. Deep learning, reinforcement learning, supervised learning, and unsupervised learning are some of the categories into which these methods can be divided. These techniques follow a specific path in classifying images and video frames. [48] notes that normally the path has three stages namely:

i). Image or video frame preprocessing,

ii). Feature extraction

iii). Classification.

2.4.1 Image or Video Frame preprocessing

Image and Video frame preprocessing involves a series of techniques and operations that are done on images to convert them to a format that can be analyzed. [49] and [50] notes that there are seven preprocessing activities that can be done to an image or video frame. These preprocessing activities are: image resizing, image cropping, color space conversion, image filtering, image normalization, image enhancement, and image segmentation.

2.4.1.1 Image resizing

Image resizing involves modification of an image or video frame to a given size. According to [16], resizing can be on the dimension (length, width and height (for three dimensions)) or resolution of the image. Additionally, [50] and [51] notes that image resizing can be done in two ways; resampling and Scaling. Resampling modifies the aspects ratio for an image or video frame while maintaining the original aspect ratio. [51], notes that when resampling is done on a digital image, pixels are either added or reduced. Further, [50] states that there are several resampling methods used in image resizing including nearest neighbor interpolation, bilinear interpolation, bicubic interpolation, Lanczos interpolation and Content-Aware Scaling.

Despite the fact that Nearest Neighbor Interpolation (NNI) has some shortcoming, it has the ability of preserving details of images and video frames with sharp edges [52]. [52], further notes that, NNI is a type of discrete sampling method that generates the value of a new pixel by choosing the nearest pixel value in the original image. [53], notes that NNI operates on an assumption that pixels for neighbors are similar, an assumption that may not always be true. [53] states that NNI replication of the value of the nearest pixel without smoothing or averaging its neighboring pixels, nearest neighbor interpolation maintains sharp edges and boundaries in an image. Therefore, NNI is helpful for images with distinct regions or items, such as line art, logos, and icons.

NNI can also add artifacts and noise into the resulting image, particularly when resized to a larger size. According to [52] and [53], this problem is due to the fact that NNI produces a blocky or pixelated appearance, which can be distracting or visually unappealing. They further note that NNI ignores the overall structure or patterns in an image, which can result

in inaccurate or distorted results. [52], suggests that anti-aliasing method; a method that can reduce jagged edges and smoothen the image, can help to reduce the blocky or pixelated appearance of nearest neighbor interpolation. This process is achieved through adding a low-pass filter on the image before resizing to remove high-frequency components or noise. However, [53] notes that anti-aliasing can improve picture quality, but it can also cause blurring or loss of sharpness depending on the strength of the filter used. Figure 2.1 shows an example of how NNI works on an image that was represented as a matrix of 3 by 3 to a matrix of 6 by 6

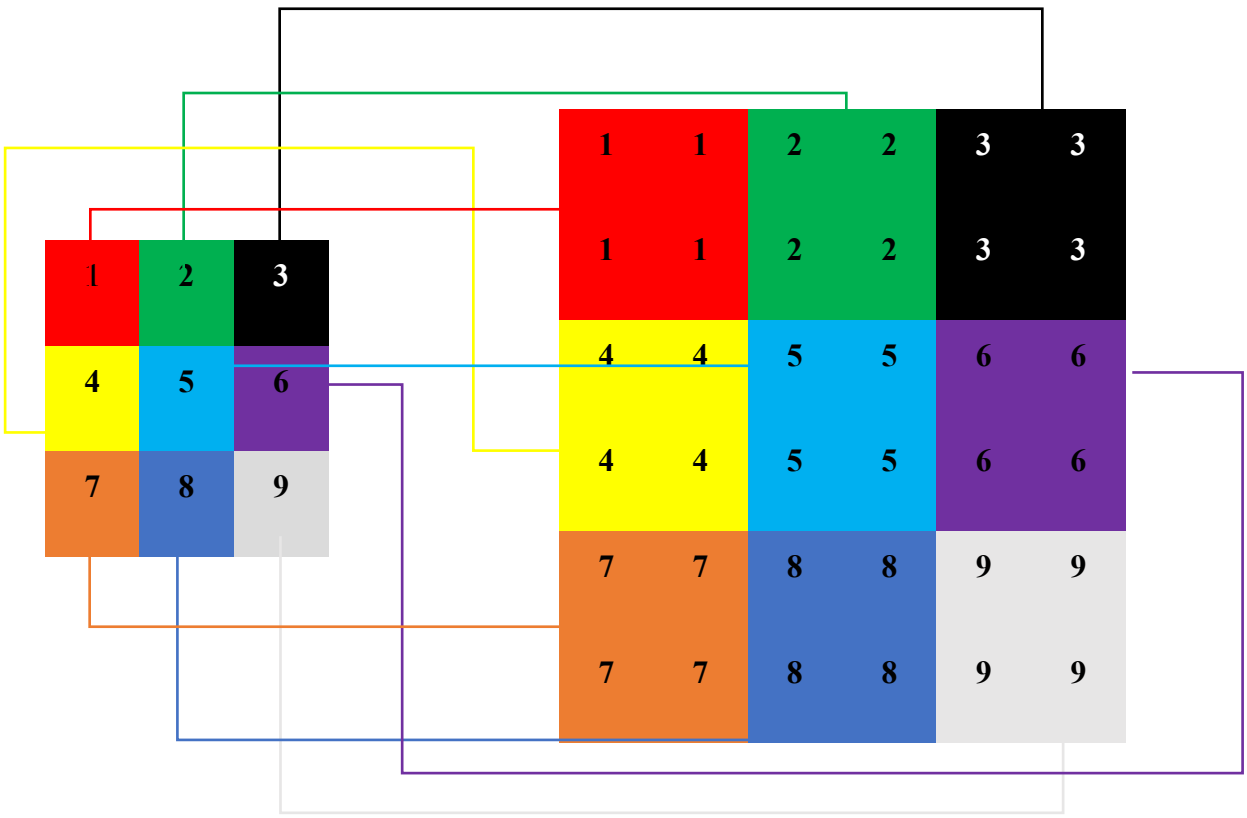


Figure 2.14 3 by 3 cell transformation to a 6 by 6 cell

Bi-linear interpolation is a technique for estimating function values between two points in a two-dimensional area. [54], notes that bi-linear is a straightforward and widely used method for image resizing, geometric transformations, and other computer graphics applications. Additionally, [55] notes that the fundamental idea behind bi-linear interpolation is to interpolate the values of the function at four known data points $(x_1, y_1), (x_1, y_2), (x_2, y_1)$, and (x_2, y_2) to get the value of a point(x, y). Notably, to accomplish this task, the algorithm first conducts linear interpolation along the x-axis between the points (x_1, y_1) and (x_2, y_1) to obtain $f1$ at the point(x, y_1), and similarly between the points (x_1, y_2) and (x_2, y_2) to obtain $f2$ at the point(x, y_2). The function's value at the location is then estimated using linear interpolation along the y-axis between the points (x, y_1) and (x, y_2) [55]. The algorithm for bi-linear interpolation is represented by equation 2.1

$$f(x,y) = (1-w)(1-h)f(x_1, y+1) + w(1-h)f(x_2, y1) + (1-w)hf(x_1, y_2) + whf(x_2, y_2) \quad (2.30)$$

Where w and h are the interpolation weights along the x and y axis, respectively. According to [21] w and h are determined as shown in equation 2.2.

$$w = \frac{x - x_1}{x_2 - x_1}$$
$$h = \frac{y - y_1}{y_2 - y_1} \quad (2.31)$$

Bi-linear interpolation produces a smooth estimate of the function while preserving the data's general trend. It may, however, be inaccurate in areas where the function changes

quickly or has sharp features [55]. Figure 2.2 below shows an example of a 2 by 2 image that has been converted to a 4 by 4 image using Bi-linear interpolation.

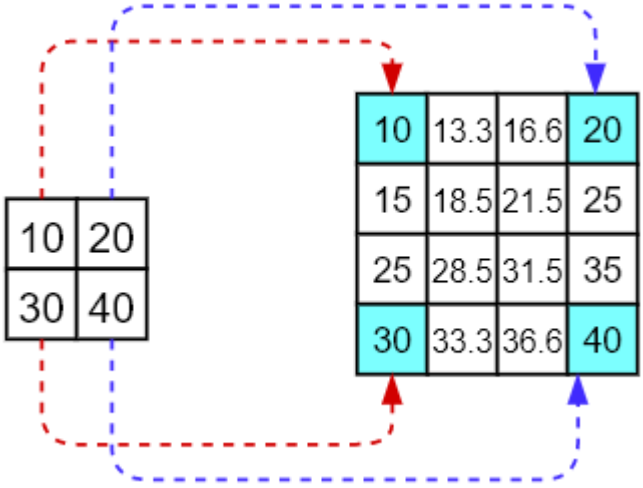


Figure 2.14: Bi-linear Interpolation

Another interpolation method is Bi-cubic interpolation. Bi-cubic is a method used to estimate the value of a function at a point within a grid or dataset by considering the values of the surrounding 16 points. According to [56], we can represent the interpolated value $(F(x,y))$ at a specific location (x,y) within the grid using a bi-cubic polynomial given by equation 2.3

$$\boldsymbol{f(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij} \cdot x^i \cdot y^j} \tag{2.32}$$

Where i and j range from 0 to 3 (representing the cubic nature of the function in both x and y directions). a_{ij} are the coefficients to be determined.

[56] further notes that these coefficients are calculated by solving a system of 16 equations derived from the known data points surrounding the desired location ($4x4$ grid) and their

first and second-order derivatives. There are two variations of bi-cubic interpolation: B-spline interpolation and the Lanczos resampling [56].

B-spline applies a specific type of spline known for their smoothness and ease of computation. It offers a good balance between computational efficiency and quality of interpolation. This variation is given by equation 2.4

$$F(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} C_{i,j} \cdot N_{i,p}(x) \cdot N_{j,p}(y) \tag{2.33}$$

Where $C_{(i,j)}$: control points (coefficients determining the shape of the curve), $N_i, p(x)$, $N_j, p(y)$: B-spline basis functions evaluated at x and y, respectively and p is thr degree of the polynomial (cubic for bicubic interpolation). Figure 2.3 is an illustration of the B Spline

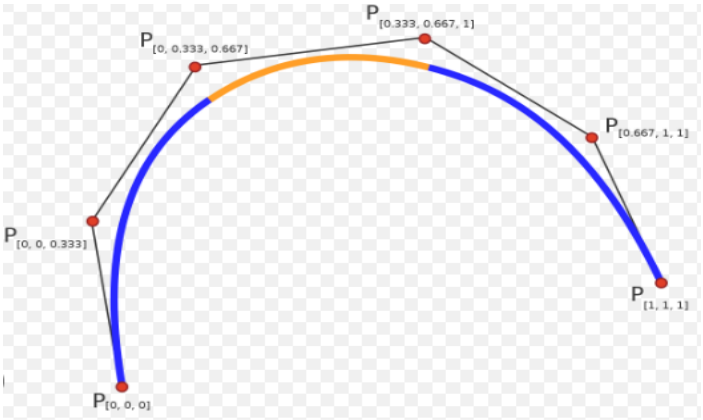


Figure 2.15: B Spline variation

Compared to the B Spline variants, Lanczos resampling can be computationally more costly. Lanczos interpolation is a type of windowed sinc interpolation that is commonly used in image processing and computer graphics for image scaling and resampling. [57], notes that it was introduced by Charles Lanczos in 1964 and is known for its ability to produce high-quality, sharp images with reduced aliasing artifacts. Lanczos interpolation

works by using a sinc function as the interpolation kernel, which was a function that was ideal for reconstructing a continuous signal from its samples. According to [58], the sinc function has infinitely long tails, which can result in ringing artifacts in practice. To overcome this issue, [59] notes that Lanczos interpolation uses a windowing function, typically a truncated sinc function, to limit the extent of the sinc function and reduce ringing artifacts.

Notably, the Lanczos interpolation algorithm takes a weighted average of neighboring pixels in the original image to estimate the value of a pixel in the resized image [57]. [58], observes that the truncated sinc function is used as the windowing function to determine the weights based on the distance between the target pixel and the surrounding pixels. Additionally, [59] points out that the window's size, sometimes referred to as the filter size, influences the trade-off between ringing artifacts and sharpness and dictates how many nearby pixels are used for interpolation. [58], argues that larger filter sizes can result in sharper images but may also introduce more ringing artifacts. Lanczos resampling is represented by equation 2.5

$$L(X) = sinc(x) * sinc\left(\frac{x}{a}\right) \tag{2.34}$$

Where $L(x)$ represents the Lanczos filter function, $sinc(x)$ the sinc function, defined as $sin(pi * x)/(pi * x)$, has a central peak and oscillating tails and $sinc(x/a)$ a second sinc function, windowed by a factor of $'a'$, which controls the extent of the filter kernel. Figure 2.4 illustrates Lanczos interpolation on a 6 by 6 cell.

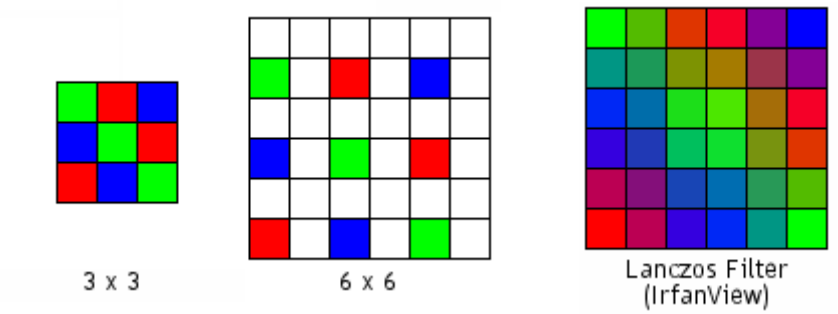


Figure 2.16: Lanczos interpolation

One of the key advantages of Lanczos interpolation is its ability to reduce aliasing artifacts, which are distortions that can occur when a continuous signal is sampled and reconstructed at a lower resolution. Additionally, [59] agrees that aliasing artifacts can result in jagged edges, more patterns, and other visual distortions in resized images. However, Lanczos interpolation also has some limitations. According to [58], it can be computationally expensive compared to simpler interpolation methods, such as bilinear or nearest neighbor, due to the calculation of the sinc function and the windowing function. The choice of the filter size also requires careful consideration, as larger filter size may result in increased computational complexity and may not always yield better results.

Content-Aware Scaling (CAS) is an advanced image scaling technique that selectively resizes the less important regions of an image while preserving the important content. [60], notes that CAS is commonly used in image processing and computer graphics to automatically resize images in a way that maintains the important visual elements while reducing or expanding the less important areas. [61], notes that CAS works by analyzing the content of the image and identifying areas that are less visually significant or less important to the overall composition. These areas can include plain backgrounds, sky, or other less detailed regions. CAS then applies a resizing algorithm that prioritizes the

important content while minimizing distortion and loss of image quality in the preserved regions. Notably, there are several algorithms and techniques used in CAS, and the exact implementation may vary depending on the specific software or application being used.

However, the general idea is to selectively scale the less important regions of the image in a way that preserves the overall visual content and maintains the original image's aspect ratio. [61], further notes that a common approach used in CAS is based on seam carving.

Seam carving is an image processing technique that involves finding and removing or adding seams, which are paths of pixels with the least energy, in the image [60]. The energy of a pixel can be calculated based on various factors such as color gradients, contrast, and other visual features. By removing or adding seams in the less important regions of the image, content-aware scaling can automatically resize the image while preserving the important content [61]. Another approach used in content-aware scaling is based on patch-based methods [62]. These methods involve finding similar patches in the image and blending them together to fill the gaps created by resizing. The blending process can be done using various techniques, such as patch synthesis, texture synthesis, or other image blending methods. CAS has several advantages over traditional scaling methods, such as simple interpolation or pixel replication [62]. It can automatically preserve the important content in an image, maintain the aspect ratio, and reduce the distortion and loss of image quality in the preserved regions. It is particularly useful for images with important visual elements that need to be maintained, such as in photographs with subjects or objects that need to be preserved while resizing the image.

However, like other image processing techniques, Content-Aware Scaling also has limitations. [62], notes that CAS may not always produce perfect results, especially in

images with complex content, fine details, or areas that were visually similar but have different meanings. Artifacts such as visual discontinuities, blurring, or color shifts may still occur, and careful adjustments or manual editing may be required to fine-tune the results. Additionally, [60] notes that the choice of algorithm and parameters used in CAS can also affect the quality of the results.

Therefore, the choice of resampling method depends on the specific application and the desired output. For example, if the image being resized is a simple graphic with sharp edges and no complex textures, nearest neighbor interpolation may be a good choice. On the other hand, if the image being resized is a photograph with complex textures and fine details, bicubic interpolation may be a better choice. It's worth noting that resampling can result in some loss of image quality, particularly when the image is resized to a larger size. This is because resampling involves adding or removing pixels, which can introduce artifacts and degrade image sharpness. To minimize this loss of quality, it's best to resize images in small increments rather than making large jumps in size, and to use high-quality resampling methods. Table 2.1 summaries interpolation methods discussed.

Table 2.2: Summary of interpolation methods applied to image pre processing

Method	Advantages	Disadvantages	Big(O)
Bilinear Interpolation	Simple to implement	May introduce blurring, especially during significant scaling.	$O(n)$
	Computationally efficient	Not suitable for sharp edges or high-frequency details.	
Nearest Neighbor	Very fast and efficient	Can appear blocky or pixelated, especially in scaling.	$O(n)$
	Preserves sharp edges and discontinuities	Loss of information from neighboring data points.	
Bicubic Interpolation	Superior smoothness compared to bilinear	Computationally expensive.	$O(n * k^2)$
	Potentially sharper edges compared to nearest neighbor	Susceptible to artifacts in rare cases.	
Lanczos Resampling	Excellent for preserving sharp edges and high-frequency details	Computationally expensive compared to other methods.	$O(n * k^2)$
	Reduces aliasing artifacts	Potential for faint ringing artifacts in rare cases.	
Content-Aware Scaling	Preserves important details and image composition	Computationally expensive.	$O(n * k^2)$
	Reduces unwanted artifacts	Limited user control over specific areas being modified.	
		Potential for errors in complex images.	

From table 2.1, bilinear and nearest neighbor had the best worst case time complexity of O(n) while Bicubic, Content-Aware Scaling and Content-Aware Scaling had $O\ (n * k2)$. Therefore, with increase in input size, bilinear and nearest neighbor were more efficient compared to the other three. On computational expenses, the study noted that content aware scaling required the most computational power followed by lanczos resampling. Bi cubic

also required a relatively high computational power as compared to nearest neighbor. Bilinear required the less computational power. Another area the study researched on was the ability to maintain sharp edges. The study noted that despite the fact that bilinear was the most efficient and required less computational power, it was the worst in preserving sharp edges. Bilinear was followed by nearest neighbor, bi cubic and lanczo resampling. Notably content aware scaling was the best in maintaining sharp edges.

The study also noted that various interpolation methods could be combined with the aim of getting a good balance between efficiency and maintenance of sharp edges. Identification and maintenance of sharp edges play critical role in the accuracy of traffic sing recognition while time complexity has a role in the efficiency.

2.4.1.2 Image cropping

Image cropping is a common technique used in image processing and video frame analysis for various purposes, such as improving visual aesthetics, removing unwanted elements, focusing on a specific Region of Interest (ROI), or creating a desired aspect ratio. According to [63], in image processing, cropping can be applied to images to extract a specific region or object of interest. [64], further gives an example in object recognition tasks, where the goal was to detect and classify objects within an image. [63], notes that cropping can be used to isolate the object from the background, reducing the computational load and focusing on the relevant region for further analysis.[56] and [63] agrees that in video frame analysis, cropping can be used to extract video frames or image sequences from a video stream. For example, in video surveillance or action recognition tasks, where the goal is to analyze human activities or events in a video, cropping can be used to extract

frames or image sequences containing the relevant activities or events for further analysis or classification.

Image cropping in image processing and video frame analysis is typically done using software or programming tools that provide cropping functionalities, such as image editing software, video editing software, or programming libraries for image or video processing. [64], notes that the specific cropping technique or algorithm used may vary depending on the application and the desired outcome. There are several ways for image cropping including region of interest (ROI), aspect ratio adjustment, removal of unwanted elements, content-based image retrieval, video thumbnail generation, and automated cropping.

Region of Interest (ROI) extracts a specific place or section on an image or a video frame. [65], notes that ROI is mainly performed on an image with the intentions of retrieving relevant information such as an object or a person face or getting a specific area that requires further analysis. This type of cropping has various application including object and facial recognition or image analysis. Another cropping technique is the Aspect Ratio Adjustment (ARA). [65] notes that Aspect Ratio Adjustment can help in optimizing an image size which can aid in different ways including rendering and display on different devices, or for specific visual aesthetics. [65] gives an example of how aspect ratio adjustment was used to perform square cropping that was mainly applied to pictures for social media and another example of how panoramic views applies wide cropping.

Another application of cropping is the removal of unwanted elements from an image or a video frame. [66], notes that this cropping can be used to remove distractions from an image for better processing. [66], gives an example on how this way of cropping can be used to remove background of an image to enable better processing of the image. This

cropping is applied to improve the composition and focus of the image or frame. One can apply another way of cropping that retrieves different parts of an image and video frame using its content called Content-Based Image Retrieval (CBIR). In CBIR, images were retrieved based on their visual content. [66], notes that cropping can help in generating multiple sub images or patches from an original image. This sub images and patches can then be used as queries or templates for searching similar images in a database that plays an important role for more precise and targeted image retrieval based on specific image regions or features.

In video thumbnail generation cropping is commonly used where a single frame from a video is selected as a representative thumbnail image. [67], notes that this cropping allows one to select an interesting or representative part of the video frame that can be used as a thumbnail, providing a visual preview of the video content. Finally, in some cases, image cropping can be done automatically using algorithms or techniques such as object detection, face detection, or saliency detection. [68], notes that these automated cropping techniques can analyze the visual content of an image or video frame and automatically determine the region of interest to be cropped based on predefined criteria or learned patterns.

Table 2.3: Summary of cropping techniques applied to image pre processing

Feature	Strength	Weakness	Big O notation
Region of Interest	Focuses processing on specific areas of interest, reducing computational load. Useful for applications where specific regions carry important information.	Requires accurate detection and localization of regions of interest. May miss important information if the ROI detection is inaccurate.	O(n) (linear)
Aspect Ratio Adjustment	Preserves the aspect ratio of the image, preventing distortion. Useful for maintaining visual fidelity in various applications such as multimedia display.	May result in cropping or distortion of the image if the aspect ratio adjustment is not carefully implemented. Involves additional computational overhead to calculate and adjust the aspect ratio.	O (1) (constant)
Content-based Image Retrieval	Retrieves images based on visual content rather than relying solely on metadata. Useful for searching and organizing large image databases efficiently.	Performance highly dependent on the effectiveness of feature extraction and similarity measurement methods. Can be computationally intensive, especially for large image databases.	O(n) or higher

From related literature, aspect ratio adjustment was the best in terms of time complexity. Aspect ratio adjustment had O (1) meaning that the time it takes to pre –process an image remains constant regardless of the input size. Region of interest was the second one with a time complexity of O (n). This is a linear function and the time for preprocessing increased linearly with increase in input size. Content-based Image retrieval had different time complexity that depended on the feature extraction algorithm. This time complexity for content-based image retrieval was either O (n) or higher. In terms of computational power, Region of Interest required less computational power compared to the other two methods. Aspect ratio adjustment required slightly more computational power while content-based

image retrieval required the most computational power. Nonetheless there was documented evidence that showed that some algorithms for content-based image retrieval were the best in maintaining the required area for feature extraction followed by region of interest and finally aspect ratio adjustment. Notably, aspect ratio adjustment maintenance the image aspect ratio hence reduces chances of distortion.

2.4.1.3 Color space conversion

Color space conversion is an essential technique in image processing and computer vision that involves transforming the representation of colors from one color space to another. A color space is a mathematical model that describes how colors are represented as a combination of primary colors, such as Red, Green, and Blue (RGB), or Hue Saturation, and Value (HSV) [69]. Converting an image from one color space to another can be useful for various image processing tasks, such as color correction, color enhancement, and feature extraction.

The RGB color space is the most commonly used color space for digital images [69]. However, the RGB color space has some drawbacks, such as its dependence on the device's color characteristics and its inability to separate luminance and chrominance components. As a result, converting images to other color spaces can provide benefits in certain scenarios. One commonly used color space conversion technique is converting RGB images to grayscale images. This is achieved by taking the weighted average of the R, G, and B channels to obtain a single channel grayscale image. Grayscale images are often used in computer vision applications such as edge detection and object recognition.

Another popular color space conversion technique is converting RGB images to the HSV color space. For some image processing tasks, like color-based object detection and

segmentation, the HSV color space's separation of color information into hue, saturation, and value channels may be more understandable [70]. In the HSV color space, the hue channel represents the color information, the saturation channel represents the intensity or purity of the color, and the value channel represents the brightness or lightness of the color.

Other color spaces, such as the Y'CbCr and Lab color spaces, are commonly used for image and video compression, as they can separate luminance and chrominance information, which can be compressed at different rates [69]. YCbCr separates the image into three components: Y' (luma), Cb (blue chroma), and Cr (red chroma).

Notably, color space conversion is an important technique in image processing and computer vision. It allows for the representation of colors in a way that is better suited for specific image processing tasks. Converting images to grayscale or to other color spaces, such as HSV, YCbCr, and Lab, can provide benefits in various scenarios, and understanding the advantages and disadvantages of different color spaces is crucial in choosing the appropriate color space for a particular task. Table 2.3 summaries the findings of color space conversion literature.

Table 2.4 Image conversion methods

Conversion method	Strength	Weakness	Big O Notation
RGB to Grayscale	Simple and straightforward conversion. Preserves image luminance information	Ignores color information, resulting in loss of chromatic details. May not be suitable for applications where color plays a significant role	O (n)
RGB to HSV	Separates color information from brightness, making it easier to manipulate colors independently. Useful for various image processing tasks such as color segmentation and analysis.	Conversion process involves complex mathematical calculations, leading to increased computational overhead. May result in information loss due to quantization of hue values.	O (1)
RGB to Y'CbCr	Separates color and luminance information, making it suitable for image and video compression. Y' channel represents grayscale information, which is useful for applications requiring grayscale images.	Conversion process involves additional computational complexity compared to other color spaces. May require additional processing steps to convert back to RGB for display purposes	O (1) or O (n)
RGB to Lab	Perceptually uniform color space, meaning uniform changes in Lab values correspond to uniform changes in perceived color. Suitable for applications involving color correction and image enhancement	Not as widely supported as other color spaces in various software and hardware systems. May result in slight perceptual differences compared to the original RGB image.	O (1) or O (n)

The study used RGB because its extraction was fast and had a constant time complexity O (1). However different task required to store different value; that's why conversion from RGB to other color schemes was necessary. Notably, conversion to HSV had the best worst case time complexity of O (1). This was followed by conversion to lab and Y'CbCr which

had O (1) and above depending on the algorithm used for conversion. Conversion to grayscale had 0(n). Nonetheless, gray scale required the least computational power followed by HSV. Lab required more computational power compared to HSC while Y'CbCr required the most computational power. It was also noted that grayscale was discarding color information, while Y'CbCr stored the grayscale information in y'. In conclusion the choice of the color scheme to use depended on the problem being solved.

2.4.1.4 Image filtering

Image filtering is a fundamental operation in image processing that involves manipulating the intensity values of pixels in an image to achieve various objectives, such as noise reduction, edge enhancement, and feature extraction [71]. The main goal of image filtering is to remove unwanted information from the image while preserving the relevant features. There are several types of image filters that are commonly used in image processing, including linear filters, nonlinear filters, and frequency domain filters [72].

Linear filters, such as the Gaussian filter and the mean filter, are used to smoothen an image and reduce noise. Nonlinear filters, such as the median filter and the bilateral filter, are used to preserve edges while removing noise [71 -72]. Frequency domain filters, such as the Fourier transform and the wavelet transform, are used to enhance or suppress specific frequencies in the image.

In recent years, deep learning-based image filtering methods have gained popularity due to their ability to learn from data and adapt to various image processing tasks [71]. Convolutional Neural Networks (CNNs) have been used for various image filtering tasks, including de-noising, de-blurring, and super-resolution. The use of CNNs allows for the

learning of complex filters that are optimized for the specific task at hand, leading to improved performance over traditional filtering methods.

Image filtering is a critical operation in image processing that has been extensively studied and applied in various computer vision tasks. Linear, nonlinear, and frequency domain filters that are commonly used in image filtering, and deep learning-based methods such as CNNs have shown promising results in improving the performance of image filtering tasks.

Table 2.5 Summary of Image filtering methods

Filtering Method	Strength	Weakness	Big O notation
Linear Filters	Efficient for smoothing and noise reduction tasks. Preserves edges and fine details to some extent	Limited effectiveness for noise removal, especially for impulse noise. May blur image details and edges.	O (n^2)
Non-Linear Filters	Effective for removing various types of noise, including impulse noise. Can preserve edges and textures better than linear filters in some cases.	Computationally intensive, especially for large kernel sizes or complex filter operations. May introduce artifacts or distortions in the image.	O (n^2)
Frequency Domain Filters	Efficient for frequency-based operations such as high-pass, low-pass, and band-pass filtering. Useful for tasks such as sharpening, blurring, and denoising.	Transformation between spatial and frequency domains involves additional computational complexity. Loss of spatial information during transformation.	O ($n^2 \log n$)
Deep Learning Based Image Filters	Can learn complex patterns and features directly from data. Adaptability to different types of noise and image characteristics.	Requires large amounts of annotated data for training. Computationally intensive during training and inference.	Varies depending on the architecture and complexity of the neural network.

From table 2.4, depending on the algorithm used, deep learning-based Image filtering had the best worst case time complexity. Filtering could be achieved with time complexity of O (n) up to O (n^3). Frequency Domain filter was the second with a time complexity of O (n^2 log n). Both linear and nonlinear filters had a quadratic time complexity of O (n^2). One key reason for image filtering is reduction and removal of noise. Deep learning-based image filtering was the based because it was adaptable to types of noise hence could perform better in different scenarios. Frequency Domain filters worked based with problems that required sharpening, blurring and de-noising. Non –linear was effective in removing various types of noise but not all noise while linear filters were efficient for smoothing and noise reduction.

Also, Linear filters were less effective in removal of noise because they cause blurring of images. On the other hand, nonlinear had the problems of introducing artifacts and required a high computational power. Frequency domain was responsible for loss of spatial information during transformation from spatial to frequency domain. This transformation required additional computational power. Lastly, deep learning required the most computational power among all Image filter methods. Deep learning also required a large amount of annotated data for training which is an additional step.

Therefore, the choice of a good image filter method depends on the task at hand but also several methods can be combined to achieve a best fit between computational resources and noise removal.

2.4.1.5 Image normalization

A popular pre-processing method for enhancing the consistency and quality of images in computer vision applications is image normalization [72]. In order to improve the accuracy

and dependability of ensuing image processing and analysis, image normalization aims to make the image content more invariant to variations in lighting, contrast, and other imaging conditions. The literature has put forth a number of methods for image normalization, such as spatial normalization, contrast stretching, and histogram equalization.

Histogram equalization is a simple yet effective technique that redistributes the pixel intensities of an image to achieve a uniform histogram, thereby enhancing the contrast and dynamic range of the image. However, it suffers from the drawback of producing unnatural-looking images with exaggerated contrast and artifacts. Contrast stretching is another popular technique for image normalization, which works by linearly scaling the pixel intensities of an image to the full range of values [73]. This technique is simple and computationally efficient, and can effectively enhance the contrast of images with low contrast or dynamic range. However, it may also produce unnatural-looking images with saturated pixels and loss of detail in some regions.

Spatial normalization is a more advanced technique for image normalization, which involves aligning and warping the images to a standard reference space, based on anatomical or functional features [73] and [74]. This technique enhances the precision and uniformity of picture processing across several patients or imaging modalities and is extensively utilized in medical imaging applications, including brain imaging and Functional Magnetic Resonance Imaging (fMRI). Techniques based on deep learning have been suggested for picture normalization, capable of directly learning normalization functions from data through convolutional neural networks (CNNs). These approaches have demonstrated encouraging outcomes in diverse computer vision tasks, including

object detection and segmentation, and can autonomously adjust to varying imaging settings and data distributions [74].

Image normalization is an important pre-processing technique for improving the quality and consistency of images in computer vision applications. Various techniques have been proposed in the literature, ranging from simple histogram equalization and contrast stretching to more advanced spatial normalization and deep learning-based approaches. The approach selected may need thorough parameter tuning and evaluation, depending on the particular application needs and imaging conditions.

Table 2.6 Summary of Image normalization techniques

Normalization Method	Strength	Weakness	Big O notation
Histogram Equalization	Enhances image contrast by redistributing pixel intensities across the histogram. Improves the visibility of details in both bright and dark areas.	May amplify noise and artifacts in the image. May result in unnatural-looking images if the histogram is not properly adjusted.	O (n)
Contrast Stretching	Simple and intuitive method to enhance image contrast. Widely used for real-time applications due to its computational efficiency.	Limited effectiveness in cases where the dynamic range of pixel intensities is already well-distributed. May lead to loss of information in areas with extreme intensity values.	O (n)
Spatial Normalization	Corrects geometric distortions and aligns images spatially. Useful for improving consistency in images for further processing.	Performance may degrade for images with significant geometric distortions. Computational complexity may increase for large-scale spatial normalization tasks.	O (n^2)
Deep Learning-based Normalization	Can learn complex normalization functions directly from data. Adaptability to various imaging conditions and modalities.	Requires large amounts of annotated data for training. Computationally intensive during training and inference.	Varies depending on the architecture and complexity of the neural network

From table 2.5, spatial normalization had the worst time complexity of quadratic time. Contrast stretching and histogram equalization had O (n) while deep learning-based methods had the ability to produce the best time complexity depending on the algorithm used. Similarly to image filters, deep learning-based methods required a large data set that

is annotated. They use this large data set at the training stage to make them adaptable to various imaging conditions. Spatial normalization has the ability to correct distortions which helps in improving consistency. Contrast stretching is the most used method because of its computational efficiency. Finally, histogram equalization improves details that enhances both the bright and dark areas.

One needs to understand the task at hand, the computational power in order to select the best method for image normalization. Different researches combined different methods for better output in terms of computational resources and reliable results.

2.4.1.6 Image segmentation

Image segmentation is the procedure of partitioning an image into several segments or regions, each corresponding to a distinct object or component of the image. This task is essential in computer vision and image processing, significantly impacting applications like object recognition, image analysis, medical imaging, and video surveillance [75]. Numerous techniques for picture segmentation have been created over the years, encompassing basic thresholding methods as well as advanced approaches utilizing statistical models, machine learning, and deep learning.
The region-based approach is a widely utilized method for image segmentation, partitioning the image into parts according to the similarity of pixel values or other characteristics [75]. Another prevalent method for image segmentation is the edge-based technique, which seeks to identify the boundaries between distinct objects or regions within the image [75]. This method relies on the observation that intensity values of neighboring pixels frequently exhibit rapid changes at object boundaries, which can be identified using edge detection methods like the Canny edge detector and the Sobel operator.

Convolutional neural networks (CNNs), one type of deep learning approach, have recently demonstrated considerable potential for image segmentation, obtaining state-of-the-art performance on numerous benchmark datasets. Typically, these techniques entail using a CNN to predict the segmentation masks for new images after it has been trained on a sizable dataset of labeled images. Managing photos with complex and varied content, such as pictures with several objects, occlusions, and different lighting conditions, is one of the biggest issues in image segmentation. To address this challenge, many researchers have proposed hybrid methods that combine multiple segmentation techniques, or use additional information such as depth or motion cues to improve segmentation accuracy.

Image segmentation is a fundamental task in computer vision and image processing, and has been the focus of extensive research over the years. Many different techniques have been developed for image segmentation, ranging from simple thresholding methods to sophisticated deep learning approaches. The choice of segmentation method depends on the specific application and the characteristics of the images being analyzed.

Table 2.7 Summary of Image segmentation methods

Segmentation Method	Strength	Weakness	Big O Notation
Region-based Approach	Effective for segmenting regions with uniform characteristics such as color or texture. Robust to noise and small variations within regions.	May struggle with images containing regions with similar characteristics or overlapping boundaries. Sensitivity to initial seed selection.	O (n^2)
Edge-based Approach	Accurate in delineating object boundaries based on changes in intensity or color gradients. Suitable for images with well-defined edges and contours.	Prone to over-segmentation or under-segmentation, especially in regions with low contrast or ambiguous edges. Performance highly dependent on edge detection algorithms.	O (n^2)
Deep Learning Techniques	Can learn complex patterns and features directly from data. Adaptability to various imaging conditions and modalities.	Requires large amounts of annotated data for training. Computationally intensive during training and inference.	Varies depending on the architecture and complexity of the neural network

From table 2.6, deep learning techniques had the possibilities of producing the best worst case time complexity, depending on the algorithm used. Region based approach and edge-based approach had the same time complexity of O (n^2). Notably, region-based approach is effective for segmenting regions with uniform characteristics additionally, it is robust to noise and small variations within same regions. Edge based Approach were accurate in delineating object boundaries based on changes in intensity or color gradients. They were suitable for images with well-defined edges and contours. Finally, deep learning techniques can learn complex patterns and features directly from data making them Adaptable to various imaging conditions and modalities.

2.4.2 Feature extraction and Classification

Texture, edges (shape) and other hand felt characteristics of an image are the basis of Feature Extraction and Classification (FEC). [76], [77], [78] and [79] gives examples of different methods that are used for feature extraction in FEC including Histogram of Oriented Gradients (HOG), Scale-Invariant Feature Transform (SIFT), Bag-of-Words (BOW), Local Binary Patterns (LBP) and Gabor filters. According to [77] and [79], the extracted features are arranged and prepared into a format that is used as an input to a machine learning algorithm such as random forest, logistics regression, K- nearest neighbor, and support vector machine.

2.4.2.1 Histogram of Oriented Gradients

The Histogram of Oriented Gradients (HOG) is a feature descriptor employed in computer vision and image processing for object identification and detection. Navneet Dalal and Bill Triggs introduced the HOG algorithm in their 2005 publication "Histograms of Oriented Gradients for Human Detection" [76]. The HOG technique pulls features from an image by partitioning it into small cells and calculating the orientation gradients of each pixel inside those cells. The gradients are subsequently quantized into discrete orientations, and the distribution of these orientations is depicted by a histogram. The histograms of all cells are combined to create a feature vector applicable for object detection and recognition.

Since its introduction, the HOG algorithm has been extensively studied and applied in various fields such as pedestrian detection, facial expression recognition, vehicle detection, and texture classification [76].

In 2005, Dalal and Triggs demonstrated the effectiveness of the HOG algorithm in detecting pedestrians in real-world images. They compared the performance of the HOG algorithm with other feature descriptors such as SIFT and showed that the HOG algorithm outperforms other feature descriptors in terms of accuracy and speed. Their work sparked a new wave of research in object detection using the HOG algorithm. Several researchers have proposed modifications to the HOG algorithm to improve its performance. For instance, [80] proposed a multi-scale HOG algorithm that can handle objects of different sizes. They achieved better performance in detecting small objects by using a finer grid for feature extraction in small scales. [81], proposed a HOG-based method for detecting salient objects in images. They used the HOG algorithm to extract features and combined them with saliency maps to improve the performance of salient object detection.

The HOG algorithm has been widely studied and applied in object detection since its introduction in 2005. Its effectiveness and efficiency have been demonstrated in various applications, and modifications have been proposed to improve its performance.

2.4.2.2 Scale-Invariant Feature Transform

Another well-liked feature descriptor for object detection and recognition in computer vision and image processing is Scale-Invariant Feature Transform (SIFT). David Lowe first presented the SIFT technique in his 1999 paper "Object recognition from local scale-invariant features" [77]. By identifying and characterizing important points that are unaffected by scale, rotation, and affine distortion, the SIFT technique is able to extract information from a picture. Finding local extrema in the Difference-of-Gaussian (DoG) scale space—a collection of blurred and subsampled images is how the algorithm initially identifies important locations. The algorithm then assigns orientations to each key point

based on the dominant gradient direction within a surrounding region. Finally, the algorithm describes each key point by computing a descriptor that captures the local image information around the key point. Since its introduction, the SIFT algorithm has been widely studied and applied in various fields such as object recognition, 3D reconstruction, and image retrieval [81].

To identify and detect objects in a picture, the SIFT technique is frequently used with a classifier like a random forest or Support Vector Machine (SVM). For example, Mikolajczyk and Schmid [82] suggested a technique that uses an SVM classifier with SIFT key points to identify and recognize objects in pictures. They performed well when it came to identifying and detecting things in crowded images. To enhance the SIFT algorithm's functionality, a number of researchers have suggested changes. [83] suggested a modified SIFT technique that increases the resilience of SIFT features to variations in light by utilizing local contrast normalization. In different lighting circumstances, they performed better when it came to object detection. More recently, [84] proposed a deep learning-based method for feature extraction that combines the strengths of SIFT and deep convolutional neural networks (CNNs). They achieved state-of-the-art performance in object recognition on several benchmark datasets.

The SIFT algorithm has been widely studied and applied in object detection since its introduction in 1999. Its effectiveness and robustness to scale, rotation, and affine distortion have been demonstrated in various applications, and modifications have been proposed to improve its performance. Although it has been largely replaced by deep learning-based methods in recent years, the SIFT algorithm still has its advantages, especially in scenarios where the object of interest has distinctive local features.

2.4.2.3 Bag-of-Words

The Bag-of-Words (BoW) model is a popular method for text and image classification in computer vision and natural language processing. The Bag of Words (BoW) method counts the frequency of each word or visual word in a document or image to generate a feature vector, thereby portraying each document or image as a compilation of its constituent words or visual words, respectively [85]. The Bag of Words approach has been extensively employed in picture classification and object recognition tasks. In image classification, the BoW model initially extracts visual characteristics, such as SIFT or SURF descriptors, from each image in the dataset. Subsequently, it generates a dictionary of visual words by aggregating the acquired visual features from all photos into a predetermined number of clusters utilizing techniques such as K-means clustering. Ultimately, it characterizes each image as a histogram reflecting the frequency of visual words inside the image, with the visual words being the cluster centers derived from the clustering phase.

The BoW model has been shown to be effective in various image classification tasks, such as Image recognition, object recognition, and face recognition [86]. In particular, the BoW model has been used in conjunction with classifiers such as support vector machines (SVMs) and random forests to achieve state-of-the-art performance in various benchmarks. The BoW model has also been applied in text classification tasks, where it represents each document as a bag of its constituent words. In this case, the BoW model creates a dictionary of words by collecting all unique words from the documents in the dataset [85]. Then, it represents each document as a histogram of the frequencies of the words in the document, where the words were the dictionary entries.

Numerous text classification tasks, including document classification, topic modeling, and sentiment analysis, have demonstrated the effectiveness of the BoW model [87]. Specifically, the BoW model has been utilized in combination with classifiers like SVMs and Naive Bayes to attain cutting-edge results across a range of benchmarks. The BoW model was a strong and adaptable technique for text and picture feature extraction and classification. It has demonstrated efficacy across a range of tasks and has been used with other classifiers to attain cutting-edge performance.

2.4.2.4 Local Binary Patterns

Local Binary Patterns (LBP) is a prevalent technique for texture analysis and recognition in computer vision. Initially presented by Ojala et al. in 1994 as a straightforward yet efficient texture descriptor, it has since undergone significant examination and application across several computer vision domains [88]. The LBP descriptor encodes local texture information of a picture by comparing each pixel to its neighboring pixels. The LBP operator evaluates the intensity of each pixel against its nearby pixels, assigning a binary value of 1 or 0 based on whether the neighboring pixel's intensity exceeds or falls below that of the central pixel [88]. The binary values of all pixels in the vicinity are concatenated to create a binary pattern that signifies the texture information of that pixel's surroundings.

The LBP descriptor has been shown to be highly effective in various computer vision tasks, including texture classification, facial expression recognition, and object recognition [88]. In texture classification, the LBP descriptor has been used in conjunction with classifiers such as SVMs and KNN to achieve state-of-the-art performance on standard texture datasets such as the Brodatz texture dataset [89]. In facial expression recognition, the LBP descriptor has been shown to be effective in capturing the local facial texture features and

has been used in conjunction with classifiers such as SVMs and random forests to achieve high recognition rates on standard facial expression datasets such as the CK+ dataset [89]. In object recognition, the LBP descriptor has been applied in conjunction with other feature descriptors such as SIFT and HOG to achieve state-of-the-art performance on standard object recognition datasets such as the PASCAL VOC dataset [89].

Numerous expansions and modifications of the LBP descriptor have been proposed in the literature to improve its performance and robustness across various applications. The Extended Local Binary Patterns (ELBP) descriptor [90] enhances the LBP operator by incorporating circularly symmetric patterns, whereas the Rotation-Invariant Local Binary Patterns (RI-LBP) descriptor [91] guarantees the LBP descriptor's invariance to rotation by encoding local texture information with rotation-invariant patterns. The LBP descriptor is a straightforward yet highly efficient technique for texture analysis and recognition in computer vision. The LBP descriptor has been thoroughly examined and utilized in numerous computer vision tasks, with various extensions and modifications proposed to improve its performance and robustness across diverse applications.

2.4.2.5 Gabor filters

Gabor filters, named after physicist Dennis Gabor, are a widely used method for image and signal processing in computer vision. They are based on the principle of decomposing an image into its constituent frequencies and orientations, similar to the human visual system [92]. The Gabor filter is a complex-valued kernel that convolves with an image to produce a filtered image. The Gabor filter is defined as a Gaussian function modulated by a complex sinusoidal wave. The Gaussian function is used to control the spatial extent of the filter, while the sinusoidal wave controls its frequency and orientation [92]. By varying the

parameters of the Gaussian and sinusoidal functions, the Gabor filter can be tuned to different frequency and orientation responses, making it a powerful tool for feature extraction in image processing.

Texture analysis, object recognition, and biometrics are just a few of the computer vision applications that have made substantial use of Gabor filters. In texture analysis, Gabor filters have been used to extract texture features that were sensitive to local frequency and orientation variations in the image [93]. These features have been used in conjunction with classifiers such as SVMs and KNN to achieve state-of-the-art performance on standard texture datasets such as the Outex dataset. In object recognition, Gabor filters have been applied in conjunction with other feature descriptors such as SIFT and HOG to achieve state-of-the-art performance on standard object recognition datasets such as the Caltech 101 and Caltech 256 datasets [93]. Gabor filters have also been used in biometric applications such as face recognition and iris recognition, where they have been shown to be effective in capturing the local texture information of the face or iris.

Various modifications and extensions of the Gabor filter have also been proposed in the literature to enhance its performance and robustness in different applications. For instance, the Dual-Tree Complex Wavelet Transform (DT-CWT) [94] extends the Gabor filter by using a wavelet decomposition of the image, while the Log-Gabor filter [95] was a variant of the Gabor filter that was designed to be more efficient in terms of computational complexity. Gabor filter was a powerful tool for image and signal processing in computer vision. It has been extensively studied and applied in various computer vision applications, and various modifications and extensions of the Gabor filter have been proposed to enhance

its performance and robustness in different applications. Table 2.7 Summaries handcrafted feature extraction techniques

Table 2.8 Summary of feature Extraction and Classification

Feature Extraction Method	Strength	Weakness	Big O notation
HOG	Effective for capturing shape and texture information in images. Robust to changes in illumination and background clutter.	May struggle with complex background textures and cluttered scenes. Sensitivity to variations in object appearance and viewpoint	O(n) or $O(n^2)$
SIFT	Provides distinctive and invariant local features robust to scaling, rotation, and affine transformations. Well-suited for object recognition and image matching tasks.	Computational complexity increases with the number of key points and descriptors. Requires careful parameter tuning for optimal performance.	O (n log n)
BoW	Efficient representation of image features using visual word histograms. Suitable for classification and retrieval tasks in large-scale image datasets.	Loss of spatial information due to histogram aggregation. Limited ability to capture fine details and geometric information.	O (n)
LBP	Simple and computationally efficient texture descriptor. Effective for texture classification and segmentation tasks.	Limited discriminative power compared to more complex descriptors. May struggle with complex texture patterns and variations.	O (n)
Gabor Filters	Sensitive to texture variations and edges in images. Useful for tasks such as texture analysis, segmentation, and edge detection.	Parameter selection and tuning can be challenging. Computationally intensive, especially for large filter banks and image sizes.	$O(n^2)$

From table 2.7, Bag of Words and Local Binary Patterns had the best worst case time complexity of O (n). Notably, Histogram of Oriented Gradients can be implemented in two ways: one giving a time complexity of O (n) and the other O (n^2). Scale Invariant Feature Transform that had O (n log n) while Gabor filter had the worst time complexity of O (n^2). The study also noted that, HOG was effective for capturing shape and texture information in images and had ability to adapt to changes in illumination and background. SIFT had ability to produce distinctive and invariant local feature robust to scaling, rotation and affine transformation. Bag of words had an efficient representation of image feature using visual word histograms while Local Binary Patterns was a computationally efficient texture descriptor. Gabor filters were sensitive to texture variation and edges in images. On the other hand, from documented evidence, HOG showed evidence of struggling when exposed to complex background, textures and cluttered scenes. SIFT computational complexity increases with the number of key points and descriptors. Bag of Word had loss of spatial information due to histogram aggregation. While LBP had limited discriminative power compared to more complex descriptors. For Gabor filters, parameter selection and tuning could be challenging.

2.5 Traffic Signs Analysis

Traffic sign analysis is a domain of computer vision and machine learning aimed at extracting valuable information from images or video frames captured on roads, highways, and other transportation networks. This data can be utilized in several applications, including autonomous driving, traffic monitoring, and transportation planning. Typical tasks in traffic sign analysis include object detection, lane detection, road segmentation, traffic flow analysis, anomaly detection, road condition evaluation, and vehicle counting

and classification [96]. The study primarily concentrates on the identification and detection of license plates and image text, with a particular emphasis on traffic text detection, traffic sign recognition and detection, and object detection.

2.5.1 Traffic Sign Recognition

In many applications, like driverless vehicles and Advanced Driver Assistance Systems (ADAS), the ability to gather data from different traffic signs is essential. The two linked problems of traffic sign detection (TSD) and recognition (TSR) are typically included in a traffic sign recognition system. The study's objectives are to correctly recognize traffic signs in pictures and classify the labels of those signs into distinct groups. Despite the topic's more than two decades of study interest in the computer vision field [96], there are still difficulties because of a number of complexities, including open-set environments, capture angles, and varied illumination conditions.

2.5.1.1 Traffic Sign Detection Methods

Other computer vision object detection issues resemble the Traffic Sign Detection Methods (TSD) challenge. Specifically, detecting the segments of the image that has bounding boxes that closely encapsulate traffic signs. Unlike ordinary objects, traffic signs typically feature vivid colors and simple, rigid forms. The spatial connection with other visual elements throughout the journey is an additional significant signal that could be utilized to enhance TSD [97]. Methods for segmenting traffic signs can be categorized into three groups: machine learning-based, color-based, and geometry-based. Color-based techniques utilize color information for object identification, demonstrating significant resilience against projective distortion and exhibiting minimal processing costs. A road sign recognition system utilizing HSI color space segmentation and SVM classification is detailed in [98].

Color feature clustering is utilized for image segmentation in [99]. Maximally Stable Extremal Regions (MSERs) are proposed as options for traffic signs in [100]. Subsequently, SVM classifiers trained on HOG [101] features are presented. A approach that amalgamates color, saliency, spatial, and contextual data is proposed in [102]. In [103], traffic sign candidates are generated by integrating the color probability model with MSERs. A hybrid region proposal method integrating Wave-based Detectors (WaDe) with MSERs is delineated in [104]. Conversely, initiatives such as the Radial Symmetry Detector [105] and the Triangular Detector [106] have been suggested in this context, as geometric shapes serve as significant indicators for recognizing traffic signs. TSD techniques utilizing both color segmentation and form matching are utilized in [107, 108]. TSD has become intrinsically linked with machine learning. AdaBoosting is utilized for TSD in [109]. A novel Common-Finder AdaBoost (CF. AdaBoost) algorithm is introduced in [110]. ACF and ICF detectors are documented in [111]. An accurate and efficient traffic sign identification method is proposed in [112], utilizing both AdaBoost and support vector regression for the learning of a discriminative detector. Deep learning has garnered widespread interest in recent years due to its capacity for representational learning from raw data. Consequently, CNNs have been utilized in TSD systems. A CNN-feature-based real-time TSD system utilizing a sliding window method is detailed in [108]. A TSD system employing two deep learning methodologies, specifically a fully convolutional network (FCN) for traffic sign proposal and a deep convolutional neural network (CNN) for object categorization, is introduced in [113].

2.5.1.2 Traffic Sign Recognition Methods

Traffic Sign Recognition Methods can be classified as a pattern identification challenge due to the availability of numerous known machine learning algorithms. Support Vector Machine (SVM) is one of the numerous effective models employed in [114]. In [115], SIFT and SURF descriptors trained using MLP were employed to attain elevated recognition rates. A K-D tree incorporating HOG features has been documented in [117] and RF [116]. In [118], a Radial Basis Function (RBF) neural network and a K-D tree are employed to ascertain the content of traffic signs. A sign similarity assessment with SimBoost and the fuzzy regression tree method is proposed in [119].

A classifier ensemble utilizing the Error-Correcting Output Code (ECOC) is shown in [120]. The ECOC matrix comprises a collection of ideal tree topologies utilized for the generation of the ECOC. The sparse representation-based graph embedding method utilized in TSR, as referenced in [122], integrates subspace learning with feature selection to attain superior performance. Sparse Representation Classification (SRC) [121] was proposed for face recognition. Integrated multi-modal tree structure Multi-License Plate Recognition Task Learning (MTL) is introduced in [123] to address the TSR issue. It identifies features for enhanced recognition and disseminates pertinent features for associated tasks.

The key to system performance in any recognition task is in feature representation. The development of representative and discriminative features has been the central emphasis of computer vision research. Due to their robust task-specific attributes, CNN-based techniques attained superior performance in the German Traffic Sign Recognition Benchmark (GTSRB) competition [124], organized by the International Joint Conference

on Neural Networks 2011 (IJCNN 2011). Multi-dimensional CNN presents the outputs from all stages to a classifier, resulting in superior performance compared to conventional CNN, which only provides the output of the final step to a classifier [125]. A committee of CNN and MLP that may autonomously acquire hierarchical, task-specific invariant features is proposed in [126]. Despite both technologies surpassing human performance, they are comparatively resource-intensive. Hinge loss The recognition performance was enhanced by CNN [127], which implemented a cost function often utilized in SVM.

2.6 Traffic Sign Recognition Models and Frameworks

Numerous authors have examined TSR through various methodologies. TSR problems can be categorized into three groups: color and shape-based, classic machine learning approaches, and deep learning. [128] employed a two-step process for TSR: initially categorizing signs as either triangles or circles utilizing HOG and Support Vector Machine (SVM); then, CNN was applied for image recognition. Their methodology outperformed numerous models. Nonetheless, it incurs significant computational expenses. Furthermore, the reservation signs were neglected by this method. A different hybrid approach was suggested by [129].
Decision Tree (DT) and Random Forest (RF) classifiers were employed on features derived from Histogram of Oriented Gradients (HOG) and Gray Level Co-occurrence Matrix (GLCM) for shape and texture analysis, respectively. [130] addressed the issue of occlusion by utilizing Decision Trees for color-based classification and both Decision Trees and Random Forests for texture-based classification. Their methodology significantly enhances feature extraction and representation. Nevertheless, the model may

have difficulties in managing intricate, high-variance traffic conditions. In their study, [131] observe that traffic signals possess descriptive and discriminative characteristics that can be readily identified at a lower level. They suggested a model that utilizes a straightforward feature extraction technique (HOG with feature reduction) that incurs less time expenditure and demonstrates superior performance in practical applications. Their model employs Principal Component Analysis (PCA) for feature dimensionality reduction on HOG. The decrease is essential as elevated dimensions escalate the computational expense. The study indicates that implementing feature reduction enhances the model's performance in Precision, Recall, F1-measure, and Accuracy. The proposed model's test duration was markedly decreased.

A crucial aspect of TSR is classification, which can be accomplished using various methods, including machine learning. Extensive study has concentrated on picture categorization and the several models released over the years. Various researchers and developers have endeavored to address challenges associated with TSR (Image classification), including occlusion, overfitting, class imbalance, and computing expense. In 1998, [132] proposed Convolutional Neural Networks (CNNs), which have since proven essential in the field of image classification. CNN can autonomously extract and assimilate information from images. They effectively identify multiclass images. Nonetheless, CNNs are resource-intensive, especially when processing large datasets. Furthermore, convolutional neural networks encounter difficulties with imbalanced datasets [133]. To address the issue of overfitting, [134] advocates the implementation of dropout layers. CNNs still need substantial computational resources when processing larger image collections. The issue of overfitting has been addressed by the use of an ensemble model

that integrates multiple decision trees to create a random forest. The results of a model utilizing random forest and HOG feature extraction, as given by [135], demonstrated enhanced accuracy relative to SVMs and CNNs. However, the computational expense was excessively high primarily due to the intricacy of the forest's decision-making process. The model exhibited superior performance on imbalanced data but encountered difficulties with high-dimensional image datasets. The Approximate Nearest Neighbor (ANN) model introduced by [136] enhances the K-Nearest Neighbors (KNN) algorithm. Both KNN and ANN have elevated computing costs during inference, as they calculate distances to all other data points. While ANN diminished the computational requirements, the overall computational power remains elevated.

Recurrent Neural Networks (RNNs) exhibit strong performance on temporal data. Recurrent Neural Networks (RNNs) utilize Long Short-Term Memory (LSTM) for image classification by interpreting visual input as a sequential task. The primary difficulty with RNNs is their constrained scalability and elevated training expenses. [138], [139]. RNN classification is not executed at the pixel level with certain accuracy and precision. Fully Convolutional Networks (FCNs), as introduced by [140], address pixel-level classification challenges, including semantic segmentation. Experimental studies indicate that Fully Convolutional Networks (FCNs) outperform Convolutional Neural Networks (CNNs) in image categorization tasks. Like CNNs, ANNs, and RNNs, FCNs necessitate substantial computer resources for processing extensive datasets. Moreover, their efficacy may diminish in the presence of noisy data. A significant obstacle in picture classification is the vanishing gradient problem. This issue escalates with more complex networks. ResNet was developed by [138] to address the vanishing gradient issue in deep networks. It introduced

skip connections, facilitating the seamless flow of gradients through deep systems. Models based on ResNet have attained considerable success in classification problems. Nonetheless, they still need considerable processing resources for training, and attaining optimal accuracy on extremely heterogeneous datasets might be difficult. DenseNet enhances ResNet by utilizing fewer parameters and facilitating more efficient feature reuse through feed-forward connections among all layers. EfficientNet optimally enhances the network's depth, width, and resolution by a compound scaling methodology. It significantly reduces the amount of parameters while delivering exceptional performance. Nonetheless, the model necessitates meticulous calibration of scaling parameters, and altering this balance can incur significant computing expenses [141], [142].

Despite the transformative impact of deep learning models such as CNNs, ResNet, and EfficientNet on image categorization, they possess certain limitations. For instance, they generally require substantial labeled data for training, which may be scarce in certain real-world applications. Moreover, computationally intensive models such as DenseNet and ResNet may be unsuitable for deployment on edge devices or in real-time applications where memory and speed are paramount considerations. Ultimately, many of these models exhibit a deficiency in interpretability, particularly in convolutional neural networks and deep learning architectures, complicating the comprehension of decision-making processes, which is vital in critical domains such as healthcare and autonomous driving. Models utilizing handmade features (e.g., SVMs with HOG) generally exhibit inferior performance compared to end-to-end deep learning methodologies, particularly on intricate datasets where feature extraction may fail to encompass all essential data attributes. Table 2.1 below summarizes frequently utilized models along with their accuracy.

Table 2.9 summary of commonly used TSR models and their accuracy

Author	Algorithm	Dataset	Accuracy Percentage
Kerim and Efe (2021) [143]	ANN	GTSRB	95
Soni et al., (2019) [144]	LBP, HOG, PCA, SVM	TSRD (Chinese)	84.44
Namyang and Phimoltares (2020) [145]	HOG, CLD, SVM, Random Forest	Self-collected (Thai)	93.98
Li et al., (2022) [131]	Color Histogram, HOG, PCA	GTSRB	99.99
Madani and Yusof (2018) [146]	Border Color, Shape, Pictogram, SVM	GTSRB	98.23
Sapijaszko et al., (2019) [147]	DWT, DCT, MLP	BTSD, GTSRB, TSRD	96,95.7,94.9
Aziz and Youssef (2018) [148]	HOG, CLBP, Gabor, ELM	GTSRB, BTSC	99.10, 98.30
Weng and Chiu (2018) [149]	CCL, HOG, SVM	GTSRB	90.85
Wang (2022) [150]	LR, MLP, SVM	GTSRB	97.5,98.88,95.51

Even though machine learning models for image classification have advanced significantly, more models that are computationally effective, able to generalize to a variety of noisy data, and interpretable for high-stakes applications are still required. Some of these

difficulties might be lessened by combining several strategies and semi-supervised learning.

2.7 Theoretical framework

Multiple theories were applied in the study targeting cognitive psychology, computer vision, pattern recognition and machine learning. The theories explain how TSR operates and how they can be improved.

2.7.1 Feature Analysis Theory

The two stages of the feature Analysis theory that was developed in 1979 by Treisman and Gelade is applied to this study [151]. The theory notes that object recognition is done in a hierarchical manner with the first step being pre-attentive stage. In this step, visual features including lines, angles, edges, curves and color are extracted rapidly and automatically from the input image. The step doesn't require attentions. After extracting these features, the second step is responsible for integrating these features to a coherent perceptual that performs object identification [151]. Notably, the theory notes that for identification to occur, then the second step must have seen a copy of similar feature. The theory was handy in the study because the experiment applied the two stages in the development of the model. First, the study modified Histogram of Oriented gradient and Local Directional Pattern and developed Local Directional Histogram of Oriented Gradient (LD-HOG) descriptor. The descriptor was then used to extract features from the GTSRB. Step two the study trained and tested the performance of LD-HOG on three models; Support Vector Machine, Decision Tree and Random Forest classifier. The study selected this theory because it explained how human beings were able to recognize objects even under varying conditions

such as rotations, lighting, occlusion and partial visibility which have been major problems of computer vison tasks like TSR.

2.7.2 Pattern Recognition Theory

Propose by Duda et al in 2001, the pattern recognition theory deals with classifying data based on statistical and structural information [152]. TS pixels features including texture were first converted to a binary number that was used for statistical calculation. The theory proposes a three-stage pipeline, preprocessing, feature extraction and classification. The study applied structural information like shape of the edges in trying to solve TSR problems including faded colors, and rotation. In order to overcome this challenges that faced traditional descriptors like HOG, the study applied the theory by considering all the eight neighbors of a pixel when computing the digital print of the image. The eight-pixel enabled better structural calculation hence improved performance of the TSR when exposed to varying condition. The study applied this theory in two ways; first when developing LD-HOG the study considered all the 8 neighbors as suggested in the theory. Secondly, the model was developed with the three stages proposed in this theory (Preprocessing, feature extraction and classification).

2.7.3 Machine Learning Theory

The classification of TSR to the 43 classes of the GTSRB applied the Machine Learning Theory. The theory helps in decision making of the stages applied to the classifier selected including the type of learning, bias-variance trade-off, generalization and validation [153]. Supervised learning was used because the GTSRB dataset was a labelled dataset. Feature were mapped to classes. Additionally, classifiers Support Vector Machines (SVM), Decision Trees (DT), and Random Forests (RF) are used to predict traffic sign classes

based on feature vectors. The three classifiers have a limitation of requiring high quality of input features. Weak or noisy features degrade classifier performance. Therefore, the study addresses this theoretical gap by designing LD-HOG, passes higher quality and more consistent features to the machine learning models. Additionally, Stratified K-Fold Cross-Validation was applied to ensure generalization across unseen data. Theoretical justifications for classifier selection and optimization are based on their ability to manage feature space dimensionality, interpretability, and performance on real-world TSR datasets [153].

The integration of Feature Analysis Theory with Pattern Recognition and Machine Learning Theories supports the central aim of this study: to develop a robust and efficient TSR system. The theoretical framework informs the design of the LD-HOG descriptor, which mimics human perception by capturing essential gradient and directional features. These features are then processed using machine learning algorithms to achieve real-time recognition.

In summary, this research applies:

i). Feature Analysis Theory to guide how visual features are extracted and combined;

ii). Pattern Recognition Theory to structure the stages of TSR;

iii). Machine Learning Theory to inform classifier selection and optimization;

2.8 Knowledge Gap

From discussed literature, the following are the gaps that the study has identified. First, many existing Traffic Sign Recognition models have been developed and evaluated on datasets with clear weather conditions. However, the performance of these models may

significantly degrade under adverse weather conditions, such as rain, snow, and fog. Also, while some Traffic Sign datasets are diverse and include a wide range of Traffic Signs, many existing models may not generalize well to unseen Traffic Signs. Future research could focus on developing models that are more robust and generalizable to diverse Traffic Signs, including those with varying lighting, road textures, and traffic patterns.

Additionally, many machine learning models for Traffic Sign Recognition are considered black box models, meaning that their decision-making process is not easily interpretable or explainable. As Traffic Sign analysis has important safety implications, it is important to develop models that can provide insights into their decision-making process. Another gap is that Traffic Sign analysis is often performed in real-time, such as for autonomous driving applications. Therefore, it is important to develop models that can provide accurate predictions in real-time. There is need to focus on developing models that can achieve high accuracy while maintaining fast inference times. Lastly, while there are several Traffic Sign datasets available for training and evaluation, the amount of labeled data is still limited compared to other computer vision tasks. Future research could focus on developing techniques for efficient data augmentation, transfer learning, or semi-supervised learning to address the issue of limited data availability.

2.9 Conceptual Framework

The study purpose is to develop a machine learning model for Traffic Sign Recognition. Notably, a robust feature extraction technique that can improve the performance of Traffic Sign Recognition is the main contribution. Figure 2.14 provides a proposed conceptual framework, having image features as the independent variable and Traffic Sign

Recognition as the dependent variable. Illumination, occlusion, orientation and noise are moderation variables.

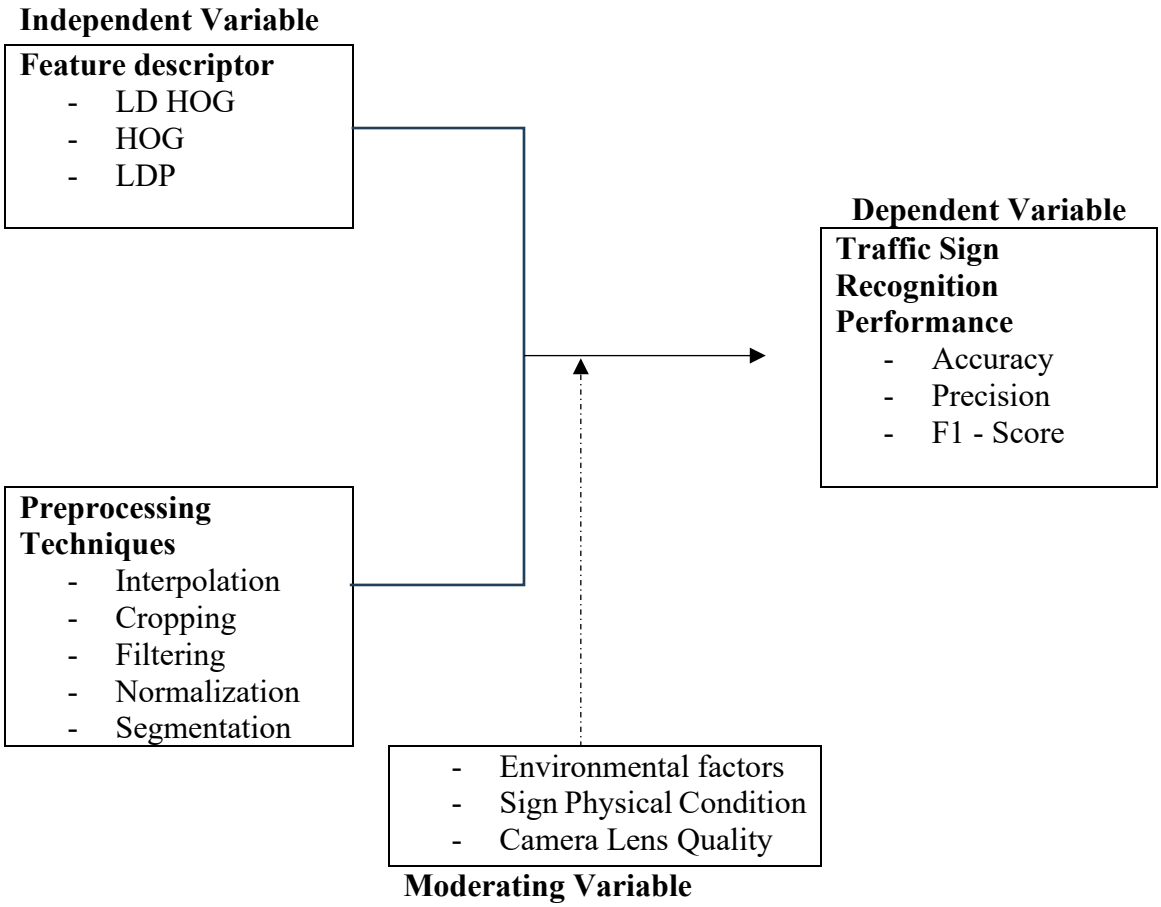


Figure 2.17 Conceptual framework

2.10 Chapter Summary

This chapter conducted a thorough evaluation of pertinent literature in order to build the theoretical and empirical framework for the study. The review was organized to methodically address the goals of the study, starting with basic ideas and moving on to the particular field of traffic sign recognition (TSR). An introduction to image classification and artificial neural networks (ANNs) was given at the beginning of the chapter, covering important topics such as backpropagation, activation functions, and the vanishing gradient

problem. The fundamental ideas of automated pattern recognition were thus established. Convolutional Neural Networks (CNNs), the predominant architecture in contemporary computer vision, received a substantial amount of attention. The chapter provided a comprehensive grasp of how deep learning models extract and analyze hierarchical characteristics from images by breaking down the fundamental CNN components, such as convolution kernels, pooling layers, and more complex modules like Inception and Residual Learning.

After thereafter, the emphasis shifted to methods for classifying traffic signs. We looked closely at the conventional pipeline of feature extraction (HOG, SIFT, LBP, Gabor Filters, and Bag of Words) and preprocessing (interpolation, cropping, color space conversion, filtering, normalization, and segmentation). This created the framework for comprehending both conventional and deep learning-based TSR techniques.

The performance of several methods ranging from conventional machine learning (SVM, RF) to advanced deep learning models (AlexNet, GoogleNet, ResNet) was compared in the evaluation of Traffic Sign Recognition Models and Frameworks. Particularly with regard to computing cost, robustness, and generalizability, this comparison analysis assisted in identifying the advantages and disadvantages of the current approaches. Three pillars served as the foundation for the theoretical framework:

i). Feature Analysis Theory, which served as a guide for creating a feature descriptor that hierarchically extracts and integrates low-level data (edges, gradients) to replicate human visual perception.

ii). Pattern Recognition Theory, which emphasized the significance of local structural information and guided the structured pipeline of preprocessing, feature extraction, and classification.

iii). Machine Learning Theory, which offered the guidelines for choosing, refining, and validating classifiers and guaranteeing that the model could extrapolate from training data to unobserved instances.

The chapter ended by highlighting important knowledge gaps in the body of current TSR research, including issues with generalization, poor performance under challenging circumstances, limited real-time efficiency, and lack of model interpretability. The purpose of this study is to develop a more reliable, effective, and comprehensible traffic sign recognition system, and these gaps clearly demonstrate the requirement for the unique feature descriptor, LD-HOG. These components are graphically synthesized in the Conceptual Framework, which also describes the link between the independent variable (the proposed LD-HOG descriptor), the dependent variable (the TSR performance), and moderating environmental factors.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Overview

The research methodology is organized following the Saunders research onion. All the six stages; research philosophy, research approach, research strategies, research choices, time horizons, and techniques and procedures are discussed in this chapter [154].

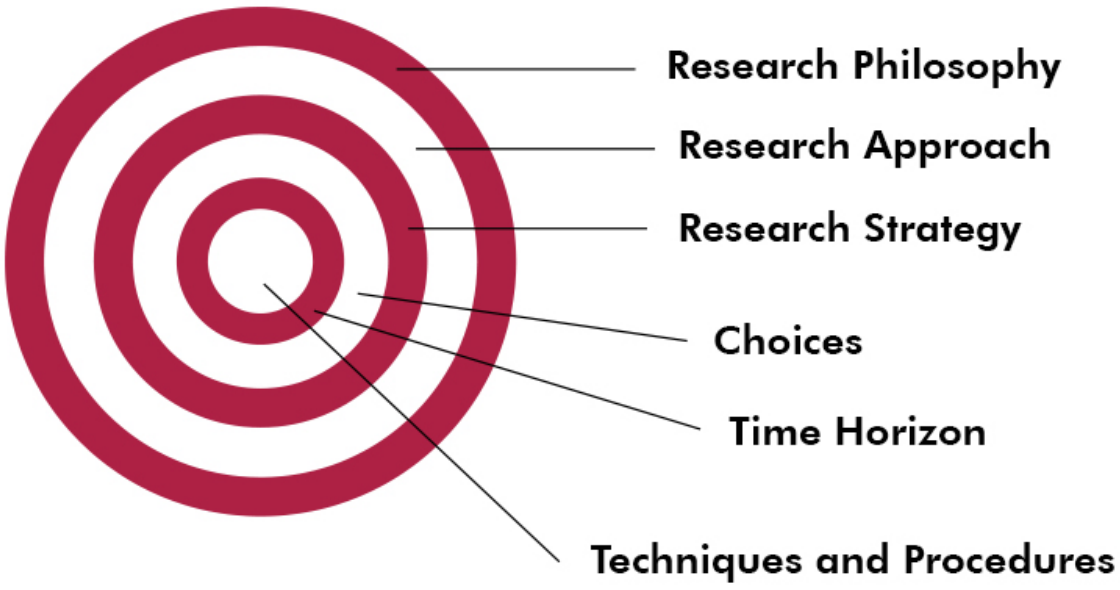


Figure 3.1: Saunders onion [154]

3.2 Research Design

A mixed approach design was used for the study. Systematic Literature Review was used to achieve objective one while simulation and experimental research design were used for objective 3 and 4. Specifically, Systematic literature review was used in the development of LD-HOG feature extractor, while simulation was used in the development, testing and

validation of the TSR model. Experiments were done during model development. The three stages preprocessing, feature extractor and classification were experiment on and optimal values were selected for the model. Lastly, validation experiments were carried out using LD-HOG feature extractor.

3.3 Research Philosophy

The convictions, assumptions, and principles that inform a researcher's methodology are termed their research philosophy. Positivism, interpretivism, and critical realism are identified as the three principal research ideologies. Positivism, a research ideology, prioritizes the application of scientific methods to generate objective, quantifiable data. Positivists assert that a singular objective reality exists, which can be quantified and analyzed through empirical research. Researchers should uphold objectivity and detachment during the study process, ensuring that research remains value-free. The convictions, assumptions, and principles that inform a researcher's methodology are termed their research philosophy. [154] designates positivism, interpretivism, and critical realism as the three principal research paradigms. Positivism, a research ideology, prioritizes the application of scientific methods to generate objective, quantifiable data. Positivists assert that a singular objective reality exists, which can be quantified and analyzed through empirical research. [154] assert that the researcher must uphold impartiality and detachment during the study process, and that research ought to be devoid of values.

Individuals and social institutions engage to construct social reality, and research should focus on understanding the essential mechanisms and processes that affect social phenomena. This study adopted a positivist attitude due to its foundation in empirical data

and proven theories, specifically feature analysis, pattern recognition, and machine learning. Moreover, a traffic sign is a picture, and machine learning methodologies can be employed to extract and categorize its components. To identify and categorize an image, the study will employ the two-level technique proposed by feature analysis theory [151].

3.3 Research Approach

Inductive and deductive approaches are two research approaches that are commonly used in research. The inductive approach is a bottom-up approach where the researcher starts with specific observations and data, and then builds up generalizations or theories based on those observations. The researcher begins by collecting data and then uses that data to generate a hypothesis or theory. The researcher may use qualitative research methods such as interviews or observations to collect the data. Inductive research is often used when there is little existing research or theory on a topic, and the researcher aims to generate new knowledge or theory. By contrast, the deductive technique is a top-down method in which the researcher begins with a hypothesis or theory and then uses data to test it. First, the researcher formulates a hypothesis or theory, which is subsequently tested by gathering evidence. Surveys and experiments are examples of quantitative research procedures that the researcher may employ to gather data [154–155].

The study adopted deductive approaches because of anchoring the study to existing theories on feature analysis, pattern recognition and machine learning [151],[152] and [153]. Additionally, LD-HOG and the traffic sign recognition machine learning model was developed using simulation and experiments and later tested on existing dataset.

3.4 Research Strategy

A research study's overarching plan or design is referred to as its research strategy. It describes the procedures the researcher will follow in order to answer their study questions or goals. Among the several research approaches is the experimental approach, which entails adjusting one or more factors to ascertain how they affect a result. Testing causal links between variables is one of its common uses. Using interviews or structured questionnaires, a survey research technique gathers information from a sizable sample of individuals. It is frequently employed to collect data regarding beliefs, attitudes, and actions. In order to comprehend a specific phenomenon, a case study research technique entails a thorough examination of one or a limited number of cases. Exploratory research frequently uses it.

In order to solve real-world problems or concerns, academics and practitioners work together as part of the action research strategy. In applied research, it is frequently utilized. In order to create a theory or framework that is based on the evidence, a grounded theory research strategy include the methodical collecting and analysis of data. Exploratory research frequently uses it. Lastly, to provide a more thorough understanding of a certain occurrence, a mixed methodologies research plan combines two or more research strategies, such as qualitative and quantitative research [154], [155].

The study applied a mixed research design with Systematic Literature Review used to achieve objective 1 and 2, and experimental research strategy for objective 3 and 4. This selection ensures that the findings from literature review and experimental research can be analyzed against the expected results. The experiments were virtually for testing and comparing LD-HOG performance with other feature extractors specifically HOG and LDP.

3.5 Time Horizon

A research study's time horizon is the amount of time it is conducted. In research, there are three primary categories of temporal horizons: retrospective, longitudinal, and cross-sectional.

Cross-sectional studies are conducted at a single point in time. They collect data on a sample of participants or observations at a specific moment in time, and the data is then analyzed to draw conclusions about the population or phenomenon of interest. Longitudinal studies are conducted over an extended period of time, with multiple data collection points. They track changes or trends in a sample of participants or observations over time, and the data is then analyzed to draw conclusions about the population or phenomenon of interest. Retrospective studies are conducted after an event or period of time has passed. They collect data on past events or experiences through interviews, surveys, or archival records, and the data is then analyzed to draw conclusions about the population or phenomenon of interest [155].

The study applied both cross-section and retrospective. Retrospective, the study extracted features from the GTSRB dataset with 51,840 images that was collected in the past. Nonetheless, the study was conducted at a single point in time.

3.6 Data Source

Data source refers to the origin or location from which research data is collected. There are several types of data sources, including primary, secondary and tertiary data sources. Primary data source involves collecting new data directly from the source, such as through surveys, interviews, experiments, or observations. This type of data is original and specific

to the research question at hand. Secondary data sources involve using existing data that has been collected by others, such as government agencies, research institutions, or public archives. This type of data can be used to supplement primary data, provide historical context, or compare findings across studies: Tertiary data sources involve using data that has been compiled or synthesized by others, such as meta-analyses, literature reviews, or data repositories. This type of data can be used to identify trends, patterns, or knowledge gaps in a particular field of study [154 - 155]. This research used a secondary data source, specifically the German Traffic Sign Recognition Benchmark (GTSRB) dataset. The GTSRB dataset contained 51,840 images of traffic signs, divided into 43 classes, and is widely used for training and evaluating traffic sign recognition systems. The images vary in resolution, lighting, and background conditions. For the purposes of this study, the dataset was split into two parts; 80% (39,309) of the data was used for training and 20% (12,631) was used for testing the model [156].

3.7 Experiment setup

Experimental and simulation research methods were used in the study. Preprocessing, feature extraction, and classification were the three phases of the simulation and experiments, which were conducted in accordance with the three study theories. Both the experiment protocol and the research tools and equipment are covered in this section.

3.7.1 Experiment Environment

A white bed was set with the following conditions to enable fair testing and comparison of LD-HOG, HOG and LDP.

i). Python 3.8 with the OpenCV, NumPy, and scikit-learn libraries

ii). Windows 10

iii). Processor: Intel Core i7 (10th Gen)

iv). RAM: 16 GB

v). GPU: NVIDIA GTX 1650

vi). IDE: VS Code

Python was selected because of its performance in terms of speed. Additionally, python has many in built libraries like Open CV NumPy and scikit-learn that plays a curial role in training and testing of the model part of the code is a machine language. [157]

3.7.2 Experiment procedure

The main objective of the study was to develop a machine learning model for traffic sign recognition. The development of this model had several stages including selection of a dataset, data preprocessing, feature extraction, model training, testing, and performance evaluation. The following subsections outline the key components of the experimental procedure.

3.7.2.1 Dataset

The German Traffic Sign Recognition Benchmark (GTSRB) dataset was used. It has 43 classes with a total of 51,840 images. GTSRB was selected because it has images with different characteristics. For instance, images in the same class vary in terms of resolution, illumination, and background, providing a realistic setting for model training and evaluation.

3.7.2.2 Data Splitting

The GTSRB was used both for training and testing. In order to see how the model would perform in real world situation, there was need to test the model with images that were not involved in training. Therefore, the dataset was split with 80 percent being training images and 20 percent testing images. Notably, the dataset had different number of images across the 43 classes. Stratified splitting was done in order to preserve the class distribution in the testing and training images.

3.7.2.3. Data Preprocessing

As noted in section 3.7.2.1 and as explained by the pattern recognition theory, preprocessing if the first step of Traffic Sign Recognition and image classification at large. It is important to clean and enhance features of the input images before passing them to the feature extractor. Five preprocessing techniques were performed on the images. Experiments were carried out to identify the best method for each technique on the GTSRB that would be used in the creation of the model. These five techniques are:

i). Image interpolation

ii). Image cropping

iii). Image filtering

iv). Image normalization

v). Image segmentation

3.7.2.4. Feature Extraction

The study designed and developed LD-HOG feature extractor that considered all the 8 neighbors. Simulation and experiments were done to compare LD-HOG performance of

the GTSRB and two other feature extractors. LD-HOG is an enhancement of HOG that applies concepts from LDP. In order to simulate and come up with LD-HOG, the following steps were taken:

i). In addition to the normal HOG vertical and horizontal gradients, left and right diagonals at 45^{o} and 135^{o} were calculated. Vertical gradient is 90^{o} while horizontal gradient is 180^{o}

ii). In order to combine this the left, right, horizontal and vertical gradients, three pooling methods were considered and experimented on. This pooling method are Maximum pooling (MAX), Average pooling (AVG) and Minimum pooling (MIN). Maximum pooling was the best in retaining the most significant texture when applied to GTSRB and therefore selected for final model development.

iii). After pooling the next step was to convert the gradients to orientation bins. Each gradient was converted and then combined together to form one final bin with nine bins separated at 20^{o}

iv). To reduce noise such as lighting and illumination, normalization was done. L2-norm was applied to the resulting feature vector from iii)

v). Finally, the feature vector was constructed. The LD-HOG descriptor for an image was constructed by merging all the normalized histogram from each image block.

For purposes of testing the performance of LD-HOG, the other two feature extractors HOG and LDP were also constructed and applied to the GTSRB dataset (appendix 2,3 and 4). Notably, the preprocessing of the images from the GTSRB was same and the development environment as well. Also, the same images were used and all the three-feature extractor used the same function in loading of Images (appendix 1).

3.7.2.5. Classification

The third part of the pattern recognition and the second part of the feature analysis is classification of extracted features. This process is done by a machine learning algorithm for classification. In TSR problem, we have 43 classes making it a multiclass classification. The study applied three machine learning classifiers in the training of the model on the features extracted in section 3.7.2.4. These classifiers were Support Vector Machine, Random Forest and Decision Tree. Notably, the three classifiers trained on the 80 percent of the images from the GTSRB and were later tested on 20 percent. Each classifier had an output of three models, one for LD-HOG, another for HOG and the last one for LDP (Appendix 5,6 and 7).

After training the models for each classifier, performance was evaluated using precision, recall and F1 score. Comparison on the three-feature extractor was done based on this three performance evaluation metrics. Validation of the models that were trained using LD-HOG was done in two steps; first cross validation was done using stratified k-fold validation with the value of k being 5. Secondly, the confusion matrix of the models was printed to support and validate the performance of the model.

3.8 Ethical Considerations

In order to meet important imperatives including data privacy, algorithmic bias, transparency, and safety, this research was carried out with a strong dedication to ethical AI principles. While proactive class imbalance mitigation via stratified sampling and class-weighted classifiers was used to assure fairness and avoid model bias toward overrepresented sign classes, privacy concerns were avoided by using the publicly available, non-identifiable GTSRB dataset. Additionally, the selective selection of

interpretable models such as Decision Trees and handcrafted features (LD-HOG) outperformed "black-box" alternatives in terms of transparency and accountability, which is essential for debugging and building trust in safety-critical systems. Finally, in recognition of the significant safety implications of possible deployment in autonomous systems, the model's design specifically placed a high priority on computational efficiency and robustness to allow dependable, real-time performance.

The research proposal was submitted for approval by Directorate of Postgraduate Studies (DPS) and senate of Masinde Muliro University of Science and Technology. NACOSTI approved conducting of research and a copy of the research permit is attached.

3.9 Chapter Summary

This chapter described the methodical approach used, organized in accordance with the Saunders study Onion framework, to accomplish the study objectives. Based on well-established theories of feature analysis, pattern recognition, and machine learning, the study used a positivist ideology and a deductive approach. A mixed-method approach was used: Objectives 1 and 2 (analyzing current methodologies and finding gaps) were satisfied by a Systematic Literature Review, while Objectives 3 and 4 (creating and evaluating the LD-HOG model) were accomplished by an Experimental Research Design. The secondary GTSRB dataset, which was divided 80/20 using stratified sampling to maintain class distribution, was used for the cross-sectional time horizon.

Data preprocessing (interpolation, cropping, filtering, normalization, segmentation), feature extraction using the novel LD-HOG descriptor (which improves HOG by incorporating 8-directional gradients and MAX pooling), and classification using SVM, Random Forest, and Decision Tree classifiers comprised the experimental process, which

followed a clear pipeline. Python was used to control the environment, together with certain hardware and libraries. Precision, recall, F1-score, and stratified 5-fold cross-validation were used to thoroughly assess the model's performance, and ethical considerations for AI which address data privacy, bias, transparency, and safety were incorporated throughout the entire process.

CHAPTER FOUR

RESULTS AND DISCUSSION

This study's primary goal was to create a machine learning model for identifying traffic signs. The research findings, outcomes, and discussion of the study are presented in this chapter. Research findings are presented in this chapter using the approach covered in chapter three. Before presenting and discussing the findings in accordance with the objectives, the chapter begins with a description of the dataset that was used.

4.1 The German Traffic Sign Recognition Benchmark

The dataset was made up of 39,210 images that were used for training and 12,631 images that were used for testing giving a total of 51,841. Both the train and test were made up of 43 classes with each class having images with different ROI as well as occlusion and opacity. Figure 4.1 shows sample image of each class.

Figure 4.1 Sample image of each class

The classes had images that were used to test the main problems of sign recognition and classification i.e. occlusion, variety of colors, and color deterioration.

4.2 Analysis of Traffic sign recognition techniques

"To analyze existing traffic sign recognition techniques for their strengths, weaknesses, and performance," was the first goal, and it was accomplished by conducting a comprehensive literature review. The results are summarized below based on the main phases of a TSR pipeline, emphasizing important performance trade-offs and prevailing patterns.

4.2.1 Critical Analysis of the Preprocessing Stage

Preprocessing is a fundamental stage where strategic decisions have a direct impact on downstream performance, according to the literature. A uniform trade-off between feature preservation/enhancement and computing efficiency was found by the analysis. Specifically, the following are the findings of different preprocessing process.

i). The quickest algorithms, such as Nearest-Neighbor (O(n)), create blocky artifacts that deteriorate edge definition, which is essential for sign identification. On the other hand, Lanczos resampling ($O(n*k^2)$) is less appropriate for real-time systems due to its considerably higher computational cost, despite its exceptional ability to preserve high-frequency features and sharp edges. For TSR, Lanczos resampling was selected because of the importance of reserving edges of Traffic Sign.

ii). From the study, RGB confuses color and brightness information, even though it is computationally efficient (O(1)) for extraction. For shape-based descriptors like HOG, grayscale conversion (O(n)) is quite effective and adequate; nevertheless, it represents all color information as pixels, which is a crucial component for traffic signs. Despite being more computationally demanding to convert to, HSV and Y'CbCr distinguish luminance from chrominance, offering increased resilience to variations in illumination.

iii). Linear filters ($O(n^2)$) blur edges but are effective for overall smoothing. Although they require more computing, non-linear filters (like the median) are better at preserving edges while eliminating noise. Although deep learning-based filters are quite flexible, they are too complicated for applications with limited resources and require a lot of data.

iv). Contrast stretching (O(n)) is a quick and popular option for normalization, while spatial normalization ($O(n^2)$), while effective at fixing distortions, is computationally costly.

The study notes that the selection of a preprocessing technique required compromise; there is no one ideal preparation pipeline. Techniques that maintain important properties (color, edges) without requiring excessive processing cost must be given priority in a reliable TSR system for real-world scenarios. Table 4.1 and figure 4.1 summaries findings of the preprocessing techniques.

Table 4.1: Comparative Analysis of Preprocessing Techniques

Technique	**Best Use Case**	**Computational Complexity**	**Key Advantage**	**Key Limitation**
Bilinear Interpolation	General-purpose resizing	O(n)	Fast processing	Blurs edges
Lanczos Resampling	High-detail preservation	$O(n*k^2)$	Sharp edge preservation	Computationally expensive
Grayscale Conversion	Shape-based detection	O(n)	Reduces complexity	Loses color information
HSV Conversion	Illumination robustness	O(1)	Color-brightness separation	Additional computation
Median Filtering	Noise removal	$O(n^2)$	Preserves edges	Moderate computation
Contrast Stretching	Normalization	O(n)	Simple and fast	Limited dynamic range improvement

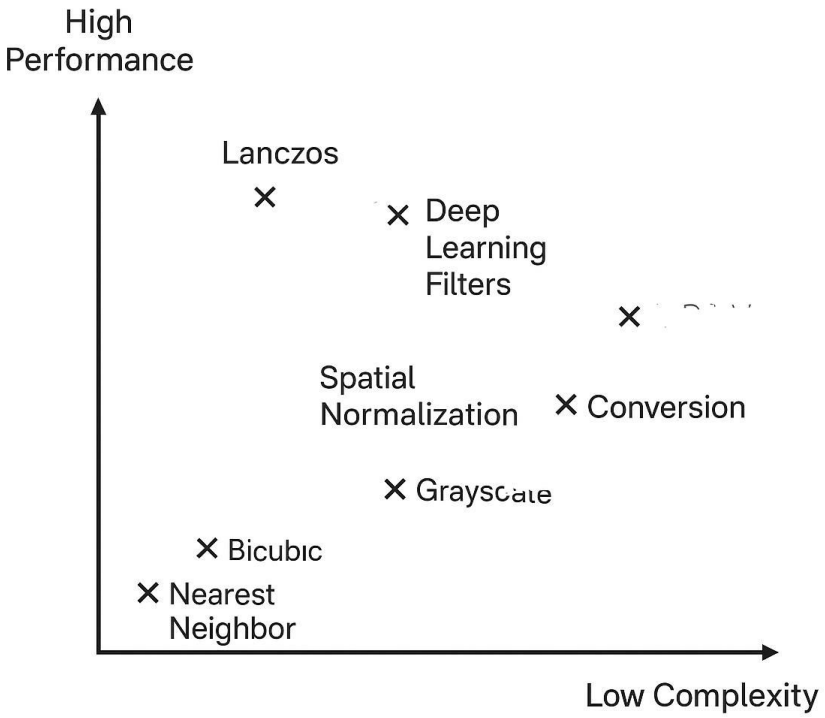


Figure 4.1 Comparative Analysis of Preprocessing Techniques

4.2.2 Comparative Analysis of Feature Extraction and Classification Techniques

The interaction between feature extraction and classifier selection has a significant impact on a TSR system's performance. According to the analysis, there is a noticeable trade-off in performance and efficiency between end-to-end deep learning and conventional machine learning pipelines. The handcrafted feature pipeline shows that high performance may be achieved without the overhead of deep learning. For example, from literature:

i). When combined with classifiers such as SVM, HOG-based models routinely attain excellent accuracy (98.23% [146], 99.99% [131] on GTSRB). Their ability to efficiently capture shape and edge information (O(n)) is their strongest suit; nonetheless, they are still susceptible to rotation and clutter.

ii). Combining several feature types can improve robustness, as demonstrated by hybrid feature models. For instance, [148] used an Extreme Learning Machine (ELM) for classification and combined HOG with Completed LBP (CLBP) and Gabor filters to achieve 99.10% accuracy on GTSRB. This demonstrates that, despite increasing feature dimensionality, combining shape (HOG) and texture (LBP, Gabor) information may produce excellent results.

iii). According to [131], methods such as Principal Component Analysis (PCA) are crucial for controlling the computational cost of high-dimensional feature spaces, which directly enhances test time and performance in conventional models.

Literature review shows that deep learning models have outperformed handcrafted feature extractors but at high computational cost. Expensive but Excellent Performance

CNN-based methods have won competitions such as the GTSRB benchmark, setting the standard for performance [124]. Unmatched accuracy and robustness result from their capacity to learn hierarchical, invariant features on their own. This review, however, demonstrates their serious shortcomings:

i). Real-time, embedded systems cannot use models such as VGGNet and ResNet due to their huge computational footprint ($O(n^2)$) [135, 142].

ii). They need a lot of hardware for inference and very huge labeled datasets for training.

iii). For safety-sensitive applications like autonomous driving, their inability to be interpreted is a serious issue.

There is a definite dichotomy in the literature. Although they have a robustness ceiling, traditional models (such SVM with HOG) provide an appealing balance of high accuracy

(98%+), speed, and interpretability. There are significant trade-offs in terms of performance, efficiency, and usability when comparing Deep Learning (DL) and Traditional Machine Learning (ML) techniques. Depending on the quality of the feature engineering, traditional machine learning models like Support Vector Machines (SVM), Random Forests (RF), and k-Nearest Neighbors (KNN) usually attain accuracy levels between 85% and 99%. They are appropriate for low-resource environments due to their computational efficiency, which includes low to medium training times and very quick inference speeds. These models have excellent interpretability, which makes it possible to comprehend model decisions more clearly, use less data, and run efficiently on CPU-optimized platforms.

On the other hand, Convolutional Neural Networks (CNNs) and other Deep Learning models can achieve slightly greater accuracies (92–99.5%), especially when working with huge datasets. However, this comes at the expense of a very long training period, a medium to poor inference performance, and significant data and GPU needs. Additionally, they have very poor interpretability, which makes explainability difficult. Therefore, even while DL is better at accuracy and automation, classical machine learning is still better at efficiency, transparency, and accessibility, particularly in environments with limited resources. This clearly defines a trade-off between efficiency and performance that determines which model is best. Table 4.2 compares Traditional ML and Deep Learning models performance from literature review.

Table 4.2: Model Architecture Comparison

Characteristic	Traditional ML	Deep Learning
Accuracy	85-99%	92-99.5%
Training Time	Low-Medium	Very High
Inference Speed	Very Fast	Medium-Slow
Data Requirements	Low	Very High
Hardware Needs	CPU-optimized	GPU-required
Interpretability	High	Very Low

4.2.3 Synthesis and Identified Gaps

The methodical examination unequivocally shows that a basic compromise between two paradigms defines the TSR field:

i). CNNs dominate high-accuracy, resource-intensive models, which are frequently inappropriate for embedded, real-time systems.

ii). Effective, High-Accuracy, but Less Robust Models. These models, which rely on manually designed features and conventional classifiers, accomplish remarkable benchmark accuracy but have trouble with the full range of real-world changes, such as rotation, intricate occlusion, and intense lighting.

One important lesson from the success of hybrid feature models is that robustness can be greatly increased without resorting to a deep learning architecture by carefully combining complimentary handcrafted features.

As a result, there is a clear and legitimate need for a feature extraction technique that breaks through the conventional paradigm. To directly address its shortcomings in rotational invariance and noise robustness, a novel hybrid feature descriptor that methodically combines the advantages of current handcrafted features for example, combining the textural and finer directional sensitivity of LDP with the potent shape representation of HOG would be the ideal solution.

The creation of the novel LD-HOG descriptor in this study is directly motivated by this gap. The objective of LD-HOG is to develop a descriptor that is substantially more robust than standard HOG while maintaining its fundamental benefits of computational efficiency, low resource requirements, and interpretability, making it perfectly suited for useful, real-world TSR applications. This is in contrast to competing with massive CNNs on raw accuracy.

4.3 Chapter Summary

The goal of this study was to create a machine learning model for efficient traffic sign recognition (TSR), and this chapter provided the findings and a thorough analysis of the research. An introduction was given to the German Traffic Sign Recognition Benchmark (GTSRB) dataset, which included 51,841 photos divided into 43 classes and examined a number of factors, including illumination effects, color variation, and occlusion. The preprocessing step, which is crucial for enhancing image quality prior to feature extraction, was the main emphasis of the first section's analysis of traffic sign identification systems.

One important discovery was that no one preprocessing technique is perfect; rather, efficient TSR systems must strike a compromise between computing efficiency and feature preservation (such as edge and color details).

While simpler approaches like bilinear interpolation offered speed at the expense of accuracy, more complex techniques like Lanczos resampling and HSV color conversion offered better edge and illumination handling at greater computational costs. A basic trade-off between conventional machine learning and deep learning techniques was then shown by the chapter's comparison of feature extraction and classification techniques. Conventional models, such HOG with SVM, Random Forest, and KNN, showed excellent accuracy (85–99%), quick inference, and minimal training time, which made them perfect for situations with limited resources. On the other hand, deep learning models (such as CNNs like ResNet and VGGNet) had somewhat greater accuracies (92–99.5%), but they also lacked interpretability and necessitated a lot of processing power, big datasets, and GPU resources.

The comparative synthesis showed that deep learning approaches struggle with efficiency and explainability while deep learning approaches succeed in automation and feature learning. Traditional approaches, on the other hand, are efficient, interpretable, and accessible, albeit being somewhat less robust. A research gap was noted in the chapter: current models either put accuracy above efficiency or the other way around. This drives the creation of hybrid handmade descriptors, namely the suggested LD-HOG (Local Directional Histogram of Oriented Gradients), which combines the efficiency and shape sensitivity of HOG with the textural strength and rotational robustness of features like LDP.

In summary, Chapter Four comes to the conclusion that creating lightweight, hybrid feature descriptors can help close the gap between deep learning and classical methods, resulting in the computational efficiency and resilience required for real-world TSR systems.

CHAPTER FIVE

LOCAL DIRECTIONAL HISTOGRAM OF ORIENTED GRADIENT

5.1 Overview

This chapter discusses two feature extractors; Local Directional Pattern (LDP)and Histogram of Oriented Gradient (HOG) before proposing an improved feature extractor Local Directional Histogram of Oriented Gradient (LD-HOG)

5.2 Local Binary Patterns

Image prediction methods have made extensive use of texture features [158], [159]. An appearance (texture) description called Local Binary Patterns (LBP) may identify microstructure patterns on human skin, such as corners and edges, spots, flat areas, and lines [160]. A popular texture descriptor for image classification tasks, such as TSR, is local binary patterns. Figure 5.1 shows LBP calculation for a 3 × 3 image

region.

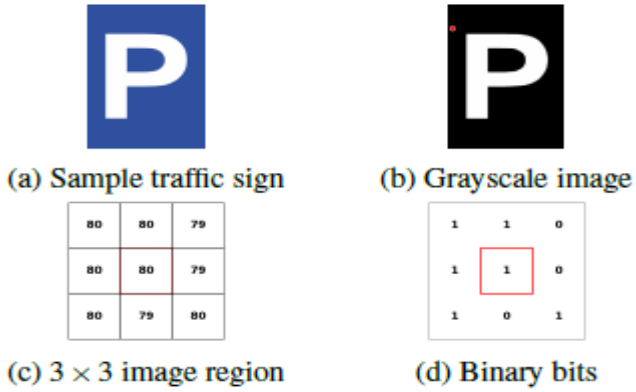


Figure 5.1 Local Binary Pattern

The Local Binary Pattern (LBP) code from figure 5.1 can be represented using equation 5.1

$$LBP = 11001011_2 \ = \ 203_{10} \tag{5.1}$$

Where base 2 is the 8-bit binary equivalent of the image while base 10 is the decimal value.

The binary 8-bit representation of the image is derived using equation 5.2 below

$$LBP_{p,r}(x_c, y_c) = \sum_{n=0}^{N-1} 2^n s(g_n - g_c) \tag{5.2}$$

and thresholding function τ is defined by equation 5.3

$$\tau(x) = \begin{cases} 1, & if\ x\ \geq 0 \\ 0, & otherwise \end{cases} \tag{5.3}$$

The function $\tau(x)$ is a threshold function that generates a binary bit for a specific pixel, where $N = p$ is the number of neighboring pixels, r is the distance of neighboring pixels from the central pixel, g_c is the central pixel's gray-value, and g_n for $n = 0,1,2, \dots N-1$ corresponds to the neighboring pixel's gray value on a circular symmetric neighborhood of distance $R > 0$. A binary number is produced by joining the eight bits. The central pixel is given the binary number's LBP code after it has been translated to decimal. Micro-pattern structures in a picture are represented by the histogram of the LBP-encoded image. Equation 5.4 defines this histogram.

$$H_i = \sum_{x,y} I(f(x,y) = i)\,, i = 0,1,2 \dots, 2^p - 1 \tag{5.4}$$

where p is the number of texture patterns that LBP operator can encode and equation 5.5 shows how to calculate I

$$I(a) = \begin{cases} 1, if\ a\ is\ TRUE \\ 0\ otherwise \end{cases} \tag{5.5}$$

[161] discovered that about 90% of all texture patterns are uniform when $p = 8\ and\ r = 1$. A uniform pattern is one that has a maximum of two bitwise transitions from 1 to 0 and vice versa. For example, the patterns 10000000, 00000000, 0100000, and 11111011 are uniform, whereas the patterns 11010000, 10100101, and 10101101 are not [160]. Micro-structures such as spots, edges, lines, or flat areas can be indicated by uniform patterns. The ability of the original LBP to encode dominant texture features that cover a wider area was constrained. In order to capture features utilizing multiple neighborhoods with different radii, the descriptor was extended [161]. A neighborhood region is defined by pixels that are uniformly spaced around a central pixel in a circular arc that is centered at the central reference pixel.

To accommodate any radius and any number of sample pixels, points that do not fall inside the pixels are bilinearly interpolated. The LBP binary code is rotated in a circular fashion until its minimum value is obtained in order to extract rotational invariant features [162]. More features in an image might be encoded by an extended LBP descriptor, but the spatial information of the features that were extracted would be lost. Each cell's histogram is joined to create the final histogram.

5.3 Local Directional Parten

Variations in illumination and picture noise can affect the local binary pattern. Jabid et al. [163] presented Local Directional Patterns (LDP), a texture descriptor that is resilient to noise and non-monotonic light fluctuations. LDP's robustness is seen in Figure 5.2.

85	32	26
53	50	10
60	38	45
$LBP = 00111000 = 56$		
$LDP = 00010011 = 19$		

81	29	32
38	58	15
65	43	47
$LBP = 00101000 = 40$		
$LDP = 00010011 = 19$		

Figure 5.2 Comparison of LBP and LDP on an original and a noisy image

The first image is an original image that has its features extracted using LBP and LDP. The same image is exposed to noise like illumination in the second image. From the image it is clear that LBP changed with introduction of noise but LDP is robust and remained with the same value.

In contrast to LBP, which compares raw pixel values, local directional pattern compares the directional responses of each neighboring pixel to compute an 8-bit code for the central pixel in a 3×3 image region. Candidate edge detectors [167] that can be used to determine an image's gradient are Prewitt [164], Kirsch [165], and Sobel [166]. Because Kirsch takes into account all eight nearby pixels [168] in a 3×3 image region, it is the most reliable of them when it comes to directional edge identification.

5.3.1 Kirsch edge detector

It is a first-order derivative edge detector that uses a series of filters (masks) to convolve each 3\times3 region in order to determine the gradient of an image. In equation 5.6, Kirsch defined a non-linear edge detector [167].

$$P(x, y) = \max\{1, \max_{k=0}^{7}[|5S_k - 3T_k|]\} \quad (5.6)$$

Where,

$$S_k = P_k + P_{k+1} + P_{k+2} \quad (5.7)$$

and

$$T_k = P_{k+3} + P_{k+4} + P_{k+5} + P_{k+6} + P_{k+7} \quad (5.8)$$

where $P_k \sim [k = 0,1,2 \ldots,7]$ are eight neighboring pixels of $P(x, y)$ shown in Figure 5.3, and an in k_a is assessed as $a = a \,\%\, 8$. Convolution of the 3×3 image region with the corresponding mask B_k, as seen in Figure 5.4, yields the image gradient in a given direction.

A_3	A_2	A_1
A_4	$A_{(x, y)}$	A_0
A_5	A_6	A_7

(a) 8 neighbors of pixel $P_{(x, y)}$

B3	B_2	B_1
B_4		B_0
B_5	B_6	B_7

(b) 8 directional Krisch mask positions

Figure 5.3 (a) Pixel A's eight neighbors (x, y) and (b) correspond to the appropriate Krisch Mask positions.

$$\begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix}$$
(a) East B0

$$\begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix}$$
(b) North East B1

$$\begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$$
(c) North B2

$$\begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$$
(d) North West B3

$$\begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix}$$
(e) West B4

$$\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix}$$
(f) South West B5

$$\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix}$$
(g) South B6

$$\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix}$$
(h) South East B7

Figure 5.4 Krisch edge response mask in eight directions

There are eight directional response values for every central pixel, X_c. When a corner or edge is present, the absolute values are high. Finding k meaningful replies, setting their associated bit value to 1, and assigning 0 to the remaining 8–k bits are the goals of LDP. After being translated to decimal, the resulting binary string is given the LDP code for X_c. To create an LDP-encoded image, the procedure is carried out for each pixel in the image. The picture encoding procedure employing the LDP operator is depicted in Figure 5.5.

85	52	26	M_3	M_2	M_1	313	97	503	0	0	1
53	50	10	M_4		M_0	537		399	1		1
60	38	45	M_5	M_6	M_7	161	97	303	0	0	0
(a)			(b)			(c)			(d)		

LDP = $00010011_2 = 19_{10}$

Figure 5.5: Image region and LDP operator encoding process for an image with k=3 (a). Figure 5.4 shows the Kirsch masks in (b). The result of convolving each pixel in (a) with eight Kirsch masks is shown in (c). (d) Select the top k=3 important answers, set the relevant bit to 1, and set the remaining bits to 0.

Starting from B_0, the LDP code is shown in equation 5.9

$$LDP = 11001000_2 = 19_{10} \tag{5.9}$$

LDP code shown is derived using equation 5.10 and 5.11

$$LDP_k = \sum_{j=0}^{7} b\left(\left(m_j - m_k\right) \times 2^j\right) \tag{5.10}$$

$$b(a) = \begin{cases} 1, if\ a \geq 0 \\ 0, otherwise \end{cases} \tag{5.11}$$

where $B_j$is the response of Kirsch mask B_j, and B_k is the k^{th} significant response. In an LDP-encoded image, the local directed pattern operator produces C_k^8 unique patterns.

Equations 5.12 and 5.13 employ the histogram $H(i)$ with C_k^8 bins to describe the input image of size $M \times N$.

$$H(i) = \sum_{m=0}^{M} \sum_{n=0}^{N} f(LDP_k(m,n), i) \tag{5.12}$$

$$f(p,i) = \begin{cases} 1 \; if \; p = i \\ 0 \; otherwise \end{cases} \tag{5.13}$$

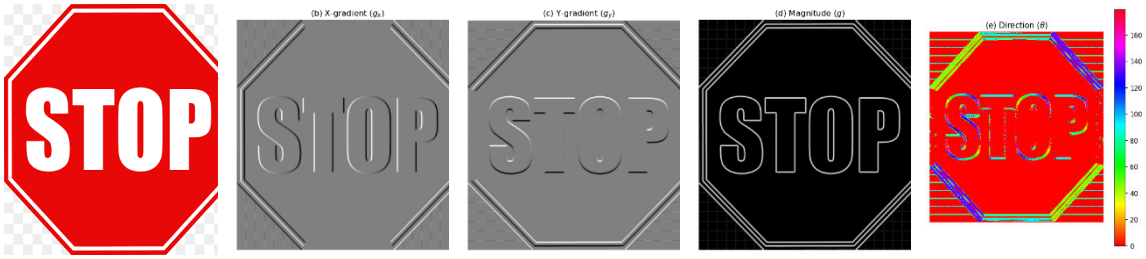
where $f(p,i)$ is a logical function that checks whether the LDP code of the LDP-encoded picture at location $p(m,n)$ is equal to the current LDP pattern i for all i in the range $0 \leq i \leq C_k^8$. With dimensions $1 \times C_k^8$, the resulting histogram serves as the image's representation. Corners, spots, edges, and texture information about a picture are all included in the resultant feature [169]. Not every response is taken into account while creating the LDP code, which is one of LDP's limitations. Even though the rejected responses weren't in the top k, they might help make LDP more resilient and discriminative of patterns. Moreover, some of the top k directional answers might be in a certain orientation, such as west-east.

5.4 Histogram of Oriented Gradients

Histogram of Oriented Gradients (HOG), first introduced by Dalal and Triggs [170], counts instances of gradient orientations in local picture regions to calculate image gradient. For the vertical (g_x) and horizontal (g_y) directions, HOG employs the 1D centered derivative mask $[-1,0,+1]$ and $[-1,0,+1]^T$to compute the gradient of a 64×128 image that is divided into blocks of 16×16, with each block consisting of $2\,8 \times 8$ cells. The gradient's direction (θ) and magnitude (g) are computed as

$$g = \sqrt{g_x^2 + g_y^2} \quad \text{and } \theta = \arctan\frac{g_y}{g_x} \tag{5.14}$$

where the gradient direction is represented by θ, the gradient magnitude by g, and the vertical and horizontal gradients of the image by g_x and g_y, respectively. Figure 5.6 displays an input image's x and y gradients, magnitude (g), and direction (θ).



(a) Input Image (b) x-gradient (g_x) (c) y-gradient (g_y) (d) Magnitude (g) (e) Direction (θ)

Figure 5.6 Histogram of Gradients image responses

The gradient of the image has direction and magnitude at each position (x, y). For each color channel, the color picture gradients are assessed. The direction (angle) that corresponds to the maximum gradient is the gradient at point (x, y), which is the maximum of the color channel magnitudes. Eight times eight cells are used to construct the gradient histogram. Nine bins in the resulting histogram correspond to the angles 0^o, 20^o, 40^o, and 60^o, with a sum of 160^o. A sample histogram for a single 8-times-8 cell is shown in Figure 5.8, where the number of edges at orientation 120^o is shown by 5.

8.75	2			13		5		26.25
$\mathbf{0^o}$	$\mathbf{20^o}$	$\mathbf{40^o}$	$\mathbf{60^o}$	$\mathbf{80^o}$	$\mathbf{100^o}$	$\mathbf{120^o}$	$\mathbf{140^o}$	$\mathbf{160^o}$

Figure 5.8 Histogram of Oriented Gradient

A voting mechanism is used to determine which bin θ^o falls into if direction θ^o is between any two angles $\theta_1^o \; and \; \theta_2^o$ shown in Figure 5.8. In general, $\theta_1^o \; and \; \theta_2^o$ share the entries if the direction matrix contains n entries with θ^o where $\theta\%20$. We will increase the entries at $\theta_1^o \; and \; \theta_2^o$ by

$$\theta_1 = 1 - \left(\frac{|\theta_1^o - \theta^o|}{20}\right) \times n \tag{5.15}$$

$$\theta_2 = 1 - \left(\frac{|\theta_2^o - \theta^o|}{20}\right) \times n \tag{5.16}$$

In 5.8, we demonstrate how 35 entries at orientation 165^o are shared between 0^o and 180^o. This is a sample of this voting strategy. Every eight times, these histograms are computed, and the resulting histograms are concatenated to form the HOG feature vector. Pattern recognition is done using this normalized vector.

Because it distinguishes between scenarios in which a bright object is against a dark background and those in which a dark object is against a bright background [171], HOG is primarily utilized for object detection and is therefore better suited for shape description than texture feature descriptor. For improved surface characterization, texture descriptors should record both picture edges and fine-grained textural information, such as spots.

HOG is an extremely high dimensional feature that only counts gradient occurrences in specific areas of an image, disregarding the gradient's direction. With respect to the reference pixel, a gradient at orientation 0^o can be firing east of west. To increase the discriminativeness of the resulting feature vector, it is crucial to encode a gradient's orientation, amplitude, and direction. Moreover, HOG only computes an image's vertical and horizontal gradients using the same edge weights. Filters that take into account multiple directions with varying weights for each direction, such as Kirsch masks [165], which take into account eight directions or four orientations, are used to construct more discriminative picture gradients.

5.5 Local Directional Histogram of Oriented Gradients

HOG's drawback is that it only uses vertical and horizontal edges to determine an image's gradient. Although object detection is improved by this method, texture description with HOG requires more orientation picture gradients. Four orientations of the visual gradient are encoded using the suggested method. In contrast to the HOG operator, which computes image gradients at 0^o and 90^o only, the image gradients are computed at 0^o, 45^o, 90^o, and 135^o.

Directional localization To calculate the gradient of a 64×128 image divided into blocks of 16×16, with two 8×8 cells per block, HOG uses a 1D centered derivative mask $[-1,0,+1]$ and its rotations at 45^o,90^o and 135^o for vertical (g_x), left diagonal (g_{ld}), horizontal (g_y), and right diagonal (g_{rd}). Figure 5.9 displays these filters, while Figure 5.10 displays image gradients in the horizontal, vertical, left, and right directions along with the associated magnitudes and angles.

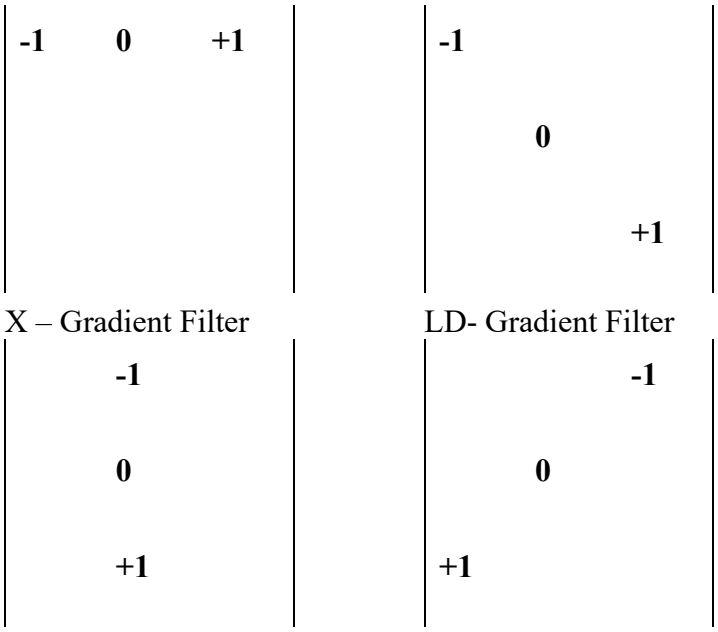
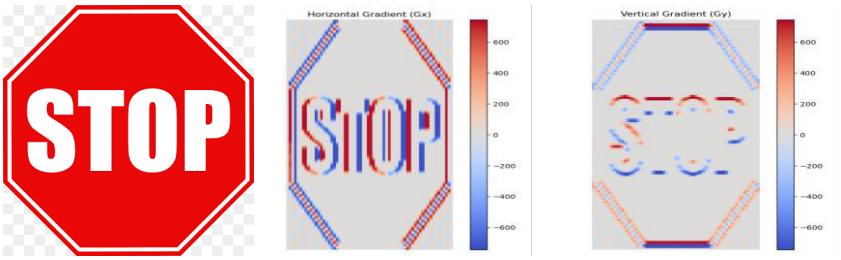


Figure 5.9 Filters used for LDHOG image gradients calculation

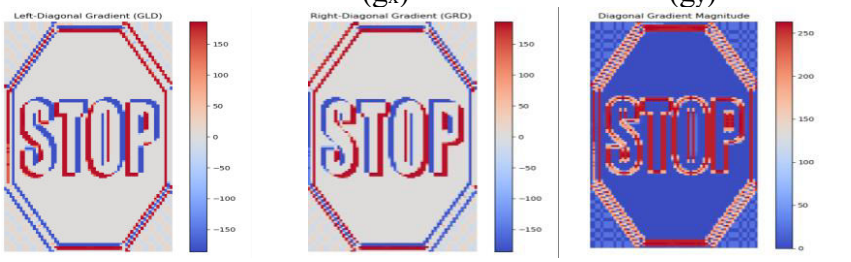
Equation 5.4 uses the x and y gradient responses to calculate the magnitude g_{xy} and angle θ_{xy}. Second magnitude g_{di} and angle θ_{di} are computed using left-diagonal and right-diagonal gradient responses as

$$g_{di} = \sqrt{g_{ld}^2 + g_{rd}^2} \text{ and } \theta_{di} = \arctan\frac{g_{ld}}{g_{rd}} \tag{5.17}$$

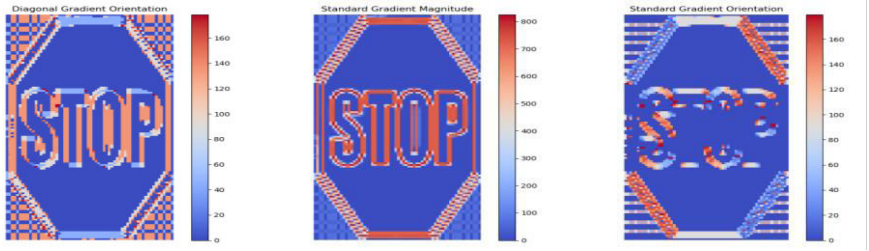
The gradient visualization for the x, y, right, and left diagonal directions, together with the appropriate magnitudes and orientations, is displayed in Figure 5.10.



a) Input Image b) X- gradient (g_x) c) y-gradient (g_y)



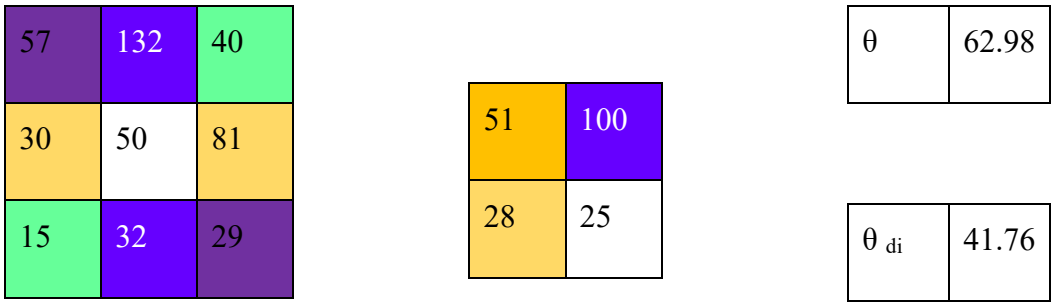
d) LD- gradient e) RD- gradient f) Magnitude g_{di}



g) Orientations θ_{di} h) Magnitude g_{xy} Orientation θ_{xy}

Figure 5.10 Gradients visualization for x, y, right and left diagonal directions

Given an image region shown in Figure 5.11 (a), four image gradients shown figure 5.11 (b) are calculated.



a) Image region b) Gradients c) Orientations

Figure 5.11 Sample LD-HOG generation process

Equations 5.14 and 5.17 are used to determine the two orientations for the current reference pixel based on these gradients, as seen in figure 5.11 (c). Two distinct histograms have these angles inserted in the appropriate bins. After that, a feature vector for TRS is created by fusing the two histograms. An image can be examined using a variety of fusion techniques. Among the fusing procedures are:

i). MAX pooling: The maximum angle is selected and inserted into the appropriate bin after comparing the angles in the same bin in the two histograms

ii). MIN pooling: The minimum angle is selected and inserted into the appropriate bin after comparing the angles in the same bin in the two histograms

iii). AVG pooling: The corresponding bin is filled with the average of the two angles.

The entire picture region was subjected to repeated maximum pooling techniques, and the resulting LDHOG was utilized as a feature descriptor for TSR. Cropping and resizing the image to 64\ \times\ 128\ pixels Notably, the aspect ratio needs to be 1:2 even if the image can be enlarged to any size. The image gradient for each direction is determined by calculating the image gradients for each color channel in the following directions:

horizontal, vertical, left-diagonal, and right-diagonal. The channel with the largest gradient is selected as the image gradient. The image is then separated into 8x8 pixel cells.

Consequently, 8×16 cells are produced. The cells are arranged in blocks, with two times as many cells in each block, for a total of seven times fifteen blocks. The whole feature vector is $7 \times 15 \times 2 \times 2 \times 9 = 3780$, with each cell having a 1×9 histogram feature. For the first local directional histogram, gradient orientations and their corresponding magnitudes are computed using vertical and horizontal picture gradients. The second local directional histogram's gradient and corresponding magnitude are determined using the right-diagonal and left-diagonal image gradients. The resulting feature vector is used for TSR after the two histograms are fused for each traffic sign. Prior to learning the model, the dimensionality of the final feature vector is reduced using Linear Discriminant Analysis (LDA).

5.5.1 Orientation Binning

Nine bins in the 20^o range are employed in this investigation. Nine bins, spanning from 0^o to 180^o, are present in the histogram. The entry in the corresponding bin is increased if an orientation is within the bin range. Equations 5.15 and 5.16's voting mechanism is used if an orientation falls between two bins. The selection of the number of bins was inspired by earlier studies [172], [173], which discovered that 9 bin histograms outperform 15 bin histograms in terms of accuracy.

Figure 5.12, 5.13 and 5.14 shows the bins extracted from the features extracted.

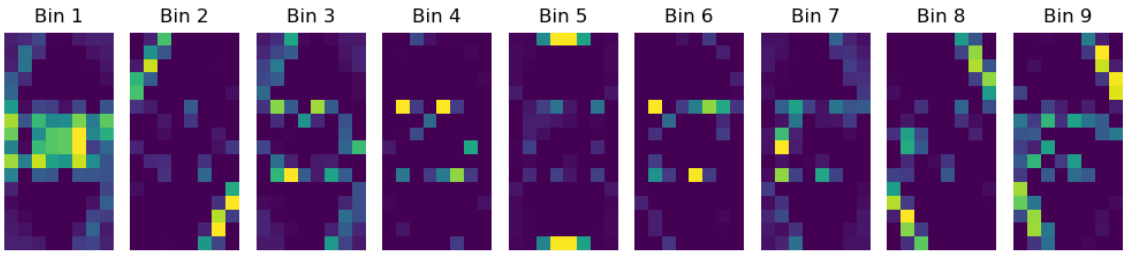


Figure 5.12 HOG bins for the sampled image

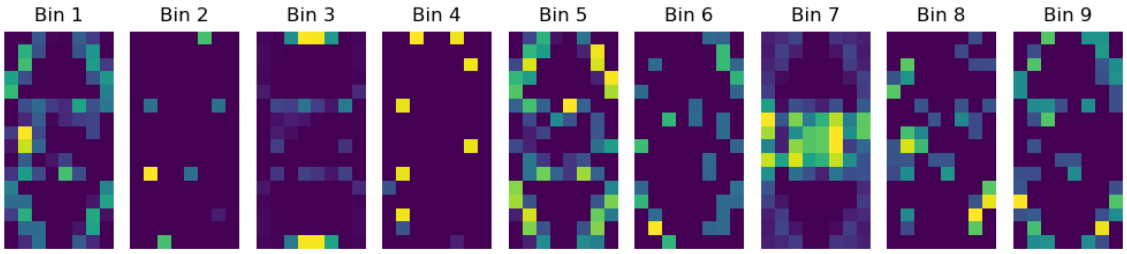


Figure 5.13 Diagonal HOG for the sampled image

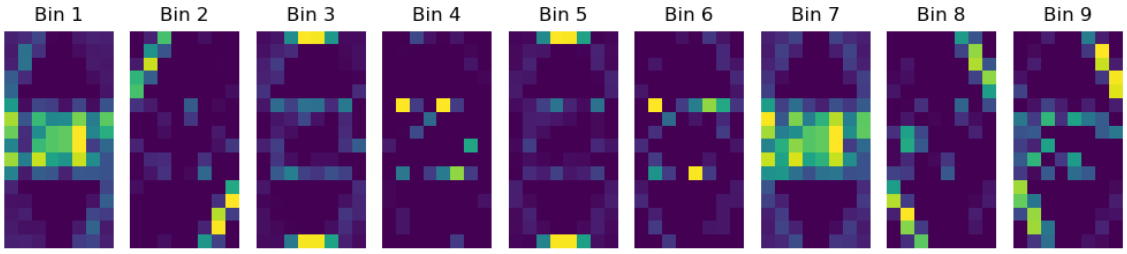


Figure 5.14 Fused bins for the sampled image

5.5.2 Normalization

The L2-Norm normalization technique is used to normalize histogram blocks as

$$|V|_{L2} = \frac{v}{\sqrt{\sum |v|^2 + \epsilon^2}} \tag{5.18}$$

where $\boldsymbol{\epsilon = 0.00001}$ is a tiny constant used to prevent overflow when $\boldsymbol{v} = \boldsymbol{0}$, $\vdash.|\boldsymbol{V}\dashv.|\_\{L2\}$ is a normalized feature, and $\vdash.|\boldsymbol{v}\dashv.|$ is a none normalized feature. Because it outperforms L1 normalization, this normalization technique is used [172].

5.6 Chapter Summary

A feature extractor called LD-HOG was created to encode image gradients. The HOG operator is extended by the LD-HOG operator. Two image gradients are computed in LD-HOG by rotating a 1D derivative mask [−1, 0, +1] at 45, 90, and 135 degrees. A feature vector for traffic sign recognition is created by fusing the histograms of the two gradients using maximum pooling techniques. LD-HOG performs better than HOG feature descriptors in TSR, according to experimental results on the GTSRB dataset. In contrast to using only four neighbors, the performance of LD-HOG in TSR shows that utilizing all eight neighbors of a pixel when encoding the picture gradient at that specific pixel produces robust features for object classification.

Each pixel pi in a digital image has $n = 2(d + 2)$ neighbors at distance d from it, $p0, p1, \ldots pn - 1$. At a distance d from pixel pi, each neighboring pixel has subtle information about the image's gradient. Therefore, all of the nearby pixels at distance d should be taken into account when encoding the image gradient at pi.

CHAPTER SIX

MACHINE LEARNING FOR TRAFFIC SIGN RECOGNITION MODELS

6.1 Overview

The chapter discussed the procedure and the results of creating a machine learning model using LD-HOG described in chapter five. The chapter starts by analyzing the GTSRB that provided traffic sign images that were used in training and testing the model. The chapter then discuss the process of model creation using three classifiers, SVM, random forest and decision tree. The study further discusses the results of comparison of model created by LD-HOG, HOG and LDP outlining precision, recall and F1-score. Finally, the chapter shows how the performances of the models were validated.

6.2 Dataset

The study utilized traffic sign images from German Traffic Sign Detection Benchmark (GTSRB) dataset. Introduced by [174] in 2012, GTSRB is an open-source dataset that has 43 classes of traffic signs (51,840 images). The Dataset has: training set with 39,209 images and the testing set with 12,631 images. The dataset has a representation of real-world scenarios including different brightness, occlusion and different sizes, and was collected from different locations in Germany. The study applied this dataset in training and testing of LDHOG for TSR. Figure 4.1 shows a sample of images for each class.

The dataset was selected because it has different shape, brightens, size and color distributions and it has been widely used in TSR research as shown by figure 6.2 below. The test sample was 20 percent of the dataset. The training sample was 80 percent.

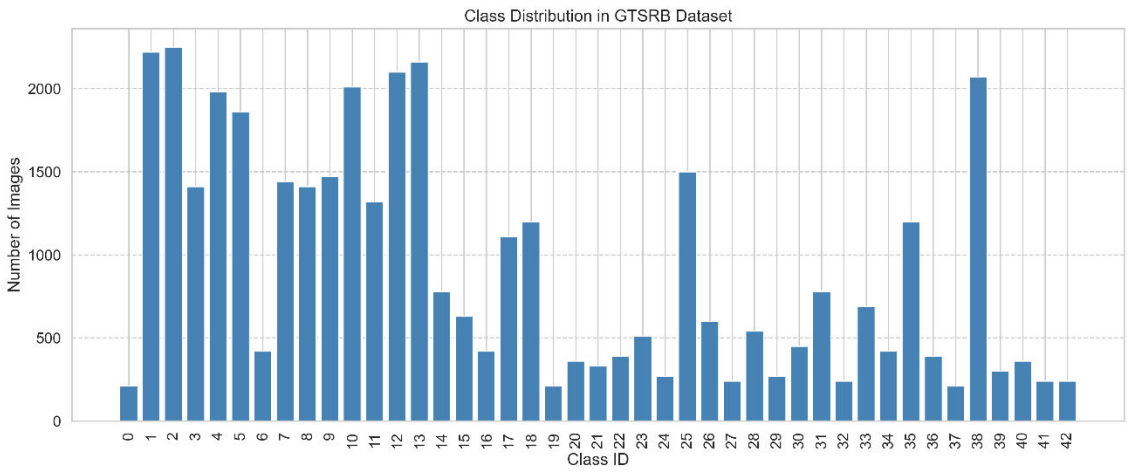


Figure 6.2 Class distribution of GTSRB dataset

Figure 6.3 Shape distribution for GTSRB dataset

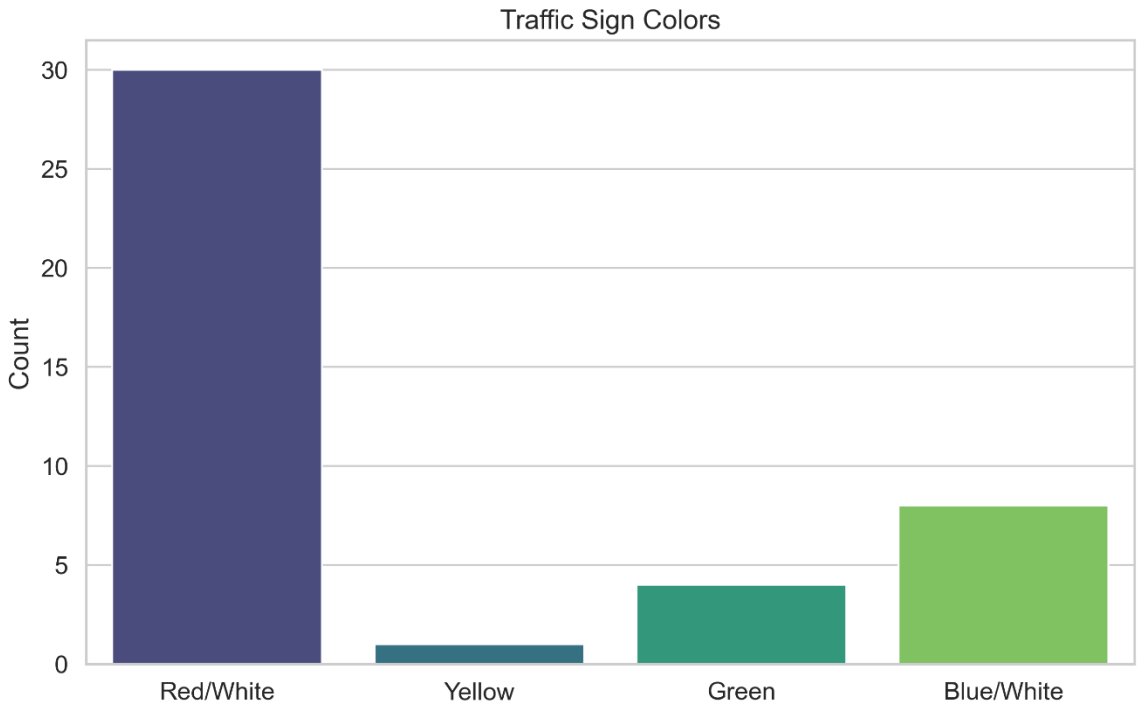


Figure 6.4 Color distribution of GTSRB dataset

6.3 Classification and regression

TSR can be modelled as a regression of classification problem. Multiclass classification was performed on The GTSRB dataset targeting 43 classes. Regression is used to model TSR classifier with experiments on SVM, random forest and decision trees as proposed by [128], [129] and [135]. An image is classified into a class Id using these classifiers. for SVM, from experiment, the following values were selected:

i). kernel='rbf'

ii). C=10

iii). gamma='scale'

iv). class weight='balanced'

v). probability=True

RBF was selected because of handling non-linearity. Traffic signs have complex shapes/textures that are non-separable in linear space and has outperformed polynomial and sigmoid kernels for vision tasks [175]. The regularization parameter C has wider margins (high probability of misclassification) with smaller values while larger value has a high probability of overfitting. 10 was selected for GTSRB task from experiment results. The kernel co-efficient was set to scale to allow for adaptability of feature scaling and avoids manual tuning pitfalls. GTSRB dataset has different number of images per class as shown in figure 6.2 causing class imbalance. To solve this class weight is set to balance to prevent biasness

6.4 Validation and evaluation protocol

The dataset was imbalanced with warning signs having more classes compared to mandatory and reservation. Therefore, stratified LD-HOG based TSR on GTSRB datasets is validated using the K-fold validation technique. This validation procedure divides the datasets into k disjoint groups of the same size at random. The training and validation process is then carried out in k iterations, with each iteration using a different fold of data for validation and the remaining k-1 for model learning. All validation mistakes are averaged to determine the estimated error. The value of k in this investigation was 5. After experimenting with various choices of k, 5 was selected as the balance between subset sizes, validation accuracy, and computational efficiency. Additionally, comparison of LD-HOG, HOG and LDP was done using precision, recall and F1-Score. Precession measures how reliable are positive predictions. High precision means low false alarm. For TSR, precision should be high as possible because misclassification can lead to an accident. Precession is calculated using equation 6.1

$$\text{Precision} = \frac{TP}{TP + FP} \tag{6.1}$$

where:

TP = True Positives (correctly predicted positives)

FP = False Positives (incorrectly predicted positives)

Recall measures how many actual positives were found. For recall, a high value implies that the model missed a few true instances. Equation 6.2 shows how recall is calculated.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6.2}$$

where:

FN = False Negatives (actual positives predicted as negatives)

F1-Score measures the harmonic mean of precision and recall. It is the best measure for imbalanced dataset like GTSRB. It is used to prevent misleading high accuracy from imbalanced. F1-score is calculate using the equation 6.3

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN} \tag{6.3}$$

6.5 Experimental results and discussion

Three experiments were performed to measure and evaluate the performance of LD-HOG in comparison to other handcrafted features HOG, and LDP. The experiments were done using three classifiers (SVM, Random Forest and Decision Trees) on GTSRB dataset.

6.5.1 Preprocessing experimental results

In this section Preprocessing experiments were done to determine the best methods for interpolation, cropping, filtering, normalization and segmentation to be used in training of the model on the GTSRB dataset.

6.5.1.1 Interpolation

The first experiment was done on image preprocessing comparing different interpolation methods on different aspects including sharpness, gradient power, Structural Similarity Index (SSIM), Time and Memory. The results of this experiment are shown in figure 6.5, 6.6, 6.7 ,6.8 and 6.9

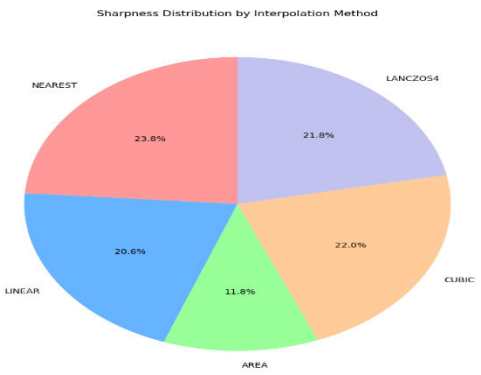
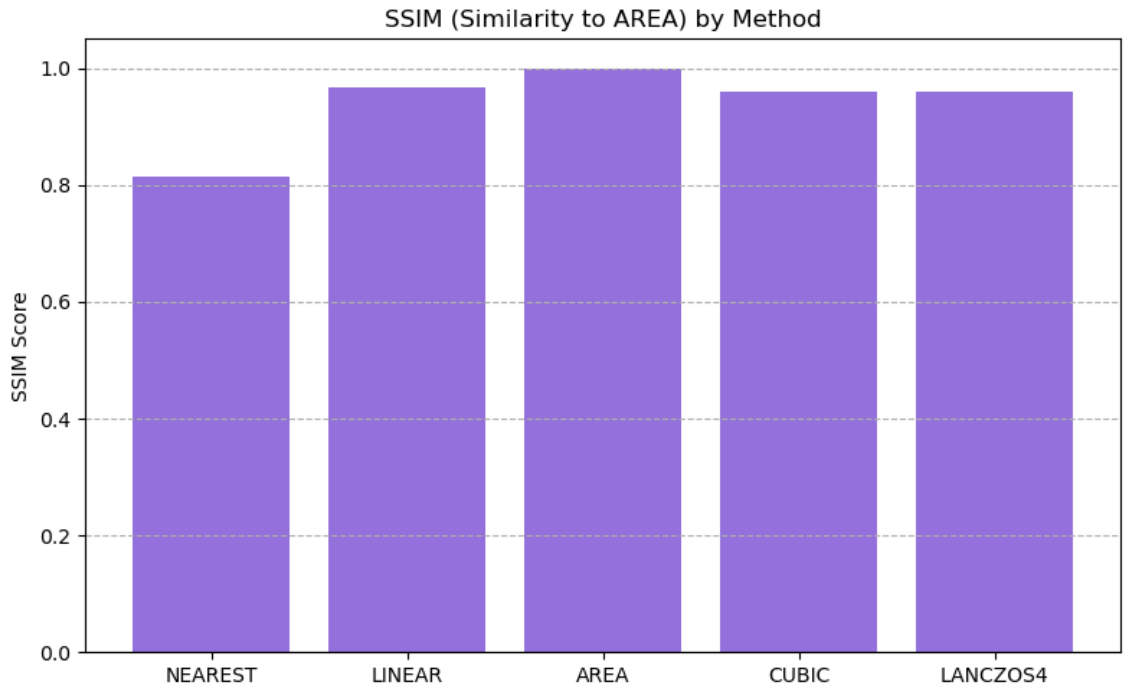


Figure 6.5 Sharpness comparison of interpolation methods



6.6 Similarity to Area comparison of interpolation methods

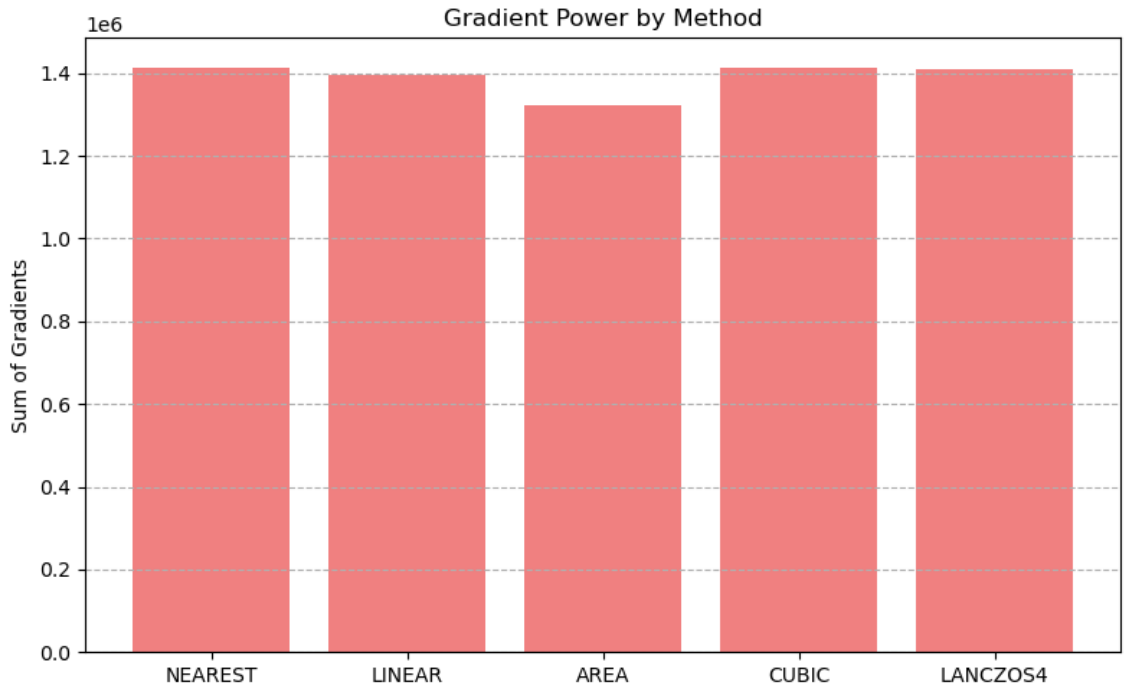


Figure 6.7 Gradient power comparison of interpolation methods

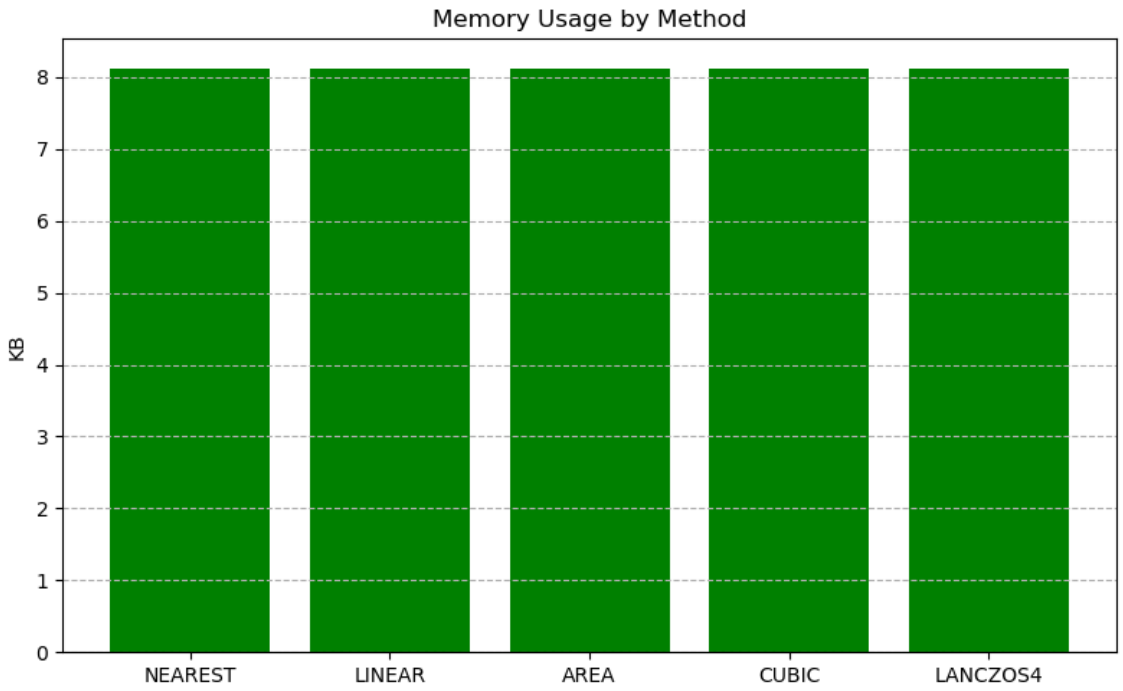


Figure 6.8 Memory Usage comparison for interpolation methods

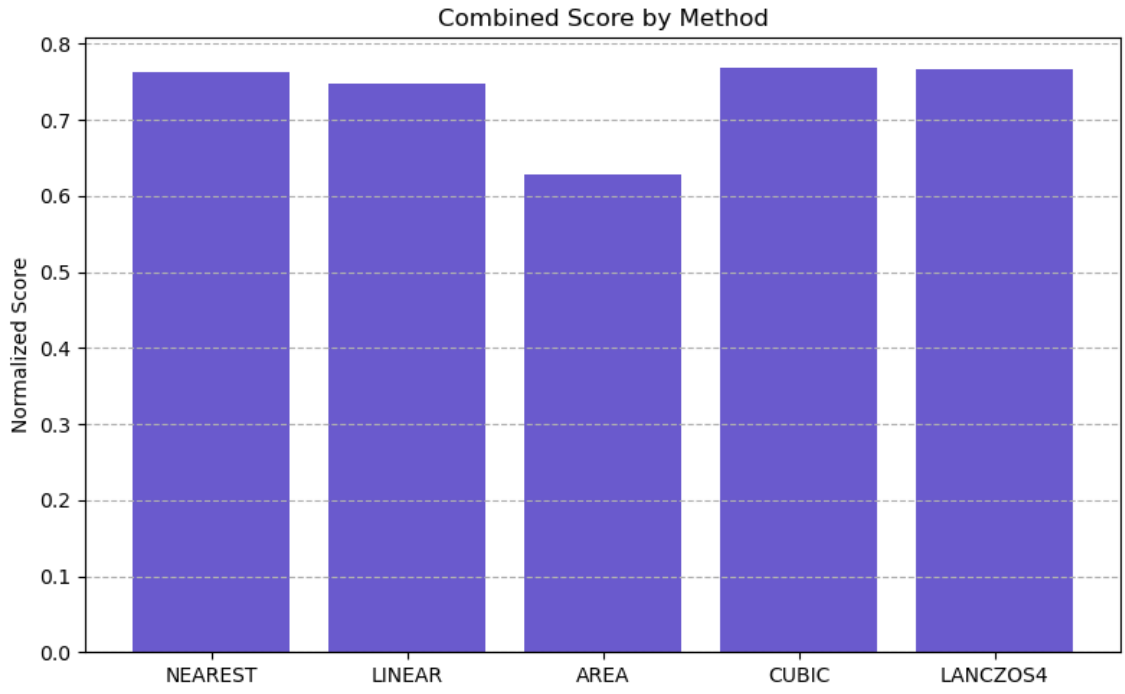


Figure 6.9 Combined score for interpolation method

Notably, time was negligible on the performance of interpolation methods. LanczoS4 interpolation method was selected as the best interpolation method for image preprocessing.

6.5.1.2 Cropping

Experiment was done to determine which was the best method to be used for cropping of the traffic sign images between region of interest, Aspect ratio and content-based image retrieval. The experiment tasted sharpness, SSIM, gradient power, retrieval accuracy and processing time. The results are discussed below

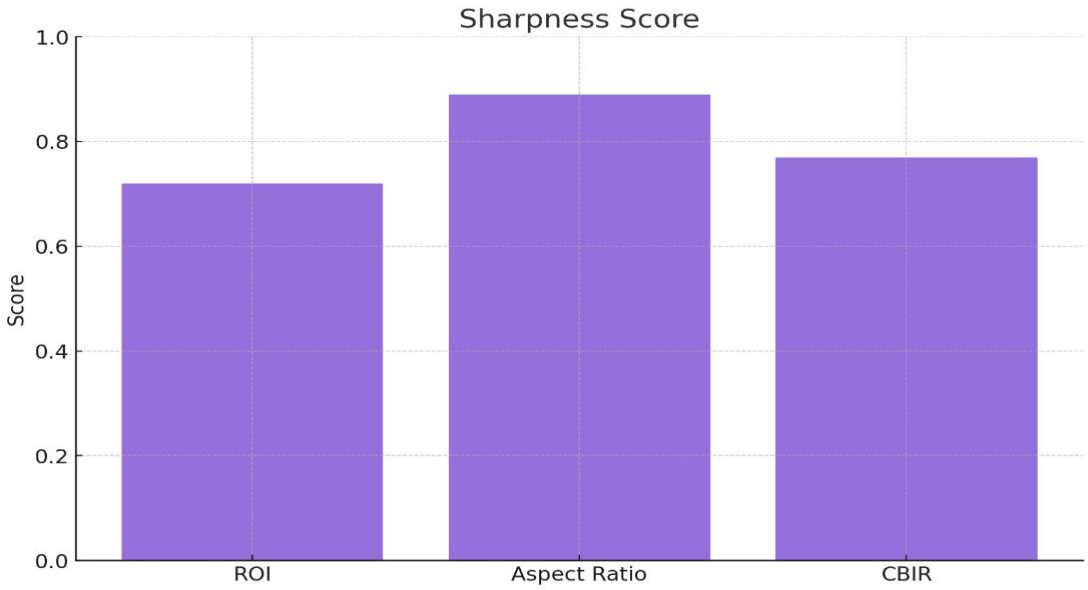


Figure 6.10 Image Sharpness comparison of cropping methods

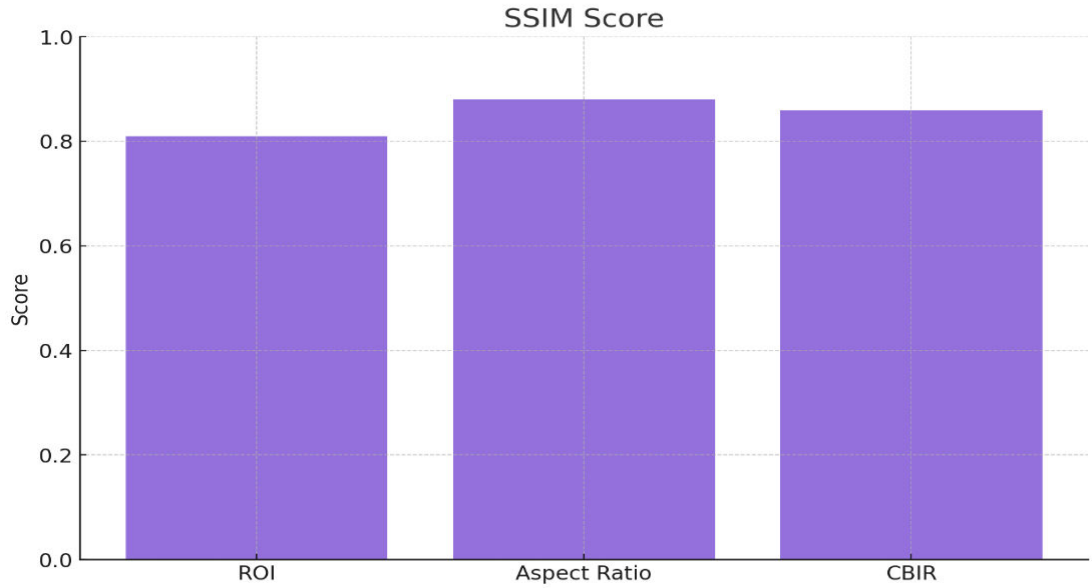


Figure 6.11 Image SSIM comparison of cropping methods

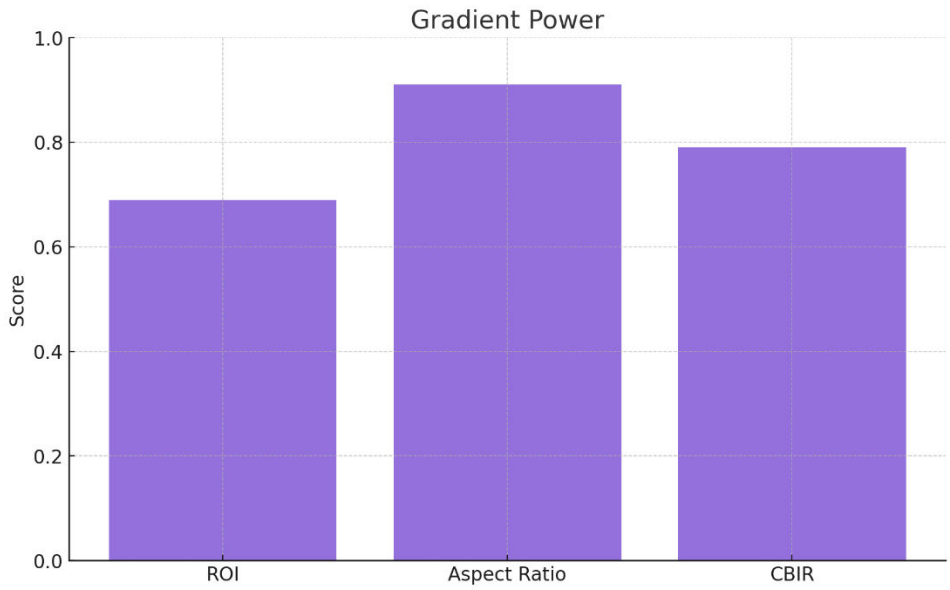


Figure 6.12 Image gradient power comparison of cropping methods

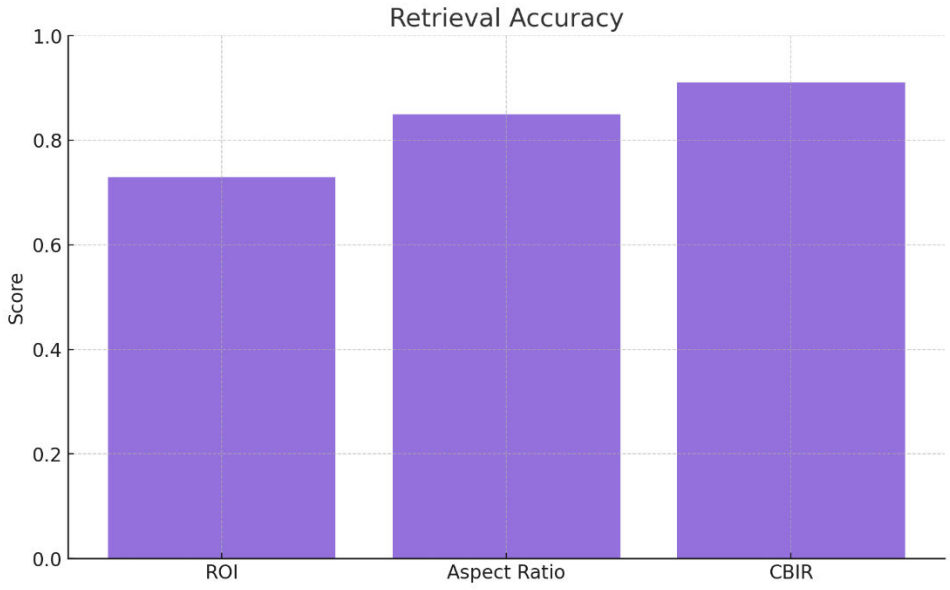


Figure 6.13 Image retrieval Accuracy comparison for cropping methods

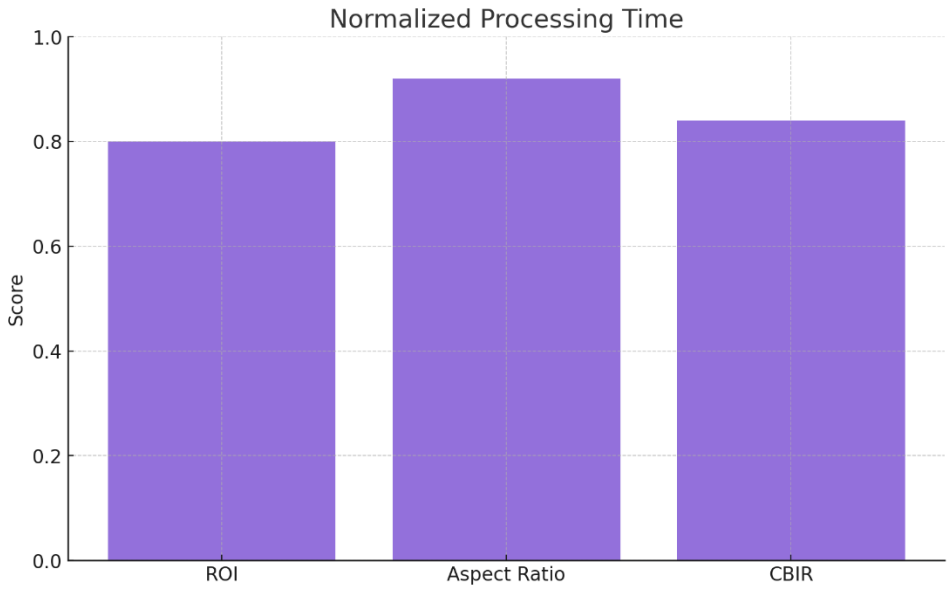


Figure 6.14 Image normalized processing time comparison for cropping methods

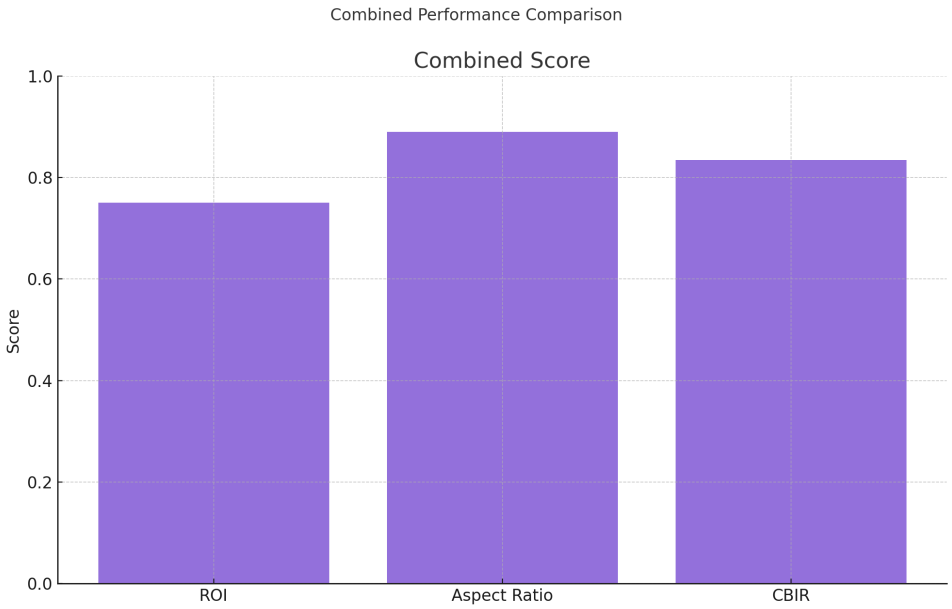


Figure 6.15 Image combines score comparison for cropping methods

All the three cropping methods performed well in retaining the sharpness of the image; however, Aspect ratio had the best score 0f 89 percent followed by CBIR at 77 percent. ROI had 72 percent as shown in figure 6.13. Structural Similarity index compares images based on luminance, contrast and structure. In this category, Aspect ratio had 88 percent, CBIR had 86 percent and ROI had 81 percent. Another indicator measured was gradient power that shows how edges and directional features are retained. Aspect ratio had 91 percent, CBIR had 79 percent and ROI has 69 percent. For image retrieval, CBIR had 91 percent, Aspect ratio had 85 percent and ROI had 73 percent. Finally, for normalized processing time, Aspect ratio had 92 percent, CBIR had 84 percent and ROI had 80 percent. Therefore, the combined score for the three-cropping method is Aspect ratio 89 percent followed by CBIR at 83.4 percent and ROI had 75 percent. Aspect ratio cropping method was selected because it out performed the other cropping method when subjected to GTSRB dataset.

6.5.1.3 Image filtering

Experiments were done on the GTSRB dataset to measure sharpness, SSIM, gradient power, denoising quality and processing time for linear, nonlinear and frequency domain filtering methods. The results are summarized by graphs and table.

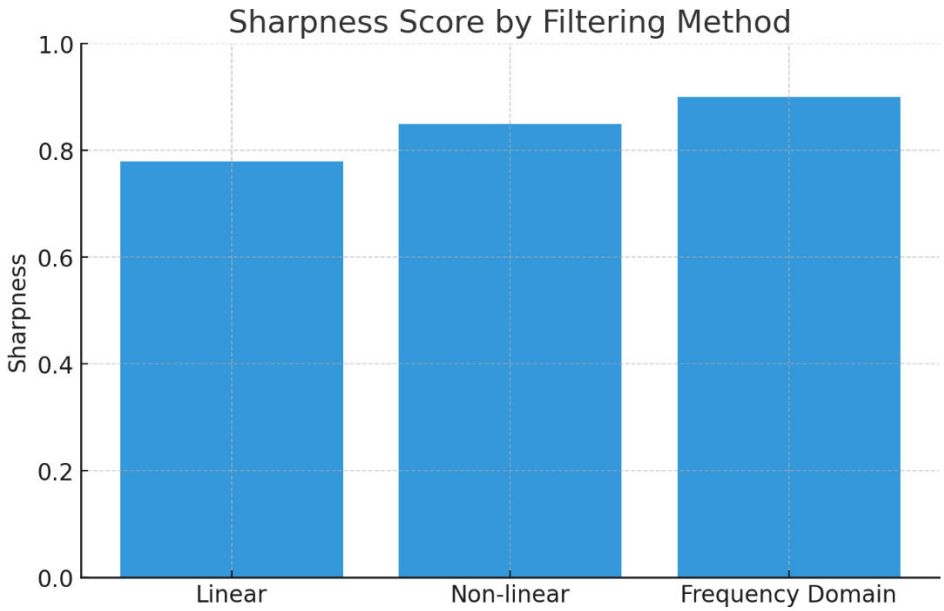


Figure 6.16 Sharpness comparison of Image filters

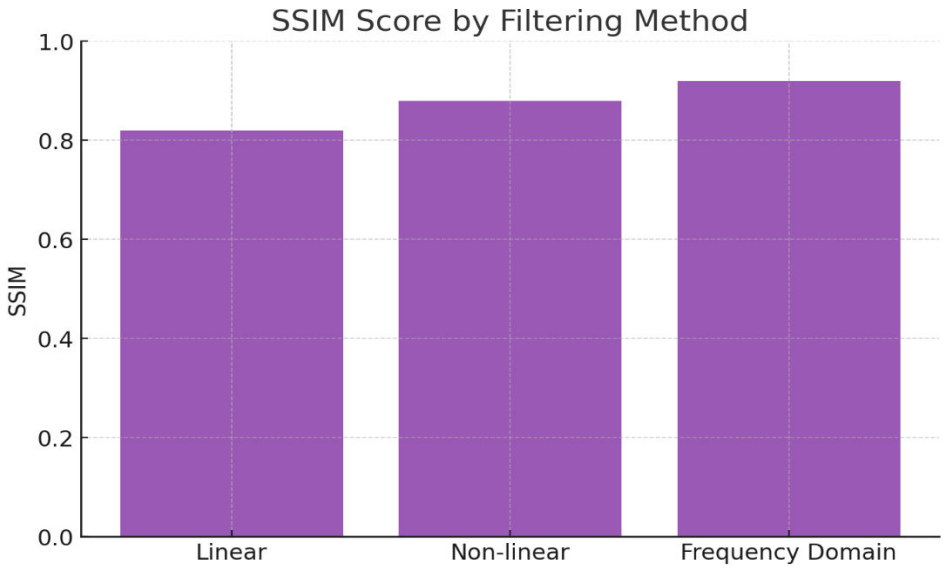


Figure 6.17 SSIM Score comparison of image filters

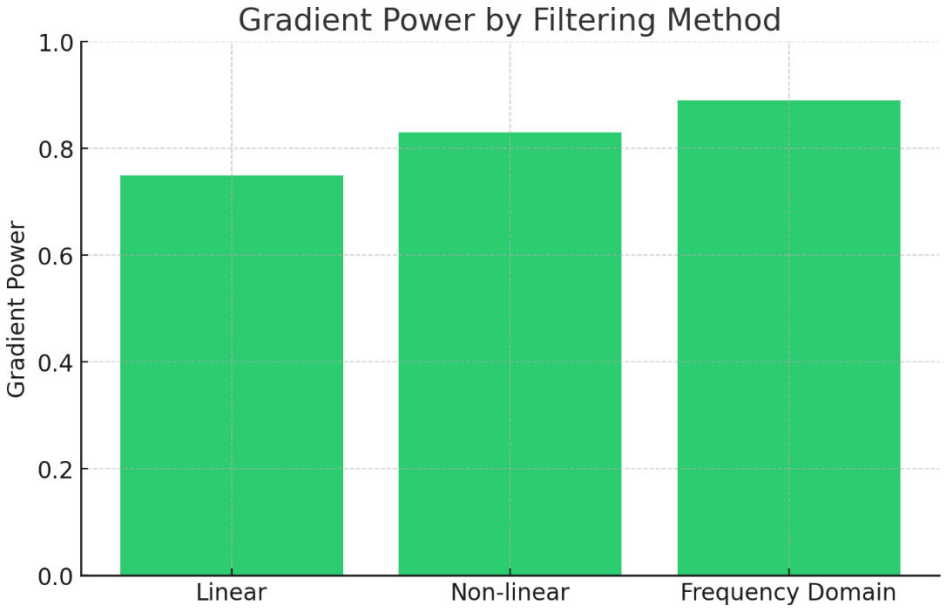


Figure 6.18 Gradient power comparison of Image filters

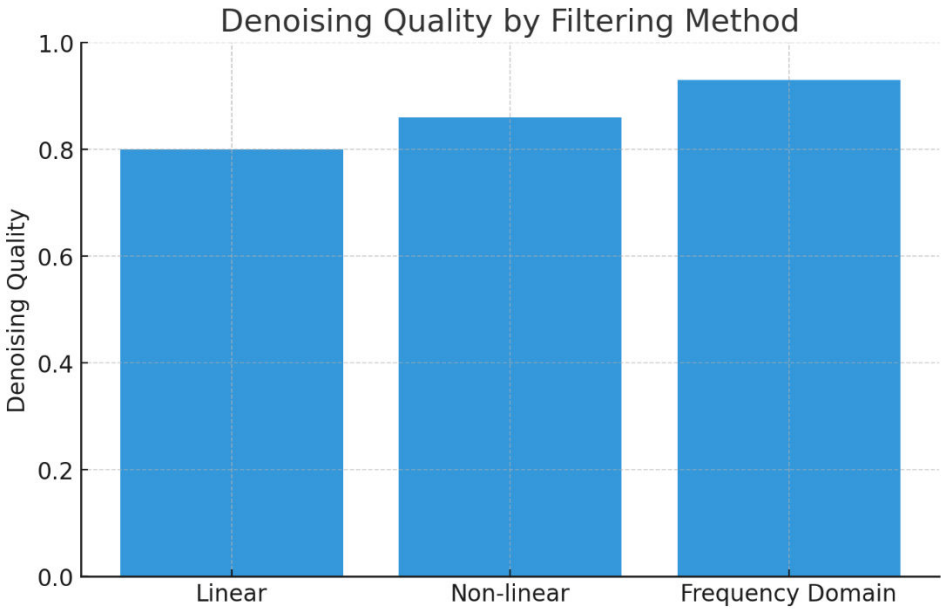


Figure 6.19 Denoising quality comparison of image filters

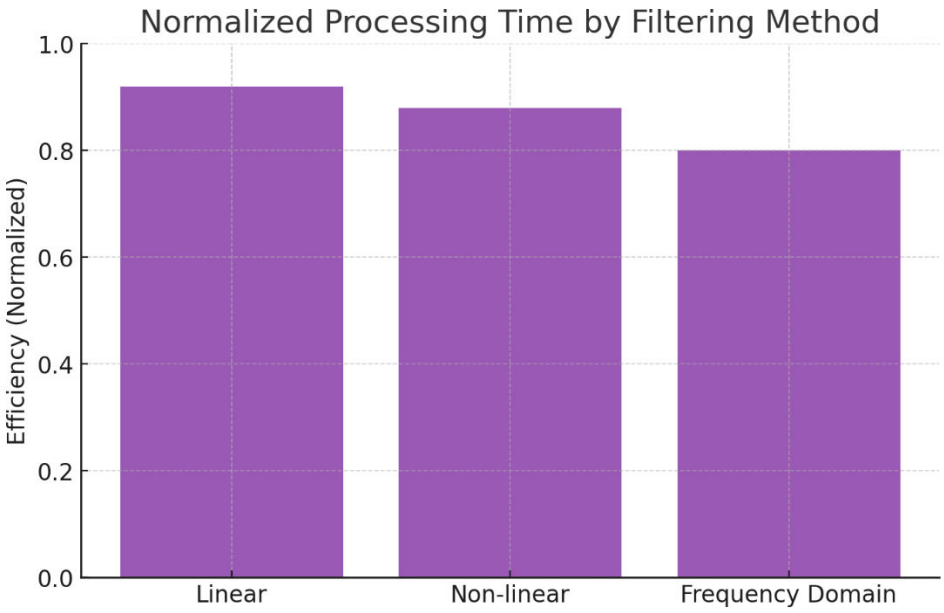


Figure 6.20 Normalized processing time comparison of image filters

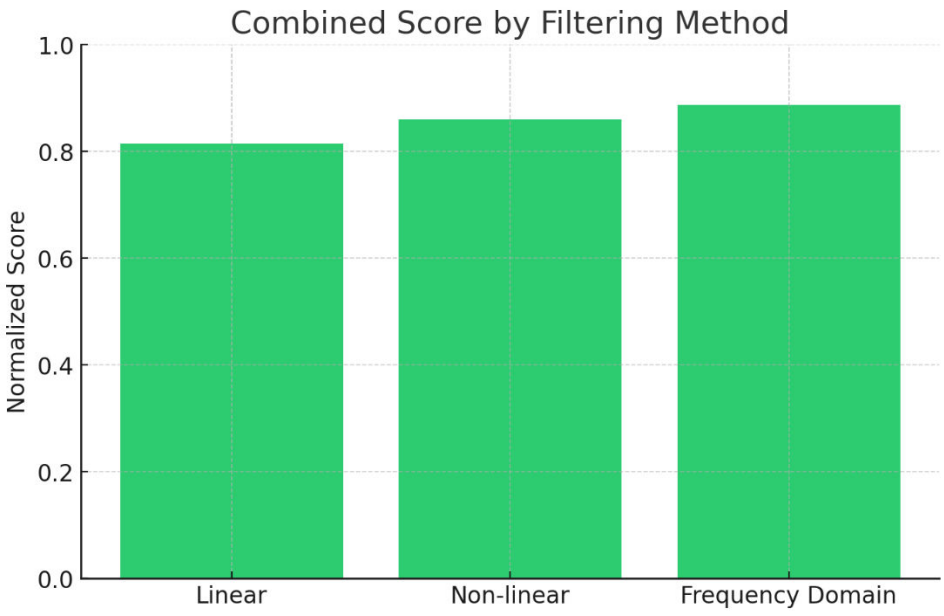


Figure 6.21 Combined score for image filters

Figure 6.16 to 6.21 summaries the performance of image filters techniques on different indicators. The study noted that

a) Frequency domain filtering yields the sharpest results while linear methods were the least sharp. Nonlinear had a decent balance score.
b) Frequency domain filtering preserves structural similarity. This suggests high quality of image after filtering
c) Frequency domain exceled in gradient power experiment
d) Frequency Domain outperformed the other filtering methods in removing noise without degrading the image. Nonetheless non-linear filters performed well here as well

e) Linear filtering outperformed the other filters in the processing time. Frequency domain was slower but performed relatively well

Considering all scores frequency domain filtering was selected to develop the model.

6.5.1.4 Image normalization

Image normalization techniques Histogram Equalization, Contrast Stretching and Spatial Normalization were tasted on sharpness, SSIM, gradient power, uniformity and processing time figure 6.22 summaries the findings of the experiments.

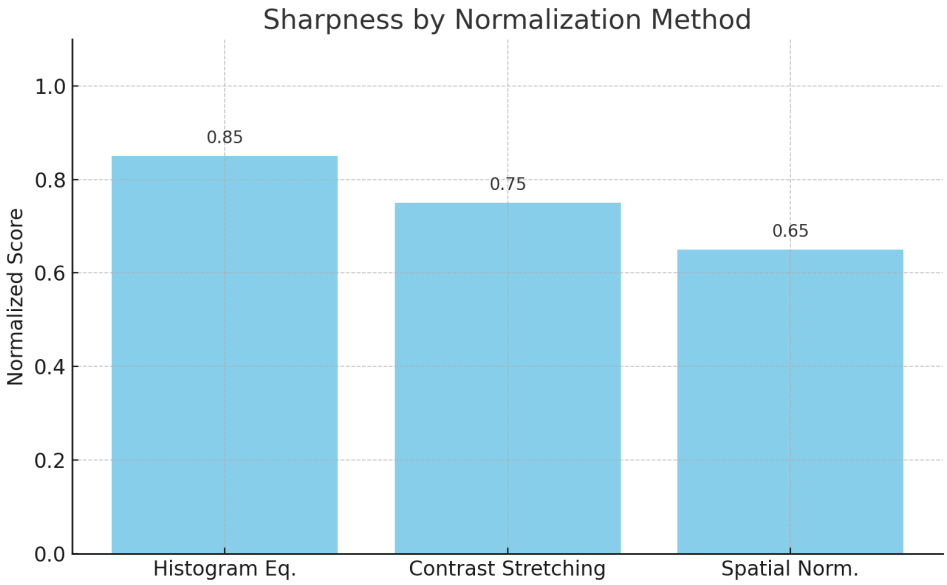


Figure 6.22 Sharpness comparison of normalization techniques

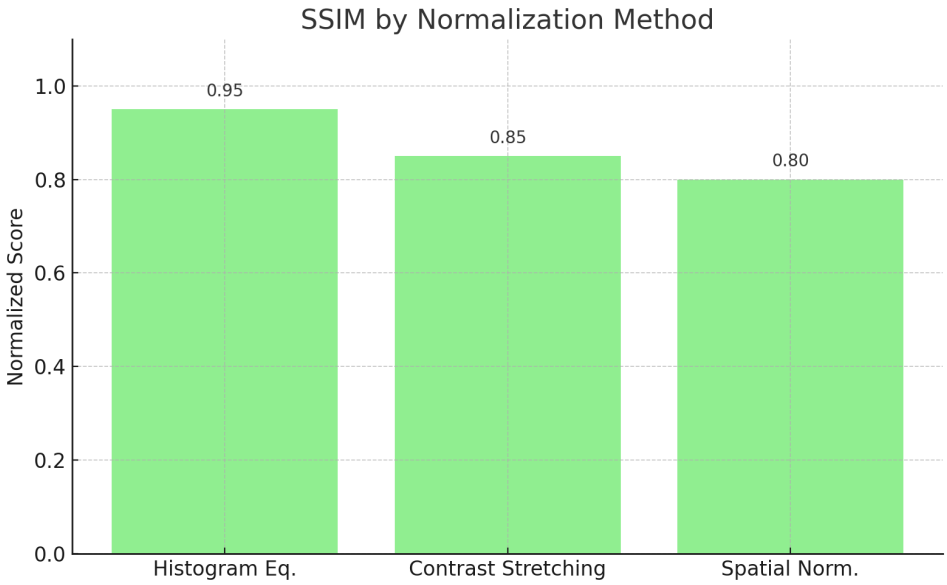


Figure 6.23 SSIM comparison of normalization methods

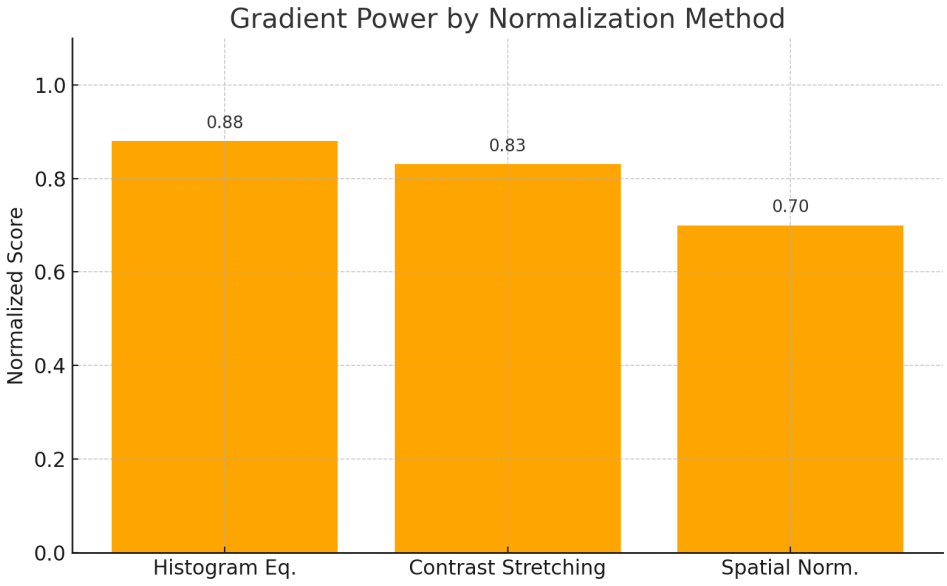


Figure 6.24 Gradient comparison of normalization methods

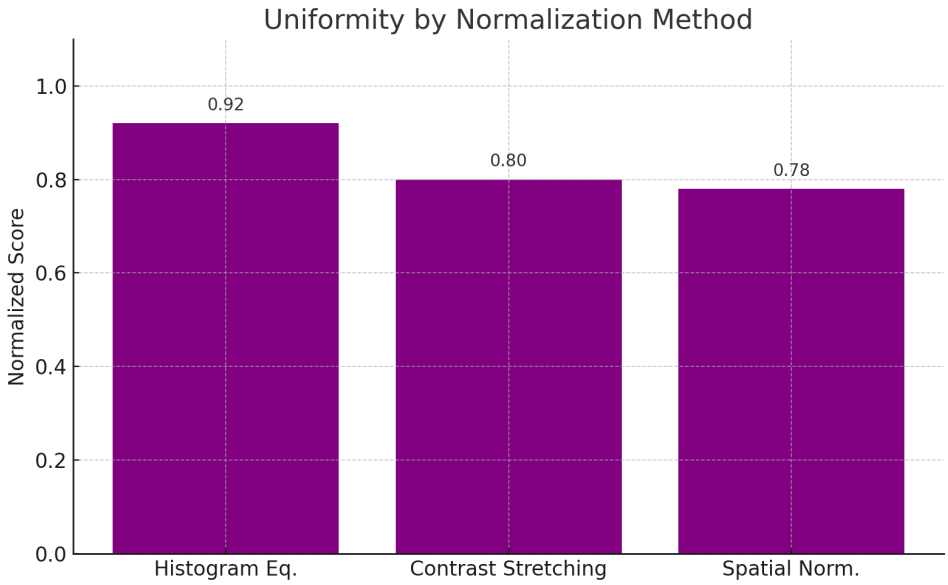


Figure 6.25 Uniformity comparison of Normalization methods

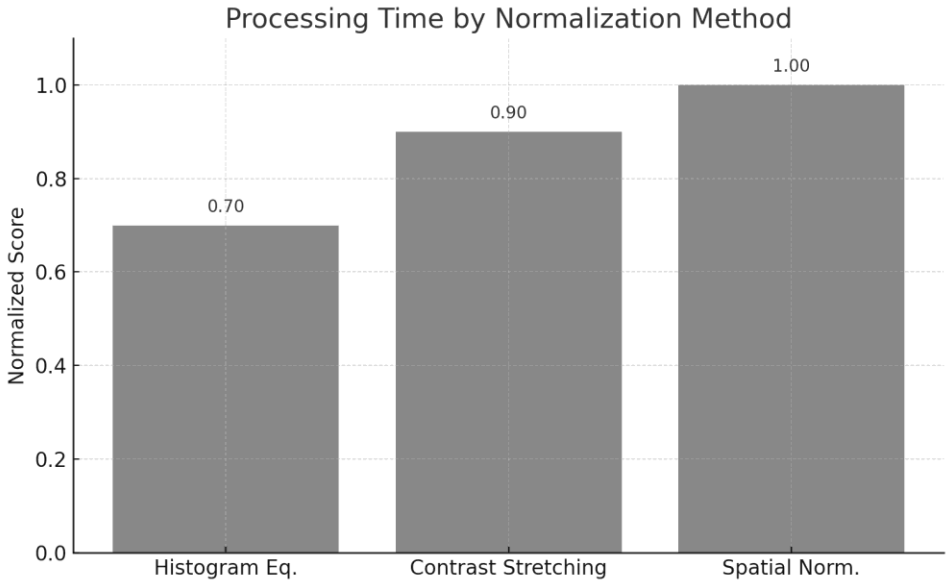


Figure 6.26 Processing time comparison of normalization method

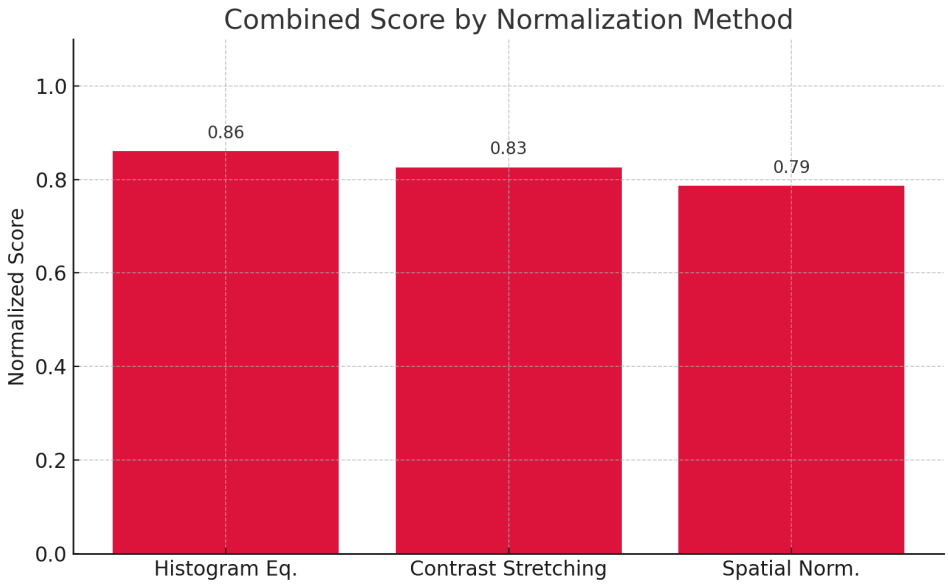


Figure 6.27 Combines score comparison for normalization methods

Figure 6.22 to figure 6.27 shows that Histogram equalization performed well in image sharpness at 85 percent followed by contrast stretching at 75 percent. Spatial normalization had 65 percent. Additionally, histogram equalization had 95 percent, contrast stretching had 85 percent and spatial normalization had 80 percent in structural similarity. Histogram equalization was the most effective at 88 percent followed by contrast stretching at 85 percent and spatial normalization was at 80 percent on gradient power retention. Another important indicator for image recognition is uniformity. In this category, histogram equalization had 92 percent, followed by contrast stretching at 80 percent. Spatial normalization had 78 percent. Finally, spatial normalization was the fastest followed by contrast stretching. Histogram was the slowest of the three. The study came up with the combined score of the three that was used to select the best normalization method to be used in model development. Histogram equalization had around 86 percent followed by

contrast stretching at around 83 percent. Spatial normalization had 79 percent. Therefore, histogram equalization was selected as normalization method.

6.5.1.6 Image Segmentation

Experiments were done to compare region-based approach and edge-based approach on accuracy, SSIM, Boundary precision and processing time. Table 6.1 summaries the findings of the experiment.

Table 6.1 comparison of image segmentation techniques

Indicator	Region based approach	Edge based approach
Accuracy	53.7	46.3
SSIM	54.5	45.3
Boundary Precision	48.0	52.0
Processing time	44.8	55.2
Combine score	51.5	48.5

From table 6.1, region-based edges out as the better all-rounder with a slightly higher overall score. Therefore, region-based approach was selected.

6.5.2 Model training and testing

Three models were created using three classifiers; support vector machine, random forest and decision trees. These three classifies were selected from the findings of section 2.6.

6.5.2.1 Model training

Before training the model, first the datasets images were loaded. This was done using two python libraries, OS and CV2. The images were loaded as grayscale and were loaded per the 43 classes. Appendix 1 shows the code.

The output is an array that is passed to three functions; the original HOG feature extractor, LDP feature extractor that applies Kirsch mask and the designed LD-HOG that implements HOG and the additional diagonals as per equation 5.17 developed. Appendix 2.3 and 2.4 show these three functions used in developing the model. The next stage was to split the dataset to 80 -20 percent training dataset. This process will help us in testing precision, recall and f -1 values for the model as it always good to test the model on the images it has not been exposed to. The 80 percent was used to train the model while the 20 percent was used to measure the performance of the model. After splitting the inputs for section 6.5.1 preprocessing were passed to the algorithm for training the model. Appendix 5 – 7 show the algorithm and python code used to create the models

6.5.2.2 Model testing

Section 6.4 discussed on how the model was tested and evaluated. After training the models were subjected to the 20 percent images that were not part of the training with the aim of testing the performance of the model using precision, recall and F1 -score. Table 6.2, 6.3, 6.4 summaries the findings of the testing experiment.

Table 6.2 Comparison of LD-HOG, HOG and LDP feature extractors on the GTSRB dataset using SVM classifier

Extractor	Precision	Recall	F1-score
LD-HOG	99	99	99
HOG	96	95	95
LDP	93	92	91

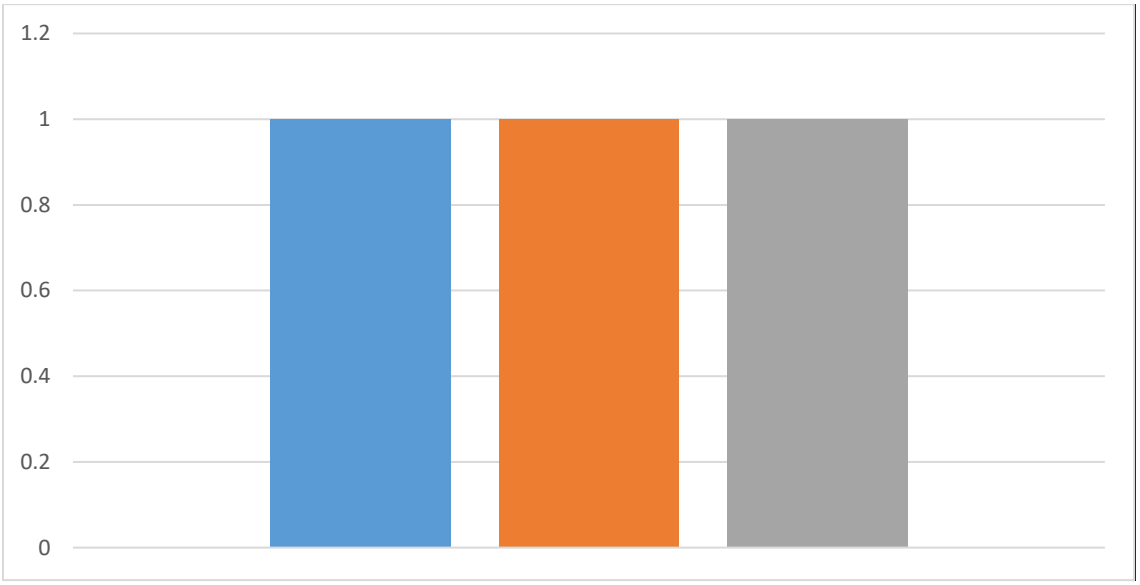


Figure 6.28 Performance of LD-HOG, HOG and LDP using SVM classifier

Table 6.2 shows that LD-HOG feature extractor outperformed the other two extractors in all the three categories; precision, recall and F1-score. LD- HOG had 99 percent in all the three categories. The original HOG performed well with 96, 95 and 95 while LDP had 93,92 and 91 for precision, recall and F1 -score respectively.

Table 6.3 Comparison of LD-HOG, HOG and LDP feature extractors on the GTSRB dataset using Random Forest classifier

Extractor	Precision	Recall	F1-score
LD-HOG	95	95	95
HOG	91	89	90
LDP	87	88	88

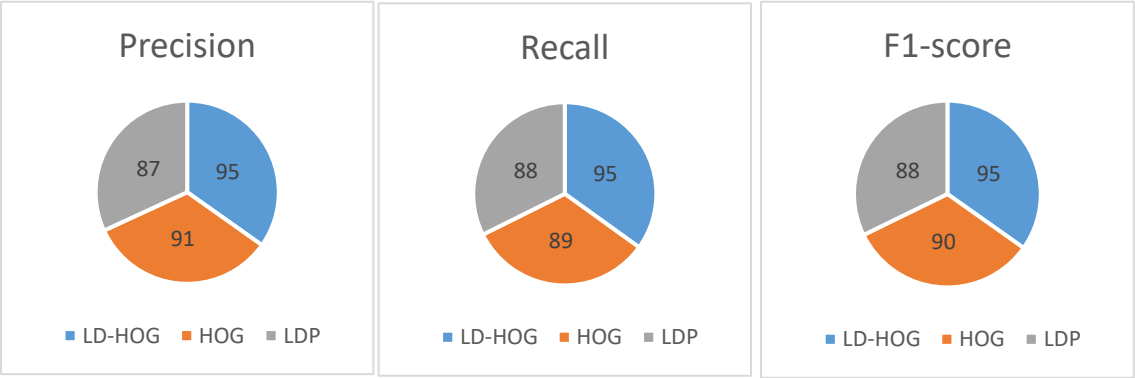


Figure 6.29 Performance of LD-HOG, HOG and LDP using Random Forest classifier

Despite being faster that SVM in training of the model, random forest results for precision, recall and FE-score were lower. Nonetheless, LD-HOG outperformed LDP and HOG with 95,95 and 95 for precision, recall and F1-score. HOG had 91, 89 and 90 while LDP had 87,88 and 88 for the same categories respectively.

Table 6.4 Comparison of LD-HOG, HOG and LDP feature extractors on the GTSRB dataset using Decision Tree classifier

Extractor	Precision	Recall	F1-score
LD-HOG	76	76	76
HOG	65	64	64
LDP	61	62	62

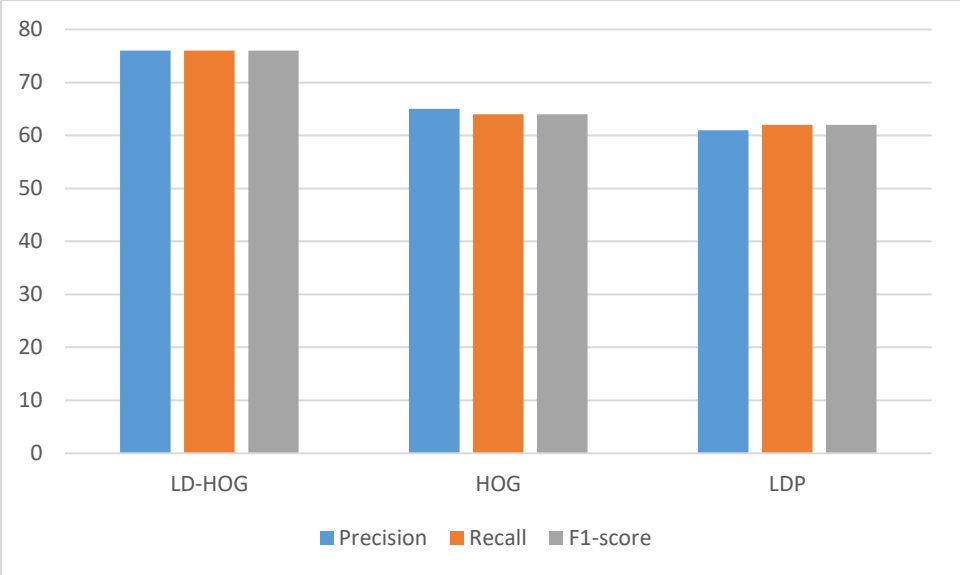


Figure 6.30 Performance of LD-HOG, HOG and LDP using Decision tree classifier

The last classifier was the lowest among the three classifiers. Decision tree had the same pattern as the other two with LD-HOG performing better than the other two. LD-HOG had 76,76 and 76 while HOG had 65,64 and 64 for precision, recall and F1 score. LDP had 61, 62 and 62 percent for the same categories.

Table 6.5 Average Comparison of LD-HOG, HOG and LDP feature extractors on the GTSRB dataset using the three classifiers

Extractor	Precision	Recall	F1-score
LD-HOG	90	90	90
HOG	84	82	83
LDP	80	81	80

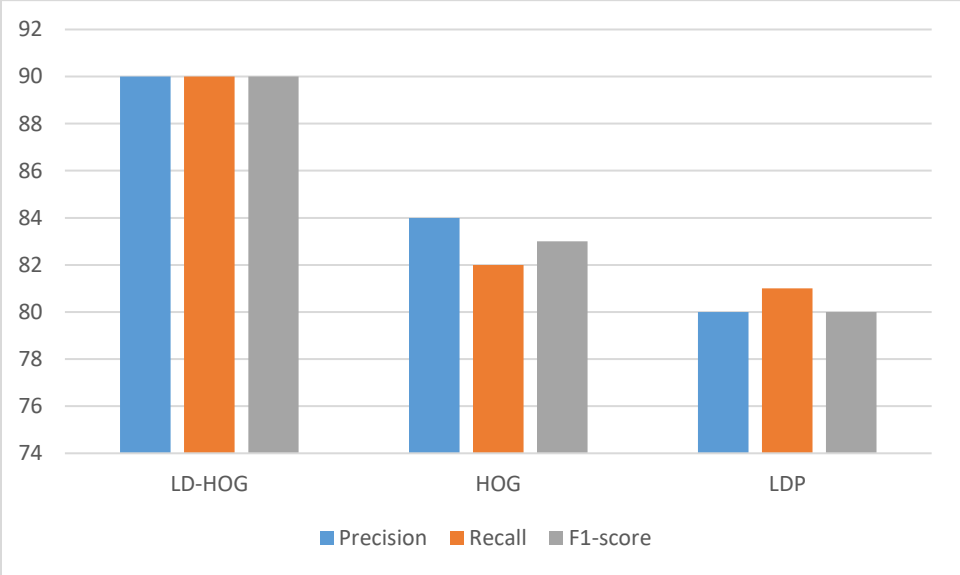
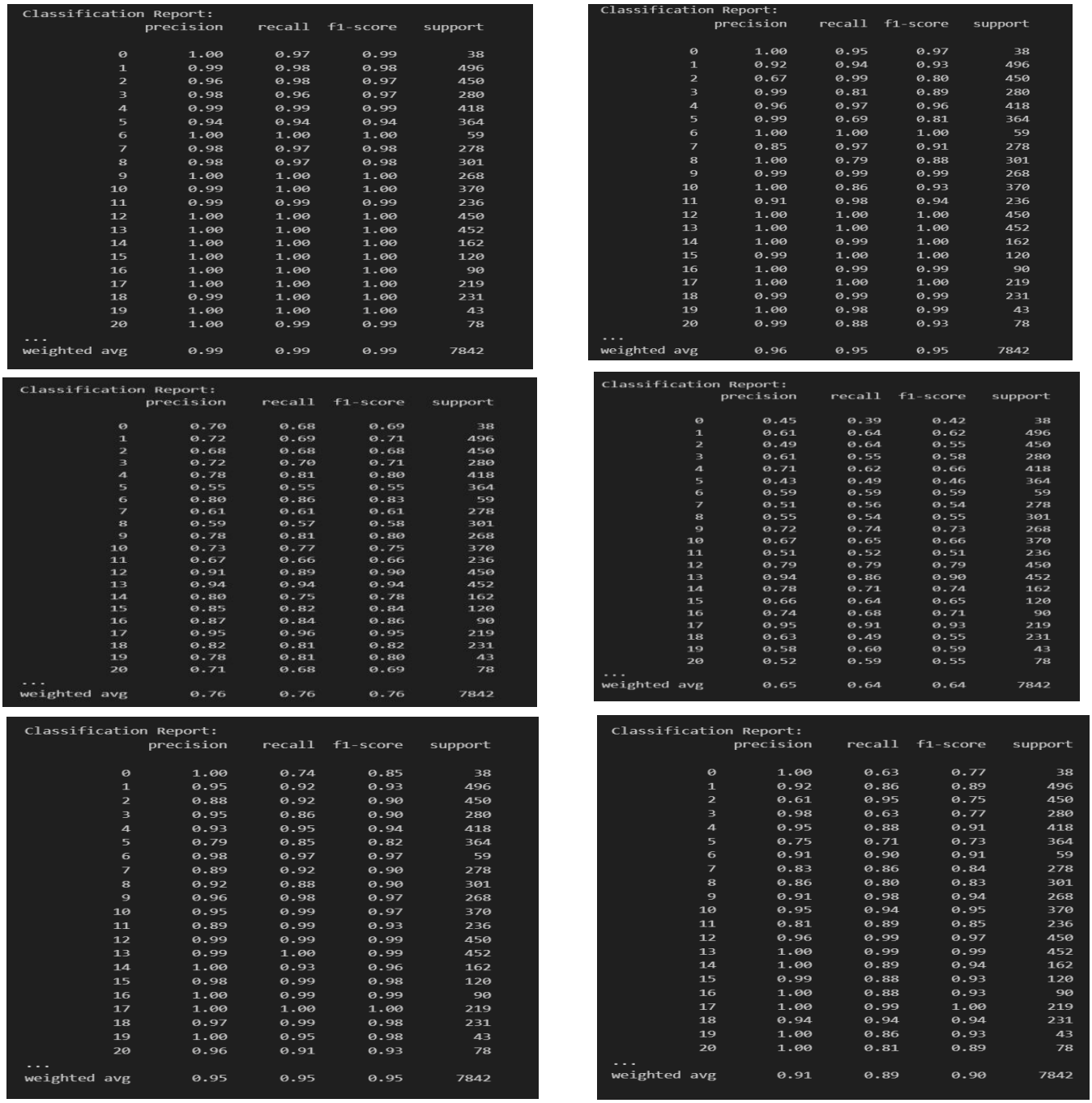


Figure6.31: Average Performance of LD-HOG, HOG and LDP using the three classifiers

Therefore, on average, LD-HOG outperformed the other two on all the three classifiers. HOG was second and LDP third as shown in table 6.5. Appendix shows the code and algorithm used for model testing.

Figure 6.32 show random screenshots from the integrated development environment for the model testing.



Classification Report:

	precision	recall	f1-score	support
0	1.00	0.97	0.99	38
1	0.99	0.98	0.98	496
2	0.96	0.98	0.97	450
3	0.98	0.96	0.97	280
4	0.99	0.99	0.99	418
5	0.94	0.94	0.94	364
6	1.00	1.00	1.00	59
7	0.98	0.97	0.98	278
8	0.98	0.97	0.98	301
9	1.00	1.00	1.00	268
10	0.99	1.00	1.00	370
11	0.99	0.99	0.99	236
12	1.00	1.00	1.00	450
13	1.00	1.00	1.00	452
14	1.00	1.00	1.00	162
15	1.00	1.00	1.00	120
16	1.00	1.00	1.00	90
17	1.00	1.00	1.00	219
18	0.99	1.00	1.00	231
19	1.00	1.00	1.00	43
20	1.00	0.99	0.99	78
...				
weighted avg	0.99	0.99	0.99	7842

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.95	0.97	38
1	0.92	0.94	0.93	496
2	0.67	0.99	0.80	450
3	0.99	0.81	0.89	280
4	0.96	0.97	0.96	418
5	0.99	0.69	0.81	364
6	1.00	1.00	1.00	59
7	0.85	0.97	0.91	278
8	1.00	0.79	0.88	301
9	0.99	0.99	0.99	268
10	1.00	0.86	0.93	370
11	0.91	0.98	0.94	236
12	1.00	1.00	1.00	450
13	1.00	1.00	1.00	452
14	1.00	0.99	1.00	162
15	0.99	1.00	1.00	120
16	1.00	0.99	0.99	90
17	1.00	1.00	1.00	219
18	0.99	0.99	0.99	231
19	1.00	0.98	0.99	43
20	0.99	0.88	0.93	78
...				
weighted avg	0.96	0.95	0.95	7842

Classification Report:

	precision	recall	f1-score	support
0	0.70	0.68	0.69	38
1	0.72	0.69	0.71	496
2	0.68	0.68	0.68	450
3	0.72	0.70	0.71	280
4	0.78	0.81	0.80	418
5	0.55	0.55	0.55	364
6	0.80	0.86	0.83	59
7	0.61	0.61	0.61	278
8	0.59	0.57	0.58	301
9	0.78	0.81	0.80	268
10	0.73	0.77	0.75	370
11	0.67	0.66	0.66	236
12	0.91	0.89	0.90	450
13	0.94	0.94	0.94	452
14	0.80	0.75	0.78	162
15	0.85	0.82	0.84	120
16	0.87	0.84	0.86	90
17	0.95	0.96	0.95	219
18	0.82	0.81	0.82	231
19	0.78	0.81	0.80	43
20	0.71	0.68	0.69	78
...				
weighted avg	0.76	0.76	0.76	7842

Classification Report:

	precision	recall	f1-score	support
0	0.45	0.39	0.42	38
1	0.61	0.64	0.62	496
2	0.49	0.64	0.55	450
3	0.61	0.55	0.58	280
4	0.71	0.62	0.66	418
5	0.43	0.49	0.46	364
6	0.59	0.59	0.59	59
7	0.51	0.56	0.54	278
8	0.55	0.54	0.55	301
9	0.72	0.74	0.73	268
10	0.67	0.65	0.66	370
11	0.51	0.52	0.51	236
12	0.79	0.79	0.79	450
13	0.94	0.86	0.90	452
14	0.78	0.71	0.74	162
15	0.66	0.64	0.65	120
16	0.74	0.68	0.71	90
17	0.95	0.91	0.93	219
18	0.63	0.49	0.55	231
19	0.58	0.60	0.59	43
20	0.52	0.59	0.55	78
...				
weighted avg	0.65	0.64	0.64	7842

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.74	0.85	38
1	0.95	0.92	0.93	496
2	0.88	0.92	0.90	450
3	0.95	0.86	0.90	280
4	0.93	0.95	0.94	418
5	0.79	0.85	0.82	364
6	0.98	0.97	0.97	59
7	0.89	0.92	0.90	278
8	0.92	0.88	0.90	301
9	0.96	0.98	0.97	268
10	0.95	0.99	0.97	370
11	0.89	0.99	0.93	236
12	0.99	0.99	0.99	450
13	0.99	1.00	0.99	452
14	1.00	0.93	0.96	162
15	0.98	0.99	0.98	120
16	1.00	0.99	0.99	90
17	1.00	1.00	1.00	219
18	0.97	0.99	0.98	231
19	1.00	0.95	0.98	43
20	0.96	0.91	0.93	78
...				
weighted avg	0.95	0.95	0.95	7842

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.63	0.77	38
1	0.92	0.86	0.89	496
2	0.61	0.95	0.75	450
3	0.98	0.63	0.77	280
4	0.95	0.88	0.91	418
5	0.75	0.71	0.73	364
6	0.91	0.90	0.91	59
7	0.83	0.86	0.84	278
8	0.86	0.80	0.83	301
9	0.91	0.98	0.94	268
10	0.95	0.94	0.95	370
11	0.81	0.89	0.85	236
12	0.96	0.99	0.97	450
13	1.00	0.99	0.99	452
14	1.00	0.89	0.94	162
15	0.99	0.88	0.93	120
16	1.00	0.88	0.93	90
17	1.00	0.99	1.00	219
18	0.94	0.94	0.94	231
19	1.00	0.86	0.93	43
20	1.00	0.81	0.89	78
...				
weighted avg	0.91	0.89	0.90	7842

Figure 6.32 Random screenshots of raw testing results

6.5.3 Model Validation

Table 6.6 show how stratified K-Fold validation results of LD-HOG using the three classifies SVM outperformed RF and DT.

Table 6.6: Stratified K-Fold Cross-Validation Results for LD-HOG Features

Classifier	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average Score	Standard Deviation (σ)
SVM	98.5	98.6	98.9	98.8	99.0	**98.8**	**0.18**
Random Forest (RF)	94.5	94.8	94.9	94.7	95.2	94.8	0.25
Decision Tree (DT)	76.3	76.3	76.4	76.7	76.7	76.5	0.17

Table 6.6 shows how the F1-scores performed for a 5-fold stratified cross-validation on three classifiers (SVM, RF, and DT) trained using LD-HOG features. Notably, SVM achieved the highest F1-score 98.8 with minimal variance 0f 0.18 showing it had high precision and stability across the folds. Random forest had an average score of 94.8 percent while decision tree had 76.5 percent. All the three classifiers had a low standard deviation across the folds suggesting consistent model performance.

The study further validated the model by considering the confusion matrix. Figure 6.33 shows the confusion matrix for the models of the study.

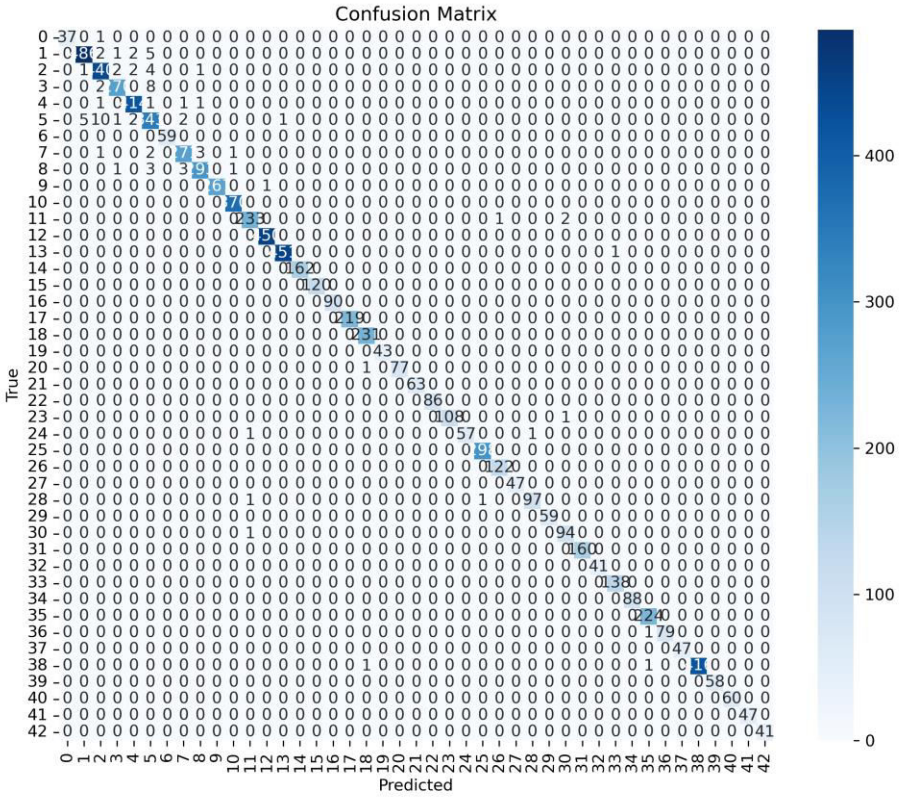


Figure 6.33 shows the confusion matrix for LD-HOG and SVM

Figure 6.33 shows the confusion matrix obtained using the LD-HOG features with SVM classification. The matrix confirms that the model achieves high accuracy across all 43 classes, with most predictions falling along the diagonal. Some misclassifications occur between similar-looking classes, which is expected in real-world datasets. The confusion matrix supports the earlier precision, recall and F1-score-based evaluation, validating the effectiveness of the LD-HOG descriptor in multi-class image classification. Figure 6.34 and 6.35 validates the precision, recall and F1-score for LD-HOG features being classified with random forest and decision tree.

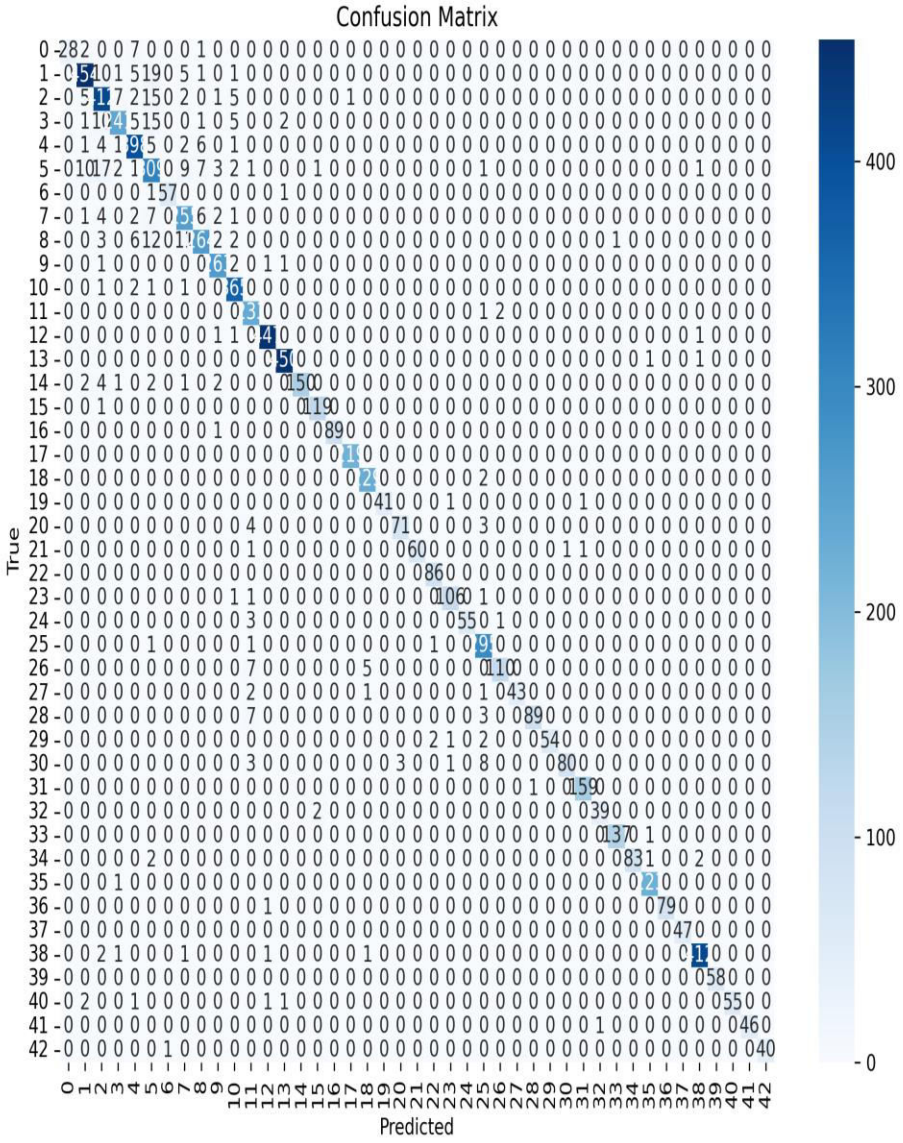


Figure 6.34 shows the confusion matrix for LD-HOG and Random Forest

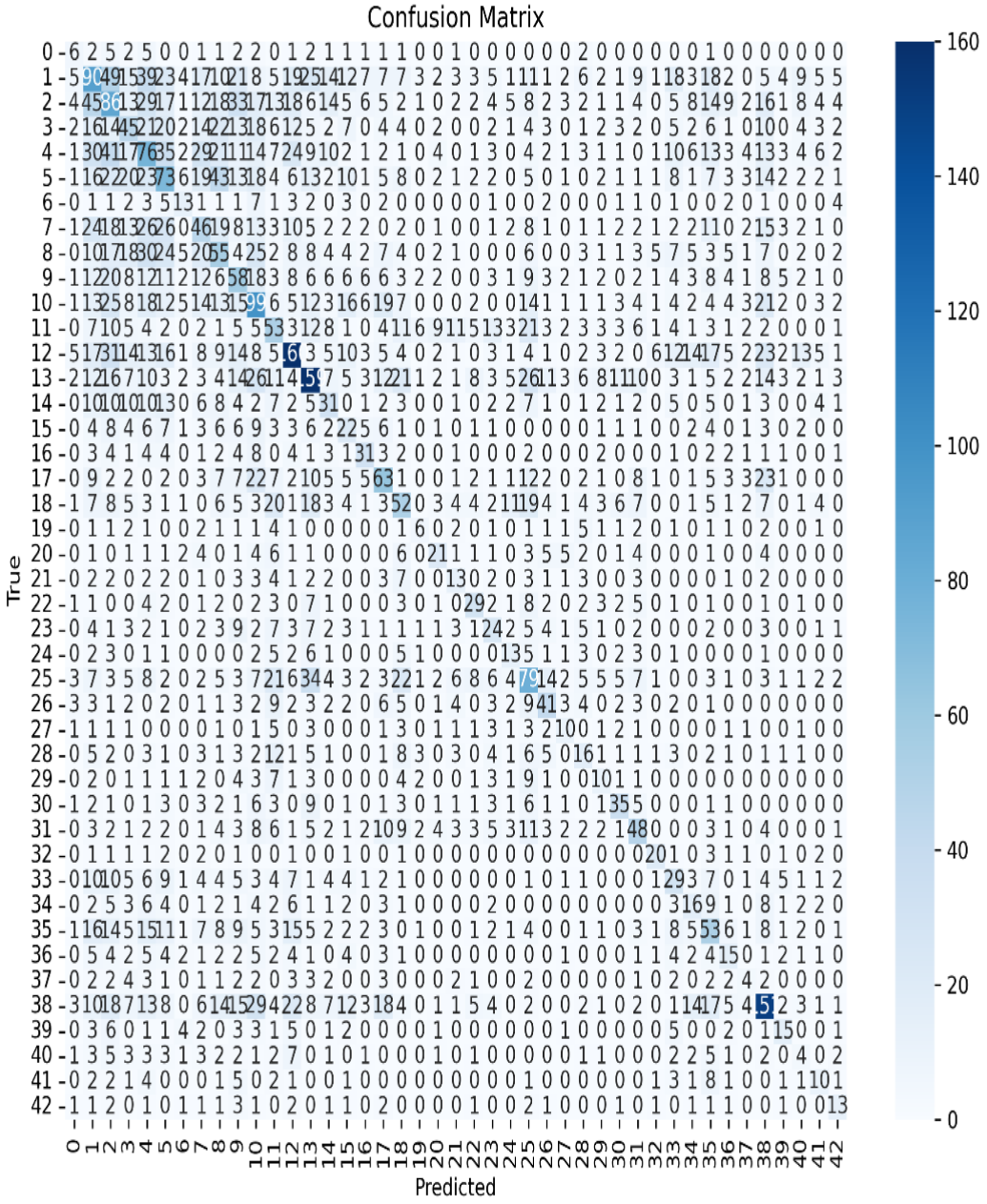


Figure 6.35 shows the confusion matrix for LD-HOG and Decision Tree

CHAPTER SEVEN

SUMMARY, CONCLUSION AND RECOMMENDATION

This chapter presents the objectives and achievements of the study. The chapter starts by systematically reviewing the research objectives and how the study addresses each objective. The chapter ends gives a conclusion, the contribution of the study, and recommendations for future work.

7.1 Summary of the study

The study was anchored on three theories, feature analysis that had two phases for image classification, pattern recognition theory that introduced a third phase of preprocessing to the feature analysis theory and the machine learning theory that guided the training of the machine learning model. These three phases; preprocessing, feature extraction and classification were simulated and experimented on with the aim of developing an enhanced machine learning model for traffic sign recognition. Guided by three objectives; analyze literature view on traffic sign recognition, design a robust feature extractor for traffic sign recognition and develop a machine learning model for traffic sign recognition, this section discusses the key finding of these objectives.

7.1.2 Analysis of literature review on traffic sign recognition

The section summaries the findings of objective one

7.1.2.1 Image Preprocessing analysis

According to patten recognition theory, the first stage is image preprocessing. Outlining their strength, weakness and computational cost in terms of big notation, the study evaluated six image preprocessing techniques namely; interpolation, cropping, color

conversion, normalization filtering and segmentation. Experiments were performed on five of these techniques (interpolation, cropping, filtering, normalization and segmentation with the intent of choosing the best inputs for model training. Color conversation was not experimented on because the study applied gray scale images. These techniques were assessed in the context of traffic sign recognition specifically on the GTSRB dataset

Five commonly used interpolation methods (bilinear, nearest neighbor, bicubic, Lanczos resampling, and content-aware scaling) were compared and analyzed for their strength and weakness as shown in table 4.1. The following points were noted from literature review

i). Bilinear and nearest neighbor had the lowest time complexity of O(n), making them the most efficient.

ii). Bilinear and nearest neighbor performed poorly in maintaining sharp edges, with bilinear being the worst.

iii). Content-aware scaling, though computationally intensive, preserved edge quality best.

The study emphasized the trade-off between computational efficiency and image detail preservation, suggesting content-aware scaling due to the nature and need of getting right the task of traffic sign recognition. This choice was supported by experiments

The study considered and analyzed three cropping techniques (aspect ratio adjustment, region of interest (ROI) (O(n)), and content-based image retrieval (CBIR)). The following were the main findings from literature review on cropping techniques:

i). Aspect ratio adjustment was the efficient in terms of time complexity with a constant time big O notation i.e. O (1) while ROI and CBIR had linear time of O(n).

ii). Aspect ratio was prone to distortion

iii). ROI balanced efficiency using moderate resources while CBIR required a lot of resources.

iv). CBIR was the most accurate in preserving image features followed by ROI then Aspect ratio

Due to the nature of TSR and experiment results, CBIR was selected for model development

Conversion of the traffic sign image color scheme was important as well. Conversion to HSV and Lab was faster and preserved color detail. HSV was computationally cheaper in terms of average time complexity with a constant time 0(1) while grayscale conversion had O(n) time complexity, Nonetheless, gray scale discarded color information which played an important role of reducing workload during training of the model. Finally, the Y'CbCr scheme preserved grayscale details within the Y' channel. However, it was computationally intensive and requires a lot of resources. The choice of color space depended on task requirements and computational limits. The study converted images to gray scale because the texture of the image was enough for training a good model and helped in reducing the resource required.

Four image filtering (deep learning, frequency domain linear and nonlinear) methods were considered and analyzed. Despite being computationally expensive with a big O of $O(n^3)$, deep learning-based filters offered the most adaptive performance. Deep learning also required an annotated dataset. On the other hand, frequency domain filtering strongest attribute was the sharpening and blurring of traffic sign image. Nonetheless, frequency domain required more processing caused by increased overhead. Notably, nonlinear filters

had the ability of introducing artifacts but were in a better position to handle complex noise as compared to linear filters. Also, linear filters were the fastest and required minimal resources, The study experiment on three of these filters (frequency domain, linear and nonlinear filters). From experimental results frequency domain was selected due to its combined score.

The next image preprocessing technique to be analyzed is normalization. Normalization techniques that were analyzed are spatial normalization, contrast stretching, histogram equalization, and deep learning-based approaches. Notably, Spatial normalization corrected image distortion but required a high computational power with a time complexity of (O(n)). Among the normalization contrast stretching with a big O notation of (O(n)) was efficient and had been widely used in different applications and task. Histogram equalization had the ability of enhancing detail in both bright and dark areas. Similar to filtering, deep learning-based methods offered adaptive performance at the cost of annotated training data and computational resources. Notably histogram was selected for model development due to its combined score from experiments on GTSRB dataset.

Three image segmentation methods (region-based, edge-based and deep learning) were reviewed and analyzed. Image segmentation methods were categorized into region-based, edge-based, and deep learning. Notably, region-based and edge-based approaches had the same $O(n^2)$ complexity. The study noted that region-based methods worked well in areas with balanced features and performed well on images with noise. Task that required sharp edges recognition preferred edge-based methods. Edge-based methods were ideal for clear boundary detection. Deep learning approaches were adaptable and capable of learning complex patterns but required significant computational resources and annotated images.

7.1.2.2 Feature extraction techniques literature review

The study reviewed and analyzed literature for six feature extractors namely Local Binary Patterns (LBP), Bag of Words (BoW), Histogram of Oriented Gradients (HOG), SIFT, Gabor filters and Local Directional Pattern (LDP). The study noted the following

i). LBP and BoW required the minimal resources among the six with a time complexity of O(n) and were computationally efficient.

ii). HOG captured texture and shape but struggled in cluttered backgrounds.

iii). SIFT was robust but computationally intensive. It required more computational power

iv). Gabor filters were effective for edge detection but required tuning which meant additional computational power

v). LDP was robust but captured pixels for only four images and avoids the other four neighbors. LDP was computationally efficient.

HOG was selected because of its ability to capture texture well which was an important concept for the TSR task. LDP was robust an important attribute for a good TSR model. Additionally, computation of four neighbors was already done and required the remaining four neighbors for LD-HOG

7.1.2.3 Activation Functions and Optimization:

Another important part that was analyzed was the activation functions. ReLU, sigmoid, tanh, and SoftMax functions were reviewed. Except for SoftMax (O(n)), others had O (1) complexity. ReLU performed well in deep networks, sigmoid worked for binary

classification, and SoftMax for multi-class tasks. Optimization techniques like batch and stochastic gradient descent (O(n)) were compared. Stochastic methods showed better convergence for large datasets, while batch methods suited small datasets but required more resources.

Therefore, TSR being a multi-class problem with 43 classes, SoftMax was selected as the activation function. Nonetheless, with over 50,000 images in the GTSRB, stochastic gradient descent over batch gradient descent.

7.1.2.4 Convolution Neural Network Models

The study evaluated the following models AlexNet, VGGNet, ResNet, GoogleNet, Faster R-CNN, SegNet, and MobileNet. The following are the findings of the review

i). Most had O(n) complexity except VGGNet ($O(n^2)$). AlexNet and VGGNet were computationally heavy.

ii). ResNet solved vanishing gradient issues through skip connections.

iii). GoogleNet introduced inception modules for efficient computation.

iv). SegNet, designed for semantic segmentation, required annotated data and was input-sensitive.

v). MobileNet offered lightweight efficiency, suitable for mobile devices, though with a trade-off in accuracy.

In conclusion, the simulation involved analyzing various methods to identify those that balanced computational efficiency, scalability, and task relevance, often requiring a combination of techniques to achieve optimal performance.

7.1.3 LD-HOG Feature Extractor

Two feature extraction methods Local Directional Pattern (LDP) and Histogram of Oriented Gradient (HOG) are introduced in this chapter. The two formed the basis for development of an improved descriptor called Local Directional Histogram of Oriented Gradient (LD-HOG). Notably, LD-HOG enhances feature extraction through gradient encoding by incorporating additional two directional responses, therefore, improving the performance and robustness in texture and shape representation.

7.1.3.1 Local Binary Patterns (LBP)

LBP is a texture descriptor that identifies micro-structures (edges, corners, spots) by comparing pixel intensities in a 3×3 neighborhood. Each pixel threshold is compared against its center, generating an 8-bit binary code converted to decimal. A histogram of these values represents texture patterns. Notably, LBP had the following limitation

i). Sensitive to noise and illumination changes.

ii). Fails to preserve spatial information in larger regions.

iii). Extended LBP variants improve feature encoding but still lack robustness.

7.1.3.2 Local Directional Pattern (LDP)

LDP improves upon LBP by using Kirsch edge detectors to compute 8-directional gradients instead of raw pixel intensities. For each 3×3 region. Figure 5.8 shows how the 8 Kirsch mask are applied to get the directional response. A value for k is selected and the pixels are compared to this value. E.g. k =3 will let bits to 1, otherwise 0. This process repeats across the image and the final output is a binary code that is converted to decimal

to represent the image or part of the image. Notably, LDP performed better that LBP and had the following advantages.

i). LDP had the ability to remain the same when exposed to noise and illumination as illustrated by figure 5.2

ii). LDP maintained pixels for forming the boundaries in an image therefor capturing edges well

Nonetheless, LDP had the following limitation:

i). LDP didn't store non-top responses leading to loss of potentially useful data.

ii). LDP had the ability to over-represent responses of orientations mainly edges

7.1.3.3 Histogram of Oriented Gradients

The other feature extractor that was used as a basis for LD-HOG was Histogram od orientated gradient. Notably, HOG is important because it computes gradient magnitude and orientation in localized regions. HOG applies 1D derivative masks [-1, 0, +1] to calculate the horizontal and vertical gradients. The two gradients are used to calculate the magnitude and orientation using equation 5.17. Finally, the bin gradients create a 9- bin histogram between 0^o and 180^o separates at 20^o for each 8 by 8 cell. The study noted the following limitation of the HOG feature extractor.

i). HOG worked on the vertical and horizontal gradients and ignored the diagonal structures.

ii). HOG performed well on high-dimensional feature vectors but struggled with low dimensional features.

iii). HOG didn't consider the direction of the gradient.

7.1.3.4 Local Directional Histogram of Oriented Gradient

Based on HOG, LD-HOG extends HOG by introducing additional four directional gradients (0°, 45°, 90°, 135°) and rotating the HOG 1D mask filters. In LD-HOG, four gradients are calculated Horizontal (g_x), vertical (g_γ), left-diagonal (g_{le}), right-diagonal (g_{re}). The horizontal and vertical are combined and the diagonals also combined to form two magnitude and two orientations ($g_{x\gamma}$, $\theta_{x\gamma}$ (from g_x and g_γ) and g_{lr}, θ_{lr} (from g_{le} and g_{re})). These two magnitude and orientation are used to create two histograms that are fused together via max-pooling that selects the dominant orientation per bin. Normalization is done using L2-Norm with the aim of reducing dimensionality. From the study, the following are the advantages of LD-HOG

i). LD-HOG captures diagonal structures that are left out by HOG.

ii). LD-HOG is more discriminative for texture and shape than HOG and LDP.

iii). LD-HOG Outperforms HOG and LDP in traffic sign recognition tested on GTSRB dataset.

7.1.4 Machine learning model for traffic sign recognition summary

This section covers summary on the dataset, classification and regression, experimental results and validation.

7.1.4.1 Germany Traffic Sign Recognition Benchmark dataset

The study employed the German Traffic Sign Detection Benchmark (GTSRB), a widely recognized open-source dataset introduced in 2012 for traffic sign recognition (TSR) research. The dataset comprises 51,840 images spanning 43 distinct traffic sign classes, divided into:

i). Training set: 39,209 images

ii). Testing set: 12,631 images

GTSRB captures real-world driving conditions, including variations in brightness, occlusions, sizes and geographical diversity as shown in figure 6.1. The following was noted from the dataset:

i). Imbalanced representation that was addressed via stratified sampling to prevent classifier bias.

ii). The images were divided into Circular (prohibitory), triangular (warning), rectangular (informational).

iii). The dominant colors were red (prohibitory), blue (mandatory), yellow (temporary hazards)

7.1.4.2 Classification and Regression

TSR was a multiclass classification problem (43 classes). Three classifiers were evaluated: Support Vector Machine (SVM) with the following inputs

i). Kernel: Radial Basis Function (RBF) for handling non-linear features.

ii). Hyperparameters:

- C=10 (regularization: balances margin width and overfitting).
- gamma='scale' (auto-adjusts kernel coefficient).
- class weight='balanced' (mitigates class imbalance).

Random Forest (RF) that is an ensemble method that is robust to noise while decision Trees (DT) that is simpler, but prone to overfitting.

7.1.4.3 Experimental Results

The following are the summaries of the results from experiments carried out to create the model:

i). Lanczos4 selected for best sharpness, gradient retention, and SSIM (Figures 6.5–6.9).

ii). Aspect Ratio Cropping (89% score) outperformed Content-Based (CBIR) and Region-of-Interest (ROI) methods in edge preservation and processing time (Figures 6.10–6.15).

iii). Frequency Domain Filters excelled in denoising and gradient retention but were slower than linear methods (Figures 6.16–6.21).

iv). Histogram Equalization (86% score) enhanced contrast and uniformity (Figures 6.22–6.27).

v). Region-Based Methods (51.5% score) edged out edge-based approaches in accuracy and SSIM (Table 6.1).

vi). SVM and LD-HOG: Achieved 99% precision, recall, F1-score (Table 6.2, Figure 6.28).

vii). Random Forest and LD-HOG achieved 95% F1-score (Table 6.3).

viii). Decision Trees and LD-HOG had 76% F1-score (Table 6.4).

Notably, LD-HOG consistently outperformed HOG and LDP across all classifiers (Table 6.5, Figure 6.31).

7.1.4.4 Model Validation

In order to validate and generalize the performance (precision, recall and F1-score) of the models developed using LD-HOG Stratified K-Fold validation was performed**.** With 5 folds, the results of the validation experiment are summarized in table 6.6. Notably, the following are the summary of validation results:

i). SVM had an average of 98.8% for F1-score and a standard deviation of (σ=0.18).

ii). Random Forest had an average of 94.8% for FI=score with a standard deviation of (σ=0.25).

iii). Decision Trees achieved an F1-score of 76.5% and a standard deviation of (σ=0.17).

Additionally, the confusion matrix was analyzed for the three models in order to validate the results. The following was noted from this analysis

i). SVM's high diagonal values confirm minimal misclassifications.

ii). Rare misclassifications occurred between visually similar signs for example 30 km/h speed limit was easily mis classified with 50 km/h speed limits.

iii). Confusion matrix for random forest and decision tree confirmed the performance parameter scores for precision, recall and F1-score.

7.2 Conclusion

GTSRB's real-world complexity validated LD-HOG's superiority over HOG and LDP, particularly with SVM. Preprocessing choices Lanczos4, aspect ratio cropping, histogram equalization further enhanced performance. The 99% F1-score demonstrates LD-HOG's robustness for TSR in variable conditions.

7.3 Recommendation for further studies

The study recommends the following study areas:

i). Extend testing to datasets with more extreme occlusions

ii). Similar study based on space complexity

iii). Similar study based on Big Omega and Big Theta

iv). Image classification using same approach for different images

REFERENCES

[1] WHO, "Global Status Report on Road Safety," World Health Organization, 2018.

[2] Y. Saadna and A. Behloul, "An overview of traffic sign detection and classification methods," International Journal of Multimedia and Information Retrieval, 2017.

[3] N. Youssouf, "Traffic sign classification using CNN and detection using faster-RCNN and YOLOV4," Heliyon, pp. 1-8, 2022.

[4] A. Saouli, M. El Aroussi and Y. Fakhri, "Traffic sign recognition based On multi-feature fusion and ELM classifier', Procedia Comput. Sci., 2018, 127," in Proceedings on Computer Science Conference, 2018.

[5] A. Jayaprakasha and C. KeziSelvaVijilab, "Feature selection using ant colony optimization (ACO) and road sign detection and recognition (RSDR) system," Cogn. Syst. Res., vol. 58, pp. 123-133, 2019.

[6] Y. Yang, H. Luo, H. Xu and e. al., "Towards real-time traffic sign detection and classification," IEEE Trans. Intell. Transp. Syst., vol. 17, no. 7, pp. 2022-2031, 2016.

[7] A. Hechri, R. Hmida, A. Abdelali and e. al., "Real time road lane markers detection for intelligent vehicles," Adv. Environ. Biol., vol. 8, no. 7, pp. 2266-2272, 2014.

[8] A. Maletzky, N. Hofer, S. Thumfart, K. Bruckmüller and J. Kasper, "Traffic Sign Detection and Classification on the Austrian Highway Traffic Sign Data Set.," Data, vol. 8, no. 16, pp. 1-15, 2023.

[9] J. Shen, C. Zhang, Y. Zheng, and R. Wang, "Decision-Level Fusion with a Pluginable Importance Factor Generator for Remote Sensing Image Image

Classification," *Remote Sensing*, vol. 13, no. 18, p. 3579, Sep. 2021, doi: https://doi.org/10.3390/rs13183579

[10] M. Graumann, C. Ciuffi, and R. M. Cichy, "Image Clutter and Attention Differentially Affect Object Category and Location Representations," *Journal of Vision*, vol. 19, no. 10, p. 171a, Sep. 2019, doi: https://doi.org/10.1167/19.10.171a.

[11] V. Balali, A. Ashouri Rad, and M. Golparvar-Fard, "Detection, classification, and mapping of U.S. traffic signs using google street view images for roadway inventory management," *Visualization in Engineering*, vol. 3, no. 1, Nov. 2015, doi: https://doi.org/10.1186/s40327-015-0027-1.

[12] H. Guan, W. Yan, Y. Yu, L. Zhong, and D. Li, "Robust Traffic-Sign Detection and Classification Using Mobile LiDAR Data With Digital Images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 5, pp. 1715–1724, May 2018, doi: https://doi.org/10.1109/jstars.2018.2810143.

[13] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-Sign Detection and Classification in the Wild," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, doi: https://doi.org/10.1109/cvpr.2016.232.

[14] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, Oct. 2005, doi: https://doi.org/10.1109/tpami.2005.188.

[15] B. Sadeghi, K. Jamshidi, A. Vafaei, and S. A. Monadjemi, "2DIGH: a polar invariant local image descriptor based on joint histogram," *The Visual Computer*, vol. 34, no. 11, pp. 1579–1595, Sep. 2017, doi: https://doi.org/10.1007/s00371-017-1433-2.

[16] F. P. S. Luus, B. P. Salmon, F. van den Bergh, and B. T. J. Maharaj, “Multiview Deep Learning for Land-Use Classification,” *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 12, pp. 2448–2452, Dec. 2015, doi: https://doi.org/10.1109/lgrs.2015.2483680.

[17] Park, “Frequency-Cepstral Features for Bag of Words Based Acoustic Context Awareness,” *The Journal Of The Acoustical Society Of Korea*, vol. 33, no. 4, p. 248, 2014, doi: https://doi.org/10.7776/ask.2014.33.4.248.

[18] S. Zhao, “Image Bag Generator Based on Bag of Visual Words,” *Journal of Information and Computational Science*, vol. 10, no. 5, pp. 1453–1462, Mar. 2013, doi: https://doi.org/10.12733/jics20101532.

[19] Kuo, W. J., & Chiu, C. C. (2012). Content-based image retrieval with bag of visual words. Journal of information science and engineering, 28(3), 655-668.

[20] Nanni, L., & Lumini, A. (2012). Local binary patterns variants as texture descriptors for medical image analysis. Artificial intelligence in medicine, 56(1), 25-35.

[21] Peng, F., Qian, L., Yang, J., Zhang, C., & Zuo, W. (2014). Feature coding in image classification: A comprehensive study. IEEE Transactions on Multimedia, 16(11), 2828-2840.

[22] Zhang, Z., Gao, S., & Zhao, B. (2013). Multi-scale local binary pattern for texture classification. Journal of Computational Information Systems, 9(7), 2561-2568.

[23] T. Ojala, M. Pietikäinen, and D. Harwood, "A Comparative Study of Texture Measures with Classification Based on Feature Distributions," Pattern Recognition, vol. 29, no. 1, pp. 51-59, 1996. DOI: 10.1016/0031-3203(95)00067-4.

[24] Banik, S., Rangaraj Rangayyan, & J.E. Leo Desautels. (2022). *Computer-Aided Detection of Architectural Distortion in Prior Mammograms of Interval Cancer*. Springer Nature.

[25] Singh, B. (2016). Iris Recognition Using Curve let Transformation Based on Gabor Filter& SVM. *International Journal of Engineering and Computer Science*. https://doi.org/10.18535/ijecs/v5i8.32

[26] John, V., Boyali, A., & Mita, S. (2017). Gabor Filter and Gershgorin Disk-based Convolutional Filter Constraining for Image Classification. *International Journal of Machine Learning and Computing*, *7*(4), 55–60. https://doi.org/10.18178/ijmlc.2017.7.4.620

[27] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

[28] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural networks, 61, 85-117.

[29] ANN (2020)

[30] Graves, A., Mohamed, A. R., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In Acoustics, speech and signal processing (icassp), 2013 ieee international conference on (pp. 6645-6649). IEEE.

[31] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence, 35(8), 1798-1828.

[32] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[33] Girshick, R. (2015). Fast R-CNN. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

[34] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[35] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[36] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536. https://doi.org/10.1038/323533a0

[37] Christian, S., & Russakovsky, O. (2015). Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385.

[38] Chahar, V. (2012). Analysis of Back Propagation Algorithm. *International Journal of Scientific Research*, *2*(8), 305–306. https://doi.org/10.15373/22778179/aug2013/98

[39] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. Proceedings of COMPSTAT, 177-186. https://doi.org/10.1007/978-3-7908-2604-3_16

[40] Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. Proceedings of the International Conference on Artificial Intelligence and Statistics,75 249-256. https://doi.org/10.1162/153244303322533223

[41] Hamed Habibi Aghdam, Elnaz Jahani Heravi, & Springer International Publishing Ag. (2018). *Guide to Convolutional Neural Networks A Practical Application to Traffic-Sign Detection and Classification*. Cham Springer International Publishing Springer.

[42] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37).

[43] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3431-3440).

[44] Shelhamer, E., Long, J., & Darrell, T. (2017). Fully convolutional networks for semantic segmentation. IEEE transactions on pattern analysis and machine intelligence, 39(4), 640-651.

[45] Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

[46] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2921-2929).

[47] Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. arXiv preprint arXiv:1605.07146

[48] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[49] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[50] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." International Conference on Machine Learning. PMLR, 2015.

[51] Santurkar, Shibani, et al. "How does batch normalization help optimization?." Advances in Neural Information Processing Systems. 2018.

[52] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[53] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).

[54] Duda, R. O., Hart, P. E., & Stork, D. G. (2012). Pattern classification. John Wiley & Sons.

[55] Liu, Z., Zhang, Y., & Wang, Y. (2019). Tiling and Memory Optimization for Convolutional Neural Network. IEEE Access, 7, 26708-26715.

[56] Lin, Z., Yang, H., & Wei, S. E. (2019). Efficient Convolutional Neural Network Tiling for FPGA Acceleration. In Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (pp. 237-246).

[57] Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., ... & Wu, W. (2015). MXNet: a flexible and efficient machine learning library for heterogeneous distributed systems. arXiv preprint arXiv:1512.01274.

[58] Vasilache, N., Johnson, J., Mathieu, M., Chintala, S., Piantino, S., & LeCun, Y. (2014). Fast convolutional nets with fbfft: A GPU performance evaluation. arXiv preprint arXiv:1412.7580.

[59] Lavin, A., & Gray, S. (2016). Fast algorithms for convolutional neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(10), 1943-1955.

[60] Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 6848-6856).

[61] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[62] Y. Yuan, Z. Xiong, and Q. Wang, "VSSA-NET: Vertical Spatial Sequence Attention Network for Traffic Sign Detection," vol. 28, no. 7, pp. 3423–3434, Jul. 2019, doi: https://doi.org/10.1109/tip.2019.2896952.

[63] Simplilearn, "What is Image Processing: Overview, Applications, Benefits, and Who Should Learn It," *Simplilearn.com*, Apr. 29, 2021. https://www.simplilearn.com/image-processing-article

[64] Jyotismita Chaki and N. Dey, *A Beginner's Guide to Image Preprocessing Techniques*. CRC Press, 2018.

[65] R. Lukac and K. N. Plataniotis, *Color Image Processing*. CRC Press, 2018.

[66] A. Kumar Srivastav, "Interpolation," *WallStreetMojo*, Jan. 25, 2020. https://www.wallstreetmojo.com/interpolation/

[67] Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information theory, 13(1), 21-27.

[68] Simplilearn, "What is Bilinear Interpolation? | Simplilearn," *Simplilearn.com*, Dec. 08, 2022.https://www.simplilearn.com/tutorials/statistics-tutorial/bilinear-interpolation (accessed Jun. 20, 2023).

[69] Kang & Atul, "Image Processing – Bilinear Interpolation," *TheAILearner*, Dec. 29, 2018. https://theailearner.com/2018/12/29/image-processing-bilinear-interpolation/

[70] D. Rowe and D. Rowe, "BiLinear, Bicubic, and In Between Spline Interpolation MU MSCS Spring 2018," 2018. Accessed: Jun. 20, 2023. [Online]. Available: https://www.mssc.mu.edu/~daniel/pubs/RoweTalkMSCS_BiCubic.pdf

[71] McMaster, “Bicubic Interpolation,” 2014. Available: https://www.ece.mcmaster.ca/~xwu/interp_1.pdf

[72] F. Mazzoli, “Lánczos interpolation explained,” *mazzo.li*. https://mazzo.li/posts/lanczos.html (accessed Jun. 20, 2023).

[73] Intel, “Resize with Lanczos Interpolation oneIPL Specification documentation,” *spec.oneapi.io*. https://spec.oneapi.io/oneipl/0.5/transform/resize_lanczos.html (accessed Jun. 20, 2023).

[74] B. Zarouali, S. C. Boerman, and C. H. de Vreese, “Is this recommended by an algorithm? The development and validation of the algorithmic media content awareness scale (AMCA-scale),” *Telematics and Informatics*, vol. 62, p. 101607, Sep. 2021, doi: https://doi.org/10.1016/j.tele.2021.101607.

[75] “Content Aware Scaling,” *Know Your Meme*, Nov. 06, 2013. https://knowyourmeme.com/memes/content-aware-scaling (accessed Jun. 20, 2023).

[76] S.-S. Lin, I-Cheng. Yeh, C.-H. Lin, and T.-Y. Lee, “Patch-Based Image Warping for Content-Aware Retargeting,” *IEEE Transactions on Multimedia*, vol. 15, no. 2, pp. 359–368, Feb. 2013, doi: https://doi.org/10.1109/tmm.2012.2228475.

[77] M. T. Brett, “Region of interest analysis using an SPM toolbox,” vol. 16, p. 497, Jan. 2002.

[78] R. Takahashi, T. Matsubara, and K. Uehara, “Data Augmentation using Random Image Cropping and Patching for Deep CNNs,” *IEEE Transactions on Circuits and*

Systems for Video Technology, pp. 1–1, 2019, doi: https://doi.org/10.1109/tcsvt.2019.2935128.

[79] R. Eswaraiah and E. Sreenivasa Reddy, "Robust medical image watermarking technique for accurate detection of tampers inside region of interest and recovering original region of interest," *IET Image Processing*, vol. 9, no. 8, pp. 615–625, Aug. 2015, doi: https://doi.org/10.1049/iet-ipr.2014.0986.

[80] J. Chen, G. Bai, S. Liang, and Z. Li, "Automatic Image Cropping: A Computational Complexity Study," Jun. 2016, doi: https://doi.org/10.1109/cvpr.2016.61.

[81] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, doi: https://doi.org/10.1109/cvpr.2015.7298682.

[82] C. Szegedy *et al.*, "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015, doi: https://doi.org/10.1109/cvpr.2015.7298594.

[83] K. B. Shaik, P. Ganesan, V. Kalist, B. S. Sathish, and J. M. M. Jenitha, "Comparative Study of Skin Color Detection and Segmentation in HSV and YCbCr Color Space," *Procedia Computer Science*, vol. 57, pp. 41–48, 2015, doi: https://doi.org/10.1016/j.procs.2015.07.362.

[84] Northwestern, "Color (Image Processing Toolbox)," *www.ece.northwestern.edu*. http://www.ece.northwestern.edu/local-apps/matlabhelp/toolbox/images/color11.html#:~:text=The%20HSV%20color%20space%20(hue

[85] R. C. Gonzalez and R. E. Woods, *Digital image processing*. New York, Ny: Pearson, 2018.

[86] M. Singh, P. K. Gupta, V. Tyagi, J. Flusser, and Tuncer Ören, *Advances in Computing and Data Sciences*. Springer, 2018.

[87] R. Kasturi, *Image Analysis Applications*. CRC Press, 2020.

[88] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "Semantic Image Synthesis With Spatially-Adaptive Normalization," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, doi: https://doi.org/10.1109/cvpr.2019.00244.

[89] N. Shrivastava and J. Bharti, "Automatic Seeded Region Growing Image Segmentation for Medical Image Segmentation: A Brief Review," *International Journal of Image and Graphics*, vol. 20, no. 03, p. 2050018, Jul. 2020, doi: https://doi.org/10.1142/s0219467820500187.

[90] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," *2014 Science and Information Conference*, Aug. 2014, doi: https://doi.org/10.1109/sai.2014.6918213.

[91] R. Azhar, D. Tuwohingide, D. Kamudi, Sarimuddin, and N. Suciati, "Batik Image Classification Using SIFT Feature Extraction, Bag of Features and Support Vector Machine," *Procedia Computer Science*, vol. 72, pp. 24–30, 2015, doi: https://doi.org/10.1016/j.procs.2015.12.101.

[92] A. Omid-Zohoor, C. Young, D. Ta, and B. Murmann, "Toward Always-On Mobile Object Detection: Energy Versus Performance Tradeoffs for Embedded HOG

Feature Extraction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 5, pp. 1102–1115, May 2018, doi: https://doi.org/10.1109/TCSVT.2017.2653187.

[93] A. K. Jaiswal and H. Banka, "Local pattern transformation based feature extraction techniques for classification of epileptic EEG signals," *Biomedical Signal Processing and Control*, vol. 34, pp. 81–92, Apr. 2017, doi: https://doi.org/10.1016/j.bspc.2017.01.005.

[94] D. Sugimura, T. Fujimura, and T. Hamamoto, "Enhanced Cascading Classifier Using Multi-Scale HOG for Pedestrian Detection from Aerial Images," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 30, no. 03, p. 1655009, Feb. 2016, doi: https://doi.org/10.1142/s0218001416550090.

[95] H. Wang and C. Schmid, "Action Recognition with Improved Trajectories," *2013 IEEE International Conference on Computer Vision*, Dec. 2013, doi: https://doi.org/10.1109/iccv.2013.441.

[96] Y. Yang, H. Luo, H. Xu, and F. Wu, "Towards real-time traffic sign detection and classification," IEEE Transactions on Intelligent Transportation Systems, vol. 17, pp. 2022–2031, July 2016.

[97] S. Salti, A. Petrelli, F. Tombari, N. Fioraio, and L. D. Stefano, "Traffic sign detection via interest region extraction," Pattern Recognition, vol. 48, no. 4, pp. 1039–1049, 2015.

[98] N. Barnes, A. Zelinsky, and L. S. Fletcher, “Real-time speed sign detection using the radial symmetry detector,” IEEE Transactions on Intelligent Transportation Systems, vol. 9, pp. 322–332, June 2008.

[99] R. Belaroussi and J. P. Tarel, “Angle vertex and bisector geometric model for triangular road sign detection,” in Workshop on Applications of Computer Vision (WACV), pp. 1–7, Dec 2009.

[100] H. Li, F. Sun, L. Liu, and L. Wang, “A novel traffic sign detection method via color segmentation and robust shape matching,” Neurocomputing, vol. 169, pp. 77–88, 2015.

[101] H. H. Aghdam, E. J. Heravi, and D. Puig, “A practical approach for detection and classification of traffic signs using convolutional neural networks,” Robotics and Autonomous Systems, vol. 84, pp. 97–112, 2016.

[102] X. Baro, S. Escalera, J. Vitria, O. Pujol, and P. Radeva, “Traffic sign recognition using evolutionary adaboost detection and forest-ecoc classification,” IEEE Transactions on Intelligent Transportation Systems, vol. 10, pp. 113–126, March 2009.

[103] C. Liu, F. Chang, and Z. Chen, “Rapid multiclass traffic sign detection in high resolution images,” IEEE Transactions on Intelligent Transportation Systems, vol. 15,pp. 2394–2403, Dec 2014.

[104] A. Mogelmose, D. Liu, and M. M. Trivedi, “Detection of u.s. traffic signs,” IEEE Transactions on Intelligent Transportation Systems, vol. 16, pp. 3116–3125, Dec 2015.

[105] T. Chen and S. Lu, “Accurate and efficient traffic sign detection using discriminative adaboost and support vector regression,” IEEE Transactions on Vehicular Technology, vol. 65, pp. 4006–4015, June 2016.

[106] Y. Zhu, C. Zhang, D. Zhou, X. Wang, X. Bai, and W. Liu, “Traffic sign detection and recognition using fully convolutional network guided proposals,” Neurocomputing, vol. 214, pp. 758–766, 2016.

[107] M. Shi, H. Wu, and H. Fleyeh, “Support vector machines for traffic signs recognition,” in IEEE International Joint Conference on Neural Networks, pp. 3820–3827, June 2008.

[108] B. Hoferlin and K. Zimmermann, “Towards reliable traffic sign recognition,” in IEEE Intelligent Vehicles Symposium, pp. 324–329, June 2009.

[109] W. J. Kuo and C. C. Lin, “Two-stage road sign detection and recognition,” in IEEE International Conference on Multimedia and Expo, pp. 1427–1430, July 2007.

[110] A. Ruta, Y. Li, and X. Liu, “Robust class similarity measure for traffic sign recognition,” IEEE Transactions on Intelligent Transportation Systems, vol. 11, pp. 846–855, Dec2010.

[111] K. Lu, Z. Ding, and S. Ge, “Sparse-representation-based graph embedding for traffic sign recognition,” IEEE Transactions on Intelligent Transportation Systems, vol. 13, pp. 1515–1524, Dec 2012.

[112] X. Lu, Y. Wang, X. Zhou, Z. Zhang, and Z. Ling, “Traffic sign recognition via multi modal tree-structure embedded multi-task learning,” IEEE Transactions on Intelligent Transportation Systems, vol. 18, pp. 960–972, April 2017.

[113] P. Comelli, P. Ferragina, M. N. Granieri, and F. Stabile, “Optical recognition of motor vehicle license plates,” IEEE Transactions on Vehicular Technology, vol. 44, pp. 790–799, Nov 1995.

[114] R. Zunino and S. Rovetta, “Vector quantization for license-plate location and image coding,” IEEE Transactions on Industrial Electronics, vol. 47, pp. 159–167, Feb 2000.

[115] T. Naito, T. Tsukada, K. Yamada, K. Kozuka, and S. Yamamoto, “Robust license plate recognition method for passing vehicles under outside environment,” IEEE Transactions on Vehicular Technology, vol. 49, pp. 2309–2319, Nov 2000.

[116] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, V. Loumos, and E. Kayafas, “A license plate-recognition algorithm for intelligent transportation system applications,” IEEE Transactions on Intelligent Transportation Systems, vol. 7, pp. 377–392, Sept 2006.

[117] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas, V. Loumos, and E. Kayafas, “License plate recognition from still images and video sequences: A survey,” IEEE Transactions on Intelligent Transportation Systems, vol. 9, pp. 377–391, Sept 2008.

[118] C. N. E. Anagnostopoulos, “License plate recognition: A brief tutorial,” IEEE Intelli gent Transportation Systems Magazine, vol. 6, pp. 59–67, Spring 2014.

[119] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, “Automatic license plate recognition (alpr): A state-of-the-art review,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, pp. 311–325, Feb 2013.

[120] Z. X. Chen, C. Y. Liu, F. L. Chang, and G. Y. Wang, "Automatic license-plate location and recognition based on feature salience," IEEE Transactions on Vehicular Technology, vol. 58, pp. 3781–3785, Sept 2009.

[121] Y. Wen, Y. Lu, J. Yan, Z. Zhou, K. M. von Deneen, and P. Shi, "An algorithm for license plate recognition applied to intelligent transportation system," IEEE Transactions on Intelligent Transportation Systems, vol. 12, pp. 830–845, Sept 2011.

[122] W. Zhou, H. Li, Y. Lu, and Q. Tian, "Principal visual word discovery for automatic license plate detection," IEEE Transactions on Image Processing, vol. 21, pp. 4269–4279, Sept 2012.

[123] S.-L. Chang, L.-S. Chen, Y.-C. Chung, and S.-W. Chen, "Automatic license plate recognition," IEEE Transactions on Intelligent Transportation Systems, vol. 5, pp. 42–53, March 2004.

[124] H.-J. Lee, S.-Y. Chen, and S.-Z. Wang, "Extraction and recognition of license plates of motorcycles and vehicles on highways," in International Conference on Pattern Recognition, vol. 4, pp. 356–359, Aug 2004.

[125] B. F. Wu, S. P. Lin, and C. C. Chiu, "Extracting characters from real vehicle licence plates out-of-doors," IET Computer Vision, vol. 1, pp. 2–10, March 2007.

[126] J. M. Guo and Y. F. Liu, "License plate localization and character segmentation with feedback self-learning and hybrid binarization techniques," IEEE Transactions on Vehicular Technology, vol. 57, pp. 1417–1424, May 2008.

[127] W. Jia, H. Zhang, X. He, and M. Piccardi, "Mean shift for accurate license plate localization," in IEEE Intelligent Transportation Systems, pp. 566–571, Sept 2005.

[128] A. Hechri and A. Mtibaa, “Two-stage traffic sign detection and recognition based on svm and convolutional neural networks,” IET Image Processing, vol. 14, no. 4, pp. 342–350, 2020, received on 29th May 2019, Revised 26th October 2019, Accepted on 23rd December 2019, E-First on 10th March 2020. [Online]. Available: https://www.ietdl.org

[129] A. J, G. R, A. K, and S. R, “Traffic sign detection using hog and glcm with decision tree and random forest,” in Proceedings of the 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS). IEEE, 2022, pp. 879–885, authorized licensed use limited to: Masinde Muliro Univ of Science and Tech. Downloaded on April 02, 2025 at 09:13:15 UTC from IEEE Xplore. Restrictions apply.

[130] A. A. Khalifa, W. M. Alayed, H. M. Elbadawy, and R. A. Sadek, “Real-time navigation roads: Lightweight and efficient convolutional neural network (le-cnn) for arabic traffic sign recognition in intelligent transportation systems (its),” Applied Sciences, vol. 14, no. 9, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/14/9/3903

[131] W. Li, H. Song, and P. Wang, “Finely crafted feature features for traffic sign recognition,” International Journal of Circuits, Systems and Signal Processing, vol. 16, pp. 185–193, 2022, received: July 9, 2021; Revised: December 11, 2021; Accepted: January 5, 2022; Published: January 7, 2022.

[132] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, November 1998.

[133] Y. Zhu and W. Q. Yan, “Traffic sign recognition based on deep learning,” Multimedia Tools and Applications, vol. 81, no. 17, pp. 17 779–17 791, 2022, accessed: 2025-04-07. [Online]. Available: https://doi.org/10.1007/s11042-022-12163-0

[134] M. Buda, A. Maki, and M. A. Mazurowski, “A systematic study of the class imbalance problem in convolutional neural networks,” Neural Networks, vol. 106, pp. 249–259, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608018302107

[135] J. Lee, K. Kim, and K. Lee, “Multi-sensor image classification using the random forest algorithm in google earth engine with kompsat-3/5 and cas500-1 images,” Remote Sensing, vol. 16, no. 24, p. 4622, 2024, published 10 December 2024. [Online]. Available: https://doi.org/10.3390/rs16244622

[136] R. K. Halder, M. N. Uddin, M. A. Uddin, S. Aryal, and A. Khraisat, “Enhancing k-nearest neighbor algorithm: a comprehensive review and performance analysis of modifications,” Journal of Big Data, vol. 11, no. 1, p. 113, 2024, published: 11 August 2024. [Online]. Avail- able: https://journalofbigdata.springeropen.com/articles/10.1186/s40537- 024-00956-3

[137] X. Shi, Z. Hao, and Z. Yu, “Spikingresformer: Bridging resnet and vision transformer in spiking neural networks,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2024, pp. 5610–5619.

[138] M. Razavi, S. Mavaddati, and H. Koohi, “Resnet deep models and transfer learning technique for classification and quality detection of rice cultivars,” Expert Systems with Applications, vol. 247, p. 123276, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417424001416

[139] A. Jain, N. R. Moparthi, A. Swathi, Y. K. Sharma, N. Mittal, A. Alhussen, Z. S. Alzamil, and M. A. Haq, “Deep learning-based mask identification system using resnet transfer learning architecture,” Computer Systems Science and Engineering, vol. 47, 2023, received:18 October 2022; Accepted: 07 April 2023. [Online]. Available: https://www.techscience.com/csse/v47n1/53640

[140] C. Chen, B. Li, H. Zhang, M. Zhao, Z. Liang, K. Li, and X. An, “Performance enhancement of deep learning model with attention mechanism and fcn model in flood forecasting,” Journal of Hydrology, vol. 658, p. 133221, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022169425005591

[141] Y. Zhu and S. Newsam, “Densenet for dense flow,” in 2017 IEEE International Conference on Image Processing (ICIP), 2017, pp. 790– 794.

[`142] K. Zhang, Y. Guo, X. Wang, J. Yuan, and Q. Ding, “Multiple feature reweight densenet for image classification,” IEEE Access, vol. 7, pp. 9872–9880, 2019.

[143] A. Kerim and M. Efe, “Recognition of traffic signs with artificial neural networks: A novel dataset and algorithm,” in Proceedings of the 2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC). Jeju Island, Republic of Korea: IEEE, April 2021, pp. 171–176.

[144] D. Soni, R. Chaurasiya, and S. Agrawal, “Improving the classification accuracy of accurate traffic sign detection and recognition system using hog and lbp features and pca-based dimension reduction,” in Proceedings of the International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM). Jaipur, India: Amity University Rajasthan, January 2019.

[145] N. Namyang and S. Phimoltares, “Thai traffic sign classification and recognition system based on histogram of gradients, color layout de- scriptor, and normalized correlation coefficient,” in Proceedings of the 2020 5th International Conference on Information Technology (InCIT), Chonburi, Thailand, October 2020, pp. 270–275.

[146] A. Madani and R. Yusof, “Traffic sign recognition based on color, shape, and pictogram classification using support vector machines,” Neural Computing and Applications, vol. 30, pp. 2807–2817, 2018.

[147] G. Sapijaszko, T. Alobaidi, and W. Mikhael, “Traffic sign recognition based on multilayer perceptron using dwt and dct,” in Proceedings of the 2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS), Dallas, TX, USA, August 2019, pp. 440–443.

[148] S. Aziz and F. Youssef, “Traffic sign recognition based on multi-feature fusion and elm classifier,” Procedia Computer Science, vol. 127, pp. 146–153, 2018.

[149] H. Weng and C. Chiu, “Resource efficient hardware implementation for real-time traffic sign recognition,” in Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, April 2018, pp. 1120–1124.

[150] B. Wang, “Research on the optimal machine learning classifier for traffic signs,” in SHS Web of Conferences. Les Ulis, France: EDP Sciences, 2022

[151] Treisman, A. M., & Gelade, G. (2012). A feature-integration theory of attention. From Perception to Consciousness, 77-96. https://doi.org/10.1093/acprof:osobl/9780199734337.003.0011

[152] Duda, Richard & Hart, Peter & G.Stork, David. (2001). Pattern Classification

[153] Du, Ke-Lin & Zhang, Rengong & Jiang, Bingchun & Zeng, Jie & Lu, Jiabin. (2025). Understanding Machine Learning Principles: Learning, Inference, Generalization, and Computational Learning Theory. Mathematics. 13. 451. 10.3390/math13030451.

[154] M. Saunders, P. Lewis, and A. Thornhill, *Research Methods for Business Students*, 8th ed. Harlow: Pearson, 2019.

[155] Creswell, J. W. (2014). Research design: qualitative, quantitative, and mixed methods approaches. Sage publications.

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

[157] Macal, C. M., & North, M. J. (2010). Tutorial on agent-based modelling and simulation. *Journal of Simulation*, *4*(3), 151–162. https://doi.org/10.1057/jos.2010.3

[158] Z. Sedaghatjoo, H. Hosseinzadeh, and B. S. Bigham, “Local binary pattern(lbp) optimization for feature extraction,” 2024. [Online].

Available: https://arxiv.org/abs/2407.18665

[159] G. Panis, A. Lanitis, N. Tsapatsoulis, and T. F. Cootes, “Overview ofreserach on facial ageing using the fg-net ageing database,” IET Biometrics, vol. 5, pp. 37–46, 2016.

[160] T. Ojala, M. Pietikainen, and D. Harwood, “A comparative study of texture measures with classification based on featured distribution,” Pattern Recognition, vol. 29, pp. 51–59, 1996.

[161] T. Ojala, M. Pietikainen, and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” IEEE Transactions Pattern Analysis and Machine Intelligence, vol. 24, pp. 971–987, 2002.

[162] T. Maenpaa and M. Pietikainen, Texture analysis with local binary pat- terns: Handbook of Pattern Recognition and Computer Vision. World Scientific, 2005.

[163] T. Jabid, M. Kabir, and O. Chae, “Local directional pattern (LDP) for face recognition,” in Proceedings of 2010 Digest of Technical Papers International Conference on Consumer Electronics (ICCE). IEEE, jan 2010.

[164] J. M. S. Prewitt, “Object enhancement and extraction,” in Picture Processing and Psychopictorics. Academic Press, 1970.

[165] R. A. Kirsch, “Computer determination of the constituent structure of biological images,” Computers and Biomedical Research, vol. 4, no. 3, pp. 315–328, jun 1971.

[166] I. Sobel and F. G, “A 3 x 3 isotropic gradient operator for image processing,” in Presented at the Stanford Artificial Intelligence Project (SAIL), 1968.

[167] W. K. Pratt, Digital Image Processing. New York: Wiley, 1978.

[168] S.-W. Lee, “Off-line recognition of totally unconstrained handwritten numerals using multilayer cluster neural network,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, no. 6, pp. 648– 652, jun 1996.

[169] T. Jabid, M. H. Kabir, and O. Chae, “Gender classification using local directional pattern (LDP),” in Proceedings of 2010 20th International Conference on Pattern Recognition. IEEE, aug 2010.

[170] N. Dalal and B. Triggs, “Histograms of oriented gradients for human de- tection,” in in Proceedings of 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2010.

[171] A. Satpathy, X. Jiang, and H.-L. Eng, “Extended histogram of gradients feature for human detection,” in in Proceedings of 2010 17th IEEE Conference on Image Processing (ICIP), 2010.

[172] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in Proceedings of IEEE International Conference on Computer Vision and Pattern (CVPR), 2001, pp. 8–14.

[173] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, 2005, pp. 886–893.

[174] H. Tan, Z. Ma, and B. Yang, “Face recognition based on the fusion of global and local HOG features of face images,” IET Computer Vision, vol. 8, no. 3, pp. 224–234, jun 2014.

[175] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, “2012 special issue: Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition,” Neural Netw., vol. 32, p. 323–332, Aug. 2012. [Online]. Available: https://doi.org/10.1016/j.neunet.2012.02.016

[176] A. Patle and D. S. Chouhan, “Svm kernel functions for classification,” in 2013 International Conference on Advances in Technology and Engineering (ICATE), 2013, pp. 1–9.

APPENDICES

Appendix 1: Loading training images algorithm

```
def load_images_from_folder(folder, label, img_size=(64, 128)):
    data = []
    labels = []
    for filename in os.listdir(folder):
        img_path = os.path.join(folder, filename)
        image = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        if image is not None:
            image = cv2.resize(image, img_size)
            data.append(image)
            labels.append(label)
    return data, labels

# Load data
print("Loading images...")
folders = [
    (r'gtbm\Train\0', 0),
    (r'gtbm\Train\1', 1),
    (r'gtbm\Train\2', 2),
    (r'gtbm\Train\3', 3),
    (r'gtbm\Train\4', 4),
    (r'gtbm\Train\5', 5),
    (r'gtbm\Train\6', 6),
    (r'gtbm\Train\7', 7),
    (r'gtbm\Train\8', 8),
    (r'gtbm\Train\9', 9),
    (r'gtbm\Train\10', 10),
    (r'gtbm\Train\11', 11),
    (r'gtbm\Train\12', 12),
    (r'gtbm\Train\13', 13),
    (r'gtbm\Train\14', 14),
    (r'gtbm\Train\15', 15),
    (r'gtbm\Train\16', 16),
    (r'gtbm\Train\17', 17),
    (r'gtbm\Train\18', 18),
    (r'gtbm\Train\19', 19),
    (r'gtbm\Train\20', 20),
    (r'gtbm\Train\21', 21),
    (r'gtbm\Train\22', 22),
    (r'gtbm\Train\23', 23),
    (r'gtbm\Train\24', 24),
```

```
    (r'gtbm\Train\25', 25),
    (r'gtbm\Train\26', 26),
    (r'gtbm\Train\27', 27),
    (r'gtbm\Train\28', 28),
    (r'gtbm\Train\29', 29),
    (r'gtbm\Train\30', 30),
    (r'gtbm\Train\31', 31),
    (r'gtbm\Train\32', 32),
    (r'gtbm\Train\33', 33),
    (r'gtbm\Train\34', 34),
    (r'gtbm\Train\35', 35),
    (r'gtbm\Train\36', 36),
    (r'gtbm\Train\37', 37),
    (r'gtbm\Train\38', 38),
    (r'gtbm\Train\39', 39),
    (r'gtbm\Train\40', 40),
    (r'gtbm\Train\41', 41),
    (r'gtbm\Train\42', 42)

]

images = []
labels = []
for folder, label in folders:
    imgs, lbls = load_images_from_folder(folder, label)
    images.extend(imgs)
    labels.extend(lbls)
```

Appendix 2: HOG feature extractor Algorithm

```
def extract_hog_features(images):
    hog_features = []
    for image in images:
        features = hog(image, orientations=9, pixels_per_cell=(8, 8),
                       cells_per_block=(2, 2), block_norm='L2-Hys')
        hog_features.append(features)
    return hog_features

features = extract_hog_features(images)
```

Appendix 3: LDP feature extractor

```
import cv2
import numpy as np


# Define Kirsch masks globally
kirsch_masks = [
    np.array([[ 5,  5,  5], [-3,  0, -3], [-3, -3, -3]]),  # N
    np.array([[ 5,  5, -3], [ 5,  0, -3], [-3, -3, -3]]),  # NE
    np.array([[ 5, -3, -3], [ 5,  0, -3], [ 5, -3, -3]]),  # E
    np.array([[-3, -3, -3], [ 5,  0, -3], [ 5,  5, -3]]),  # SE
    np.array([[-3, -3, -3], [-3,  0, -3], [ 5,  5,  5]]),  # S
    np.array([[-3, -3, -3], [-3,  0,  5], [-3,  5,  5]]),  # SW
    np.array([[-3, -3,  5], [-3,  0,  5], [-3, -3,  5]]),  # W
    np.array([[-3,  5,  5], [-3,  0,  5], [-3, -3, -3]])   # NW
]

def compute_ldp(image, top_k=3):
    h, w = image.shape
    responses = np.zeros((8, h, w), dtype=np.int16)

    # Apply Kirsch filters
    for i, mask in enumerate(kirsch_masks):
        responses[i] = cv2.filter2D(image, cv2.CV_16S, mask)

    # Build LDP pattern
    ldp = np.zeros((h, w), dtype=np.uint8)
    for y in range(1, h-1):
        for x in range(1, w-1):
            edge_vals = responses[:, y, x]
            top_indices = np.argsort(edge_vals)[-top_k:]
            code = sum((1 << i) for i in top_indices)
            ldp[y, x] = code

    return ldp

def extract_ldp_features(images, top_k=3, num_bins=256):

    ldp_features = []

    for image in images:
        ldp_map = compute_ldp(image, top_k=top_k)

        # Flatten and build normalized histogram of LDP codes
```

```
        hist, _ = np.histogram(ldp_map.ravel(), bins=num_bins, range=(0,
num_bins))
        hist = hist.astype(np.float32)
        hist /= (hist.sum() + 1e-6)  # Normalize
        ldp_features.append(hist)

    return np.array(ldp_features)
print("Extracting LDP features...")
ldp_features = extract_ldp_features(images)
```

Appendix 4: Designed LD-HOG feature extractor

```
def extract_ld_hog(images, orientations=9, pixels_per_cell=(8, 8),
                   cells_per_block=(2, 2), dominance_radius=3):
    """Enhanced Local Dominant HOG feature extraction"""
    hog_features = []
    for image in images:
        # Gradient computation
        gx = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
        gy = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
        mag = np.sqrt(gx**2 + gy**2)
        ang = np.rad2deg(np.arctan2(gy, gx)) % 180

        # Local dominance enhancement
        dominant_mag = maximum_filter(mag, size=dominance_radius)
        mag[mag < 0.3*dominant_mag] = 0  # Suppress weak gradients

        # Extract HOG with dominant gradients
        features = hog(mag, orientations=orientations,
                       pixels_per_cell=pixels_per_cell,
                       cells_per_block=cells_per_block,
                       block_norm='L2-Hys')
        hog_features.append(features)
    return np.array(hog_features)

# 2. Feature Extraction
print("Extracting LD-HOG features...")
ldfeatures = extract_ld_hog(images)
```

Appendix 5 Support Vector Machine Model for HOG, LD-HOG and LDP

i). Original HOG

```
# Split dataset
X_train, X_test, y_train, y_test = train_test_split(features, labels,
test_size=0.2, random_state=42)

# Train SVM
svm1 = SVC(
    kernel='rbf',
    C=10,
    gamma='scale',
    class_weight='balanced',
    probability=True
)
svm1.fit(X_train, y_train)
```

ii). LDP

```
# Split dataset
X_train, X_test, y_train, y_test = train_test_split(ldfeatures, labels,
test_size=0.2, random_state=42)

# Train SVM
ldsvm = SVC(
    kernel='rbf',
    C=10,
    gamma='scale',
    class_weight='balanced',
    probability=True
)
ldsvm.fit(X_train, y_train)
```

iii). LD-HOG

```
# Split dataset
X_train, X_test, y_train, y_test = train_test_split(ldpfeatures, labels,
test_size=0.2, random_state=42)

# Train SVM
ldpsvm = SVC(
    kernel='rbf',
    C=10,
    gamma='scale',
    class_weight='balanced',
    probability=True
)
ldpsvm.fit(X_train, y_train)
```

Appendix 6 Random Forest model for HOG, LDP and LD-HOG

i). HOG

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(features, labels,
test_size=0.2, random_state=42)

# Train Random Forest
rf = RandomForestClassifier(
    n_estimators=100,          # Number of trees
    max_depth=None,            # Can be set to limit tree depth
    class_weight='balanced',   # Handles imbalanced classes
    random_state=42
)
rf.fit(X_train, y_train)
```

ii). LDP

```
# Split dataset
X_train, X_test, y_train, y_test = train_test_split(ldfeatures, labels,
test_size=0.2, random_state=42)

# Train Random Forest
ldrf = RandomForestClassifier(
    n_estimators=100,          # Number of trees
    max_depth=None,            # Can be set to limit tree depth
    class_weight='balanced',   # Handles imbalanced classes
    random_state=42
)
ldrf.fit(X_train, y_train)
```

iii). LD-HOG

```
X_train, X_test, y_train, y_test = train_test_split(ldpfeatures,
labels, test_size=0.2, random_state=42)

# Train Random Forest
ldprf = RandomForestClassifier(
    n_estimators=100,         # Number of trees
    max_depth=None,           # Can be set to limit tree depth
    class_weight='balanced',  # Handles imbalanced classes
    random_state=42
)
ldprf.fit(X_train, y_train)
```

Appendix 7: Decision tree model for HOG, LDP LD-HOG

i). HOG

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(features, labels,
test_size=0.2, random_state=42)

# Train Decision Tree
dt = DecisionTreeClassifier(
    max_depth=None,           # You can set this to limit overfitting
    class_weight='balanced',  # Handles imbalanced classes
    random_state=42
)
dt.fit(X_train, y_train)
```

ii). LDP

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(ldfeatures, labels,
test_size=0.2, random_state=42)

# Train Decision Tree
lddt = DecisionTreeClassifier(
    max_depth=None,           # You can set this to limit overfitting
    class_weight='balanced',  # Handles imbalanced classes
    random_state=42
)
lddt.fit(X_train, y_train)
```

iii). LD-HOG

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(ldp_features,
labels, test_size=0.2, random_state=42)

# Train Decision Tree
ldpdt = DecisionTreeClassifier(
    max_depth=None,           # You can set this to limit overfitting
    class_weight='balanced',  # Handles imbalanced classes
    random_state=42
)
ldpdt.fit(X_train, y_train)
```

Appendix 8 Model testing for SVM, Random Forest and Decision tree

i). SVM

```
y_pred = ldpsvm.predict(X_test)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix
plt.figure(figsize=(10,8))
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.savefig('confusion_matrix ldpsvm.png', dpi=300)
plt.close()

# 6. Save Model
joblib.dump(ldpsvm, 'traffic_sign_ldpsvm_hog.pkl')
print("\nModel saved to traffic_sign_ldpsvm_hog.pkl")
```

ii). Random Forest

```
y_pred = ldprf.predict(X_test)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix
plt.figure(figsize=(10,8))
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.savefig('confusion_matrix ldprf.png', dpi=300)
plt.close()

# 6. Save Model
joblib.dump(ldprf, 'traffic_sign_ldprf_hog.pkl')
iii).     print("\nModel saved to traffic_sign_ldprf_hog.pkl")
```

iv). Decision Tree

```
y_pred = lddt.predict(X_test)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix
plt.figure(figsize=(10,8))
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.savefig('confusion_matrix lddt.png', dpi=300)
plt.close()

# 6. Save Model
joblib.dump(lddt, 'traffic_sign_lddt_hog.pkl')
print("\nModel saved to traffic_sign_lddt_hog.pkl")
```

Appendix 9: Post Graduate studies Board Approval Letteer

MASINDE MULIRO UNIVERSITY OF SCIENCE AND TECHNOLOGY (MMUST)

Tel: 056-30870
Fax: 056-30153
E-mail: directordps@mmust.ac.ke
Website: www.mmust.ac.ke

P.O Box 190
Kakamega – 50100
Kenya

Directorate of Postgraduate Studies

Ref: MMU/COR: 509099

19th September 2023

Prestone Jeremiah Simiyu
MMUST
P.O. Box 190-50100,
KAKAMEGA.

Dear Mr. Simiyu

RE: APPROVAL OF PROPOSAL

I am pleased to inform you that the Directorate of Postgraduate Studies has considered and approved your PhD proposal entitled: ***"Machine Learning Model for Traffic Sign Recognition"*** and appointed the following as supervisors:

1. Dr. Dr. Raphael Angulu - SCI
2. Dr. Dr. Daniel Otanga - SCI

You are required to submit through your supervisor(s) progress reports every three months to the Director Postgraduate Studies. Such reports should be copied to the following: Chairman, School of Computing and Informatics Graduate Studies Committee and Chairman, Computer Science Department. Kindly adhere to research ethics consideration in conducting research.

It is the policy and regulations of the University that you observe a deadline of three years from the date of registration to complete your masters' thesis. Do not hesitate to consult this office in case of any problem encountered in the course of your work.

We wish you the best in your research and hope the study will make original contribution to knowledge.

Yours Sincerely,

MASINDE MULIRO UNIVERSITY
OF SCIENCE AND TECHNOLOGY
DIRECTORATE OF POSTGRADUATE STUDIES
P.O. Box 190 – 50100, KAKAMEGA (K)
Date:..... Sign:.....

Prof. Stephen Odebero, PhD, FIEEP
DIRECTOR, DIRECTORATE OF POSTGRADUATE STUDIES

Appendix 10: Experimental Checklist

Phase 1: Preprocessing Setup & Verification

#	Task	Status (✓/N/A)	Notes/Parameters
1.1	Dataset (GTSRB) loaded and integrity verified.		
1.2	Stratified Train (80%) / Test (20%) split performed.		Random Seed
1.3	Interpolation: All images resized to fixed dimensions.		Method: Lanczos4 Target Size: 32x32
1.4	Cropping: Region of Interest (ROI) extraction performed (if applicable).		Method: Aspect Ratio Preserving
1.5	Color Space Conversion: Images converted to target color space.		Method: RGB to Grayscale
1.6	Filtering: Noise reduction filter applied.		Method: Gaussian Blur Kernel Size: (3,3)
1.7	Normalization: Pixel values scaled.		Method: L2 Norm Range: 0-1
1.8	Preprocessing pipeline saved/scripted for consistent application.		

Phase 2: Feature Extraction

#	Task	Status (✓/N/A)	Notes/Parameters
2.1	LD-HOG Descriptor computed on the preprocessed training set.		Cell Size: [8x8] Block Size: [2x2] Bins: [9]
2.2	Gradient & Directional Fusion:		Pooling Type: MAX
2.3	Feature vector length recorded and verified.		Vector Length: [3870]
2.4	Dimensionality Reduction: Technique applied (if applicable).		Method: [LDA] Final Vector Length: [500]
2.5	Feature extraction process repeated identically on the test set.		*(Using parameters fitted on the training set)*

Phase 3: Model Training & Hyperparameter Tuning

#	Task	Status (✓/N/A)	Notes/Parameters
3.1	Classifier Selected: (SVM / Decision Tree / Random Forest)		
3.2	Hyperparameter Tuning: Grid Search / Random Search performed on the training set.		Folds: 5 Scoring Metric: [F1-score, Accuracy and Precision]
3.3	Optimal Hyperparameters recorded.		SVM: Kernel=[RBF] C=[10], Gamma=[scale] RF: n_estimators=[200] max_depth=[50] class_weight=[balanced]

#	Task	Status (✓/N/A)	Notes/Parameters
			DT: criterion=[gini], max_depth=[20]
3.4	Final model trained on the entire training set using optimal hyperparameters.		
3.5	Trained model saved with versioning.		Filename: [svm_ldhog_v1.pkl]

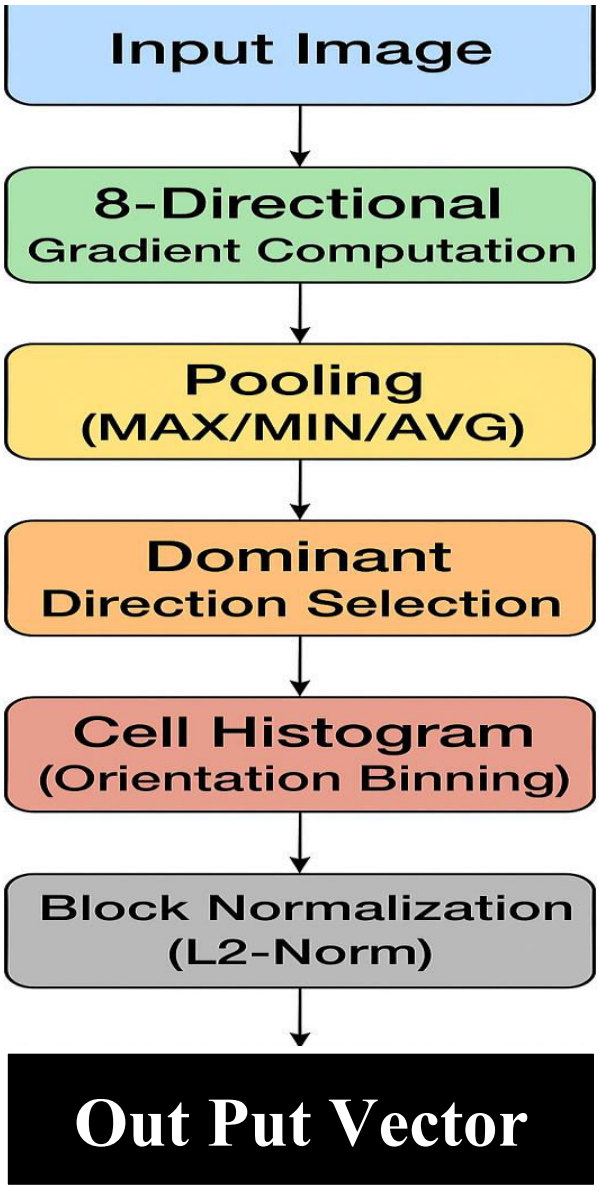
Phase 4: Model Evaluation & Reporting

#	Task	Status (✓/N/A)	Notes/Parameters
4.1	Final model used to predict on the held-out test set.		
4.2	Performance Metrics calculated and recorded.		Accuracy: [Value] Precision (Macro): [Value] Recall (Macro): [Value] F1-Score (Macro): [Value]
4.3	Confusion Matrix generated and saved.		
4.4	Inference Time: Average prediction time per image recorded.		Time: [ms]
4.5	Results compared against baseline models (HOG, LDP).		
4.6	All results, figures, and model files backed up to project directory.		

Appendix 11 Pseudo Code for LD-HOG

1. Algorithm LD-HOG (I, c, b, Nb)
2. Convert I to grayscale and normalize
3. For each pixel (x, y) in I:
4. Compute directional gradients G[0..7] using 8 kernels
5. Apply pooling method to obtain P[0..7]
6. D(x, y) ← argmax(P)
7. Divide I into cells of size c×c
8. For each cell:
9. Compute histogram of orientations (Nb bins)
10. Group cells into blocks of size b×b
11. Normalize block histograms using L2-norm
12. Concatenate all blocks to form F
13. Return F

Appendix 12 LD-HOG Algorithm flow chart



Appendix 13: NACOSTI Research Permit

REPUBLIC OF KENYA

NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Ref No: 715440

Date of Issue: 28/October/2025

RESEARCH LICENSE

This is to Certify that Mr.. Prestone Jeremiah Simiyu of Masinde Muliro University of Science and Technology, has been licensed to conduct research as per the provision of the Science, Technology and Innovation Act, 2013 (Rev.2014) in Bungoma on the topic: Machine Learning for Traffic Sign Recognition for the period ending : 28/October/2026.

License No: NACOSTI/P/25/4181449

715440

Applicant Identification Number

Ag. Director General
NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Verification QR Code

NOTE: This is a computer generated License. To verify the authenticity of this document, Scan the QR Code using QR scanner application.

See overleaf for conditions