NEURAL NETWORKS AND SENSITIVITY ANALYSIS FOR DETECTION AND INTERPRETATION OF STRUCTURAL DAMAGE

A Dissertation
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Dayakar Naik Lavadiya

In Partial Fulfillment of the Requirements
for the Degree of
DOCTOR OF PHILOSOPHY

Major Department
Civil and Environmental Engineering

July 2021

Fargo, North Dakota

North Dakota State University
Graduate School

Title

NEURAL NETWORKS AND SENSITIVITY ANALYSIS FOR DETECTION AND INTERPRETATION OF STRUCTURAL DAMAGE

By

Dayakar Naik Lavadiya

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

DOCTOR OF PHILOSOPHY

SUPERVISORY COMMITTEE:

Dr. Ravi Kiran Yellavajjala
Chair

Dr. Dinesh Katti

Dr. Simone Ludwig

Dr. Kejin Wang

Approved:

11/17/2021
Date

Dr. Xuefeng Chu
Department Chair

ABSTRACT

Computer vision (CV)-based approaches have gained a lot of attention in recent years for objective identification of damages both at structural and material scales. In this dissertation, the metallurgical phases and the two important modes of damage in structural steel, namely fracture and corrosion, are considered. Use of CV techniques for metallurgical phase identification and fracture type identification in steel microstructure is minimal and rely on pixel intensity information. When distinct phases or fracture types possess similar pixel intensities, predictions may be erroneous. In this dissertation, various texture recognition algorithms based on an ensemble of machine learning algorithms are proposed to identify the distinct metallurgical phases and fracture types in structural steels.

The existing CV-based corrosion detection techniques are efficient for the images acquired under natural daylight illumination and ignore the inherent variations in ambient lighting conditions. Further, corrosion-like hues such as bricks, surrounding vegetation, etc., present in the images yields corrosion misclassification. Furthermore, there are currently no techniques available to identify the source of corrosion (HCl, NaCl, and Na_2SO_4). In this dissertation, various color spaces are employed in conjunction with neural networks to identify the corrosion in real-world scenarios such as varying natural daylight illuminations, shadows, water wetting, and oil wetting. For eliminating the visual ambiguity and identifying the source of corrosion, the visible and near-infrared (VNIR) spectra are extracted to train support vector machines.

Deep neural networks (DNN's) popularly used in the field of CV are often regarded as the black box models. Sensitivity analysis (SA) is a model-agnostic explainable artificial intelligence (XAI) approach commonly employed to explain the outcome of a mathematical

model. SA quantifies the variation in the model's output to the change in the input of the model. In this dissertation, a novel sensitivity analysis referred to as Complex-Step Sensitivity Analysis is developed for interpreting the DNN's prediction. Numerical experiments are performed to demonstrate the efficacy of the proposed method in evaluating the derivatives of DNN predictions and identifying the important features. Using this newly developed method, the key wavelengths in the VNIR spectra contributing to the prediction of corrosion source corrosion are identified.

ACKNOWLEDGEMENTS

I would like to thank the Divine Grace for always guiding me and providing for me.

I would like to express my sincere gratitude and deep appreciation to my advisor Dr. Ravi Kiran Yellavajjala for the guidance, and the assistance provided throughout my research. The motivation provided by him has been phenomenal and has constantly inspired me to perform better and to think higher.

I would like to thank the members of my dissertation committee, Dr. Kejin Wang, Dr. Dinesh Katti and Dr. Simone Ludwig for generously offering their precious time, valuable suggestions, and guidance throughout my doctoral tenure here at NDSU. I would also like to thank Dr. Scott Payne, and Jayma Moore of Electron Microscopy Center and, Greg Strommen and Fred Haring from Research Park facility at NDSU for their continued help. I would also like to thank all the members of Damage in Materials and Structures (DAMS) Research Group for the moral support and the helpful discussions.

My tenure at NDSU was in-part funded by grants from National Science Foundation (New Discoveries in the Advanced Interface of Computation, Engineering, and Science), North Dakota Established Program to Stimulate Competitive Research (ND-EPSCoR), Iowa Highway Research Board (IHRB), and the Department of Civil and Environmental Engineering at NDSU. I would also like to acknowledge my gratitude to all the sponsors.

Finally, I would like to acknowledge with gratitude my inspiration, my late father, and the continued support and love of my wife, mother, sister, brother and friends throughout my career, and my doctoral research program in particular.

DEDICATION

To my family and teachers.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AC.....Air Cooled

AISI.....American Iron and Steel Institute

ANN.....Artificial Neural Networks

ASTM .....American Society for Testing Materials

CFDA.....Central Finite Difference Approximation

CIE .....Commission Internationale de l'éclairage

CSDA.....Complex-Step Derivative Approximation

DSLR .....Digital Single-Lens Reflex

DT .....Decision Tree

FDA.....Finite Difference Approximation

FFDNN .....Feed-Forward Deep Neural Networks

FOV.....Field of View

FTIR.....Fourier Transform Infra-Red

GLCM.....Gray-Level Co-occurrence Matrix

GLRLM.....Gray-Level Run Length Matrix

HIS .....Hyperspectral Imaging System

HL .....Hidden Layer

HSV.....Hue Saturation Value

ICDD.....International Center for Diffraction Database

ISO .....International Organization for Standardization

LBP .....Local Binary Pattern

LDA .....Linear Discriminant Analysis

LS.....Least Squares

ML.....Machine Learning

MLP .....Multi-Layer Perceptron

MP.....Mega Pixel

MSE .....Mean-Squared Error

MSI .....Multi-Spectral Image

NB.....Naïve-Bayes

NFRA.....Neuro-Fuzzy Recognition Algorithm

NN.....Neural Networks

PC.....Principal Components

PCA.....Principal Component Analysis

ReLU.....Rectified Linear Unit

SEM .....Scanning Electron Microscopy

SVM.....Support Vector Machine

UAV.....Unmanned Aerial Vehicle

VNIR.....Visible and Near-Infrared

WC .....Water Cooled

XRD .....X-Ray Diffraction

1. INTRODUCTION

1.1. Motivation

The U.S. economy relies on a vast network of infrastructure that includes roads, bridges, dams, rail, ports, energy plants, aviation, etc. [1]. Today, the average age of most of these structures in the US falls in the range of 40 to 50 years (see Table 1.1). Although these structures are still under service, according to the *2021 Report Card for America's Infrastructure* issued by the American Society of Civil Engineers (ASCE), the current US infrastructure system holds a D grade [2]. This implies that most of the structures are in poor to fair condition and below the standards, with many elements approaching the end of their service life. The deterioration and structural deficiency reported for inspected structures also indicate that there may be a high risk of structural failure (see Figure 1.1). However, with the deployment of appropriate resources and necessary maintenance strategies for rehabilitation, repair, or replacement, the life of these structures can be prolonged. The catastrophic failures can be avoided, and the safety, durability, and resilience of a structure can be improved to a greater extent. However, this complete procedure demands a detailed inspection of each component of a structure which is not only time consuming and labor-intensive process but also involves significant challenges for the inspectors at the site [3, 4]. With the integration of state-of-art computer vision (digital images) and artificial intelligence techniques, automatic inspection can be carried out that will aid in deploying appropriate and reasonable timely maintenance measures to prolong the life of the structure.

Table 1.1. Different types of structure in US [2].

Structure	Number	Average age
Bridges	614,387	50
Dams	90580	56
Roads (miles)	4 million	30
Ports	926	-
Energy (miles of transmission lines)	640,000	-

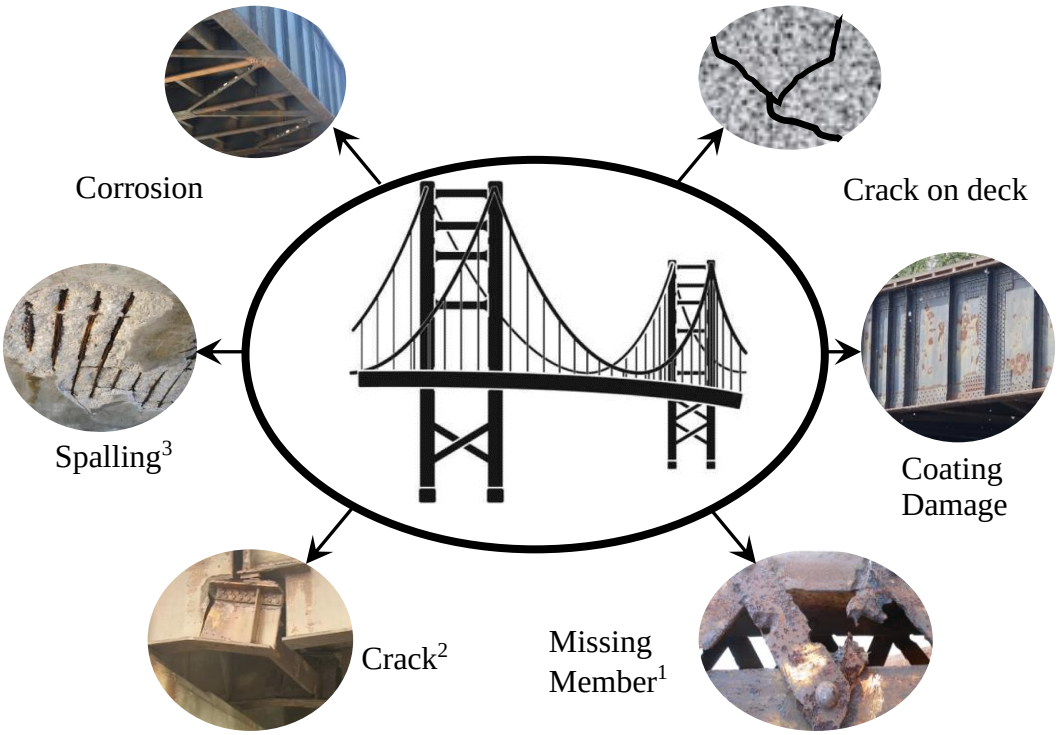


Figure 1.1. Some typical damages observed in civil infrastructure ([1]Anna Frodesiak https://en.wikipedia.org/wiki/Pitting_corrosion; [2]Mary Wisneiwski and Liam Ford, Chicago Tribune;[3]Ripon Chandra Malo, https://www.howtocivil.com/spalling-concrete-causes-prevention-and-repair/;).

In recent years, with the advent of high-resolution imaging cameras and Unmanned Aerial Vehicles (UAS/UAVs) the scope of automated visual inspection has expanded. The condition/deterioration of the surface of the structure could be assessed with the least human intervention. Employing an automated visual inspection system has the following advantages: (1) rapid inspection and assessment, (2) economical, (3) less labor, (4) access to those components

of the structure that are difficult to reach manually, and (5) more detailed data at hand for assessment. An automated visual inspection technique involves four stages (see Figure 1.2): (1) navigation of UAVs/drones over the various locations of the structure to acquire the images for training of artificial intelligence-based algorithms, (2) processing the image and extraction of descriptive features of the damage, (3) building and training an artificial intelligence-based algorithm that maps the descriptive features to the corresponding damage and (4) deploying the trained algorithm to identify and assess the damage of other structures in the field. Note that tasks 2 to 4 mainly constitute the integration of two fields, namely, computer vision and Artificial Intelligence (AI)/Machine learning (ML). Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos. In computer vision, the descriptive features of damage play a vital role in determining the accuracy of the prediction. The more distinct and relevant the features are, the better is the accuracy. Examples of image features include shape, color, the texture of the object, etc. Note that there are various algorithms available in the literature for extracting the features from the image [5].

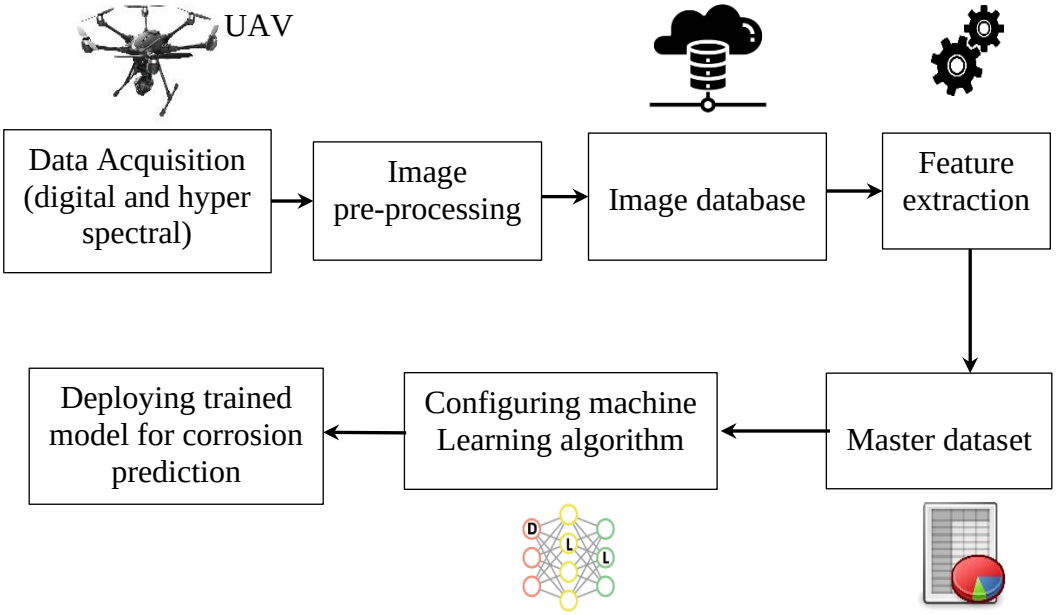


Figure 1.2. Stages involved in automated visual inspection.

Artificial Intelligence is a field that enables computers to mimic various aspects of human intelligence, including – pattern recognition, data-driven learning, knowledge-based reasoning, natural language understanding, and planning and control. While artificial intelligence is a vast field, generally, its subdomains, machine learning, and deep learning are employed in practice with computer vision tasks [6]. On the one hand, Machine learning (ML) includes the algorithms/statistical models that computer systems use in order to perform a specific task without using explicit instructions effectively, and on the other hand, Deep Learning (DL) (e.g., deep neural networks, convolutional neural networks, etc.) includes computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. ML is further classified into three categories, namely supervised (e.g., Naïve Bayes, K-Nearest Neighbor (k-NN), Decision Tree, Support Vector Machines (SVM), Neural Networks, etc.), unsupervised (e.g., autoencoders, clustering, association rules, etc.) and reinforcement learning (Q-learning, State-Action-Reward-State-Action (SARSA), etc.).

Supervised learning involves a labeled dataset wherein the mapping between the descriptive features and the associated target variable is learned (see Chapter 2), and unsupervised learning involves an unlabeled dataset wherein the relationship between the instances within the dataset is determined. However, in reinforcement learning, the goal is to find a suitable action model that would maximize the total cumulative reward of the agent.

In this dissertation, the research is carried out in two phases, namely Research phase I and Research phase II. While Research phase I deals with the research gaps identified in computer vision tasks applied to specific structural problems (see Section 1.2), Research phase II deals with the interpretations of the deep neural networks (see Section 1.3). In what follows, a brief background on these two research phases is provided.

1.2. Computer Vision for Structural Material Characterization and Damage Detection in Infrastructure

Computer vision and AI/ML algorithms have gained a lot of attention in the recent decade for identifying the damages and characterizing the materials in infrastructure. While some of the damages typically observed in structures are evident at the macroscopic scale (e.g., cracking, concrete spalling, corrosion, etc.), the other damages are evident in the form of microstructural changes at microscopic scale (e.g., change of metallurgical phase, brittle/ductile fracture etc.) (see Figure 1.3). In this dissertation, only damages associated with steel structures are considered. Specifically, metallurgical phase and fracture type identification at the microscopic scale and corrosion damage at the macroscopic scale. Metallurgical phase identification is important in the structures that are prone to fire accidents since the rise in temperature influences the composition and change of metallurgical phases (see Chapter 2). Similarly, fracture identification will aid in deploying appropriate damage models for

simulations (see Chapter 3). Corrosion damage is found to play a vital role in the overall maintenance cost of the steel structures (see Figure 1.4) [7-9]. In the United States, the average annual cost of corrosion damage for steel bridges is estimated to be ~$10.15 billion [9]. Detection of corrosion in its early stages not only results in the reduction of maintenance costs but also increases the life of the structures [10]. In this dissertation, the focus is limited to the identification of metallurgical phases and fracture type and corrosion detection (see Figure 1.3). A brief overview of each of the above topics is provided, and the research gaps are identified.

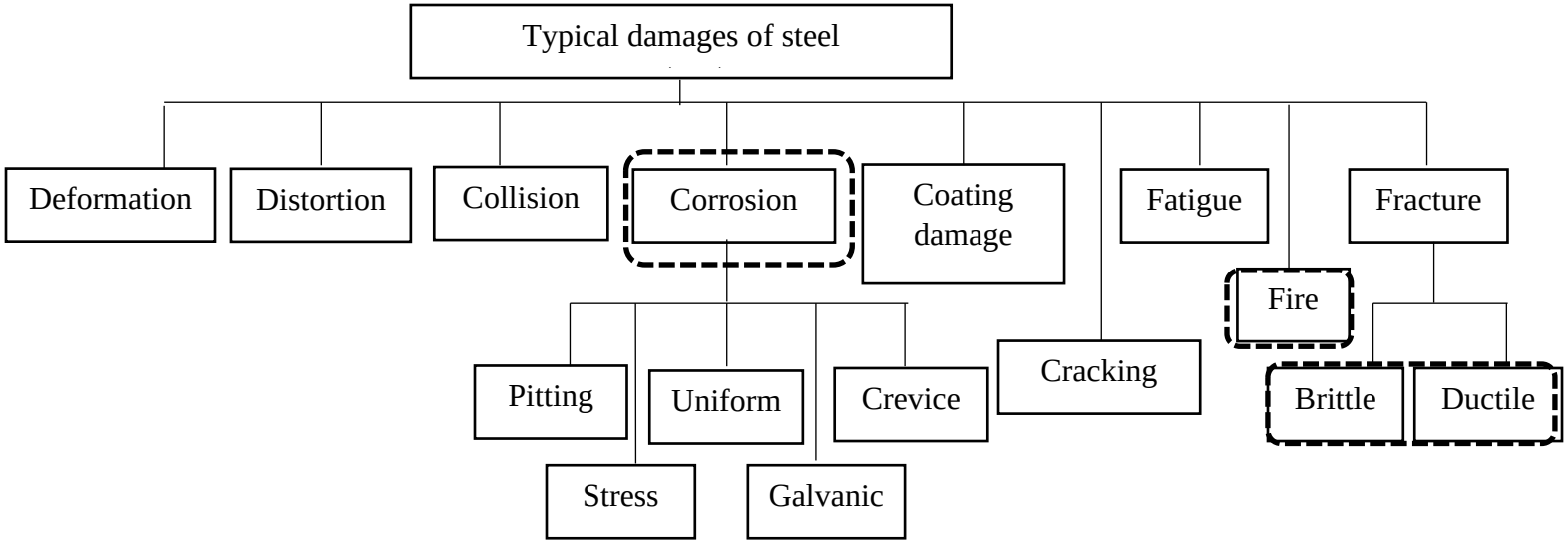


Figure 1.3. Types of damages in steel structures.

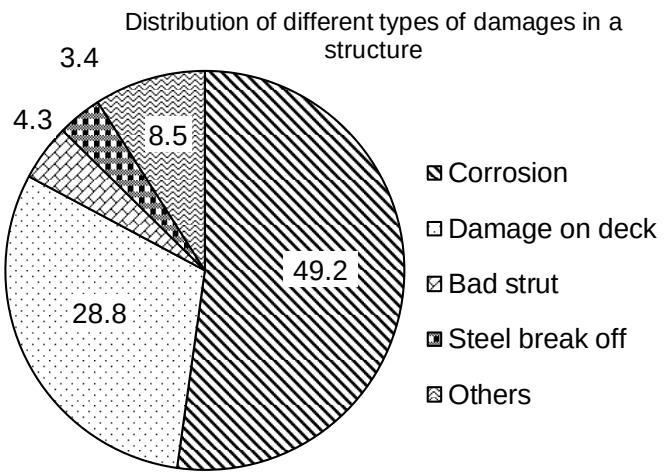


Figure 1.4. Fraction of damages observed in steel structures [11].

1.2.1. Identification of Metallurgical Phases

Engineering metallic alloys like dual-phase steels, α-β brass, α-β titanium, etc., possess multi-phase polycrystalline microstructures [12] that are characterized by the grain sizes, distinct phases, and their volume and morphology [13], also referred to as microstructural features. Microstructural information aids in determining the bulk material properties that will guide engineers design components for specialized applications (ex: Hall-Petch relation [14, 15]). Often light optical microscopy is employed in conjunction with digital image processing techniques for performing tasks such as metallurgical phase identification and evaluation of grain sizes. Among various image processing techniques, histogram-based thresholding or Otsu's method [16] is extensively used for image analysis or segmentation of microstructure [17]. A detailed review of other methods available in the literature is provided in Chapter 2. In Otsu's technique, a threshold-based criterion is established from the multimodal histogram of pixel intensities which is used for the segmentation of distinct phases. Implementation of such a technique results in an accurate segmentation of metallurgical phases whose pixel intensity distribution are distinct (multimodal) and do not overlap significantly (see Figure 1.5(a)). However, when there are multiple metallurgical phases whose pixel intensity distribution

overlaps with each other (see Figure 1.5(b)), employing histogram-based thresholding may lead to misclassification of phases.

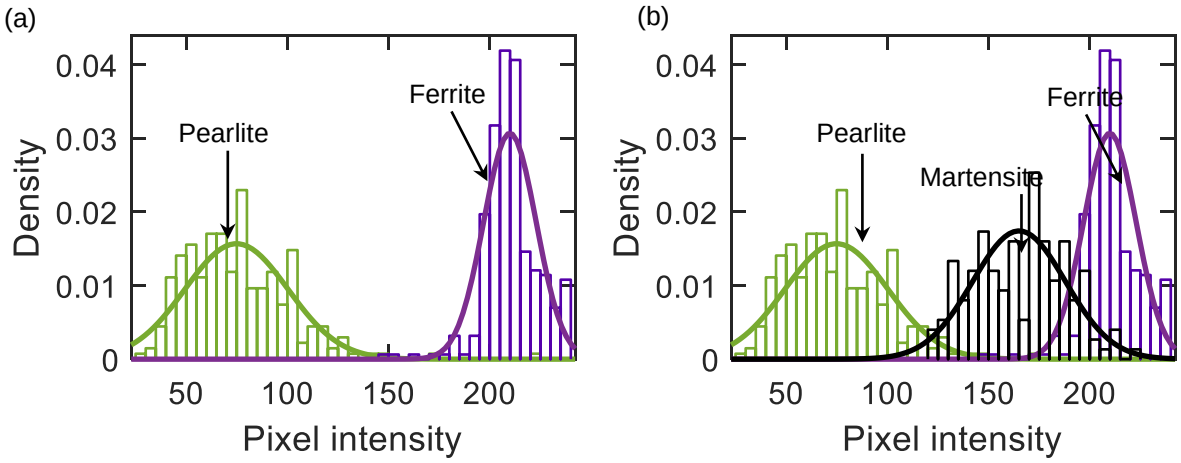


Figure 1.5. Probability density function of pixel intensities of metallurgical phases (a) well-discriminated ferrite and pearlite phase and (b) overlapped ferrite, pearlite, and martensite phases.

1.2.2. Identification of Fracture Type

Fracture in metals has led to catastrophic failures in steel buildings [18] and bridges [19], oil and gas pipelines [20], automobiles [21], and aerospace structures [22]. Commonly identified types of fracture in metals under monotonic loading conditions are ductile fracture, cleavage/ transgranular fracture, and intergranular fracture [23]. A decision about the choice of a suitable damage model to simulate the failure depends on the type of fracture as the basic microscopic damage mechanisms leading to the fracture vary from one fracture type to another. Often visual inspection of fractographic images is conducted in practice to identify the fracture type of metal. However, the visual inspection is not only slow and prone to confirmation bias but also cannot be used for quantitative analysis of fracture surfaces. Currently, there are only a few techniques available in the literature to perform the automatic segmentation of fractographic images in steel [24]. These techniques employ pixel intensities and textural features for fracture type identification. An object is considered to possess a texture if its appearance is composed of

repetitive visual patterns defined by variations in brightness and/ or color [25]. Examples of texture possessing objects include wood, grass, soil, concrete, etc. Since the brittle and ductile fracture exhibit distinct textural characteristics, the textural features are considered in the literature. However, the textural feature extraction methods employed in literature are illumination/ grayscale and rotation variant, i.e., the extracted textural features may change with the change in the illumination and rotation of an image [26]. In reality, the fracture surfaces are quite uneven at microscales. Hence, the fractographic images obtained from scanning electron microscopy (SEM) consists of varying illumination levels across the image. Therefore, an illumination/ grayscale and rotation invariant method is required to extract textural features which may result in more accurate prediction when compared to other textural feature extraction methods.

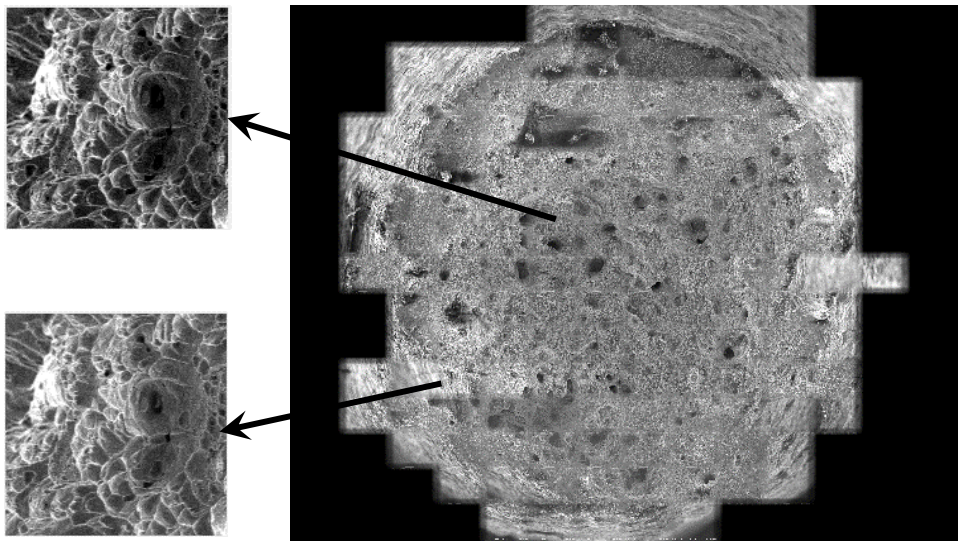


Figure 1.6. Fracture surface of 316L stainless steel illustrating varying illuminations.

1.2.3. Detection of Corrosion

Currently, either human inspection or non-destructive techniques like eddy current technique [27], ultrasonic inspection [28, 29], acoustic emission technique [30, 31], vibration

analysis [32], radiography [33], thermography [34], optical inspection [35], etc. are employed to monitor and identify the corrosion damage in the steel structures. Although each of the above-mentioned techniques has its advantages, the optical inspection technique is most commonly preferred owing to its simplicity and ease of interpretation. In recent times, various approaches have been proposed by researchers to detect corrosion in steel structures using digital images [36]. Most of these approaches included either acquisition of grayscale images or a color image of the corroded steel structure under uniform illumination conditions (i.e., same time of the day, without shadows). Color is defined as a small portion of the electromagnetic spectrum that is visible to the human eye and covers wavelength in the range of 380nm to 740nm [37]. When compared to grayscale images, color images have more information i.e. chromaticity and luminosity [38]. Chromaticity refers to the combination of the dominant wavelength of the visible light (called hue) reflected from the material surface and the purity (saturation) associated with it, and luminosity refers to the intensity of light per unit area of the light source. From a practical perspective, there is a need to develop a more robust technique that can be used to identify the corrosion in steel structures using images taken under varying illuminations, dark shadows, water, and oil wetting.

Furthermore, it is also important to note that the employment of color features alone may have some limitations. One case where this technique is not applicable is when the objects in the acquired images possess hue values similar to that of a corroded surface. For instance, coatings, dirt, or some vegetation in the background may be misclassified as corrosion. This optical confusion is regarded as visual ambiguity (see Figure 1.7). The other case where color features may have limitations is the identification of corrosion source, i.e., the corrosive media (see Figure 1.8). Different corrosive media yields different corroded surfaces that are chemically

distinct. The chemical distinctiveness is generally characterized in terms of corrosion products which governs the acceleration or deceleration of the corrosion. Identifying the source of corrosion will aid in deploying an appropriate corrosion mitigation strategy [19–21].

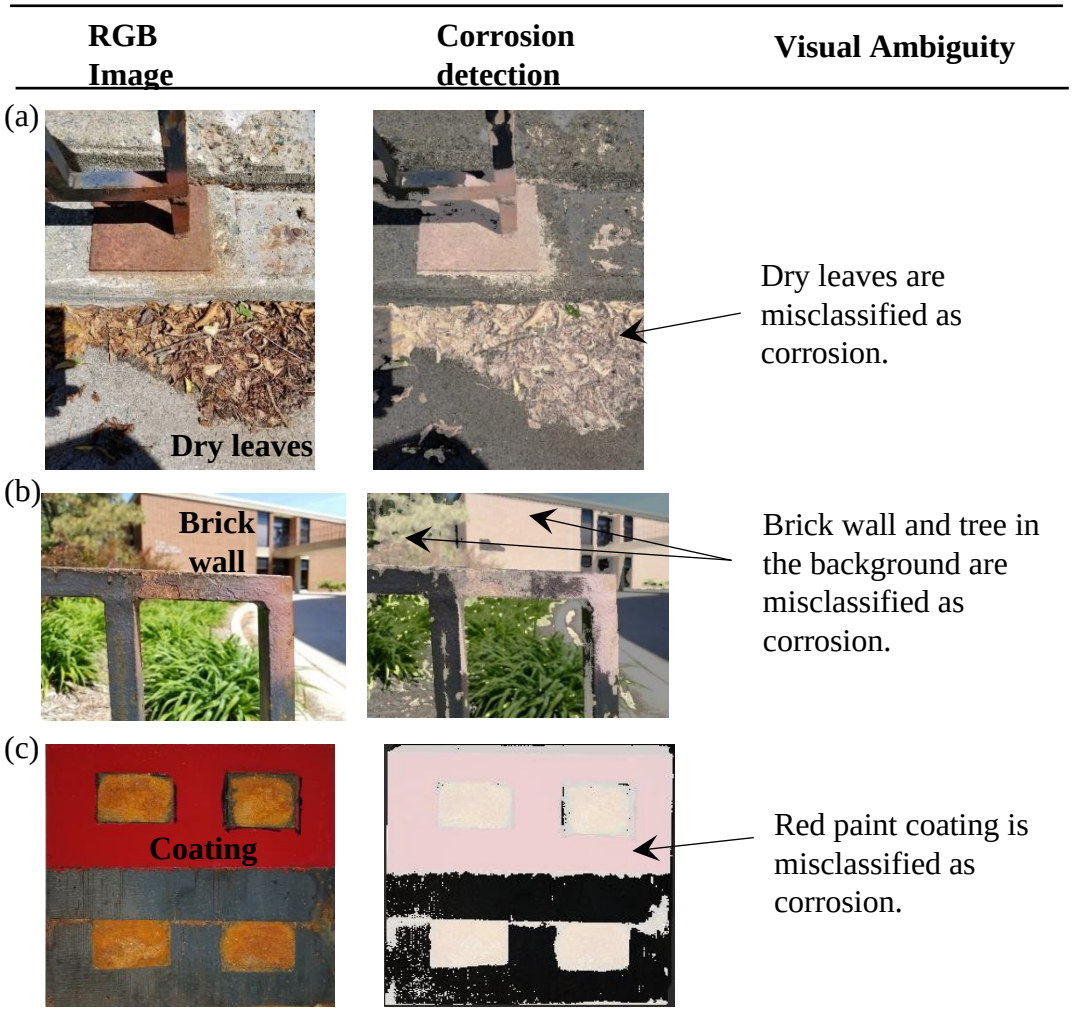


Figure 1.7. Corrosion misclassification due to visual ambiguity: (a) dry leaves misclassified as corrosion surface, (b) brick wall misclassified as corroded surface, and c) coating misclassified as corrosion surface. Note that the corrosion predictions provided in the above images are obtained using the trained multi-layer perceptron configuration described by the authors in [18].

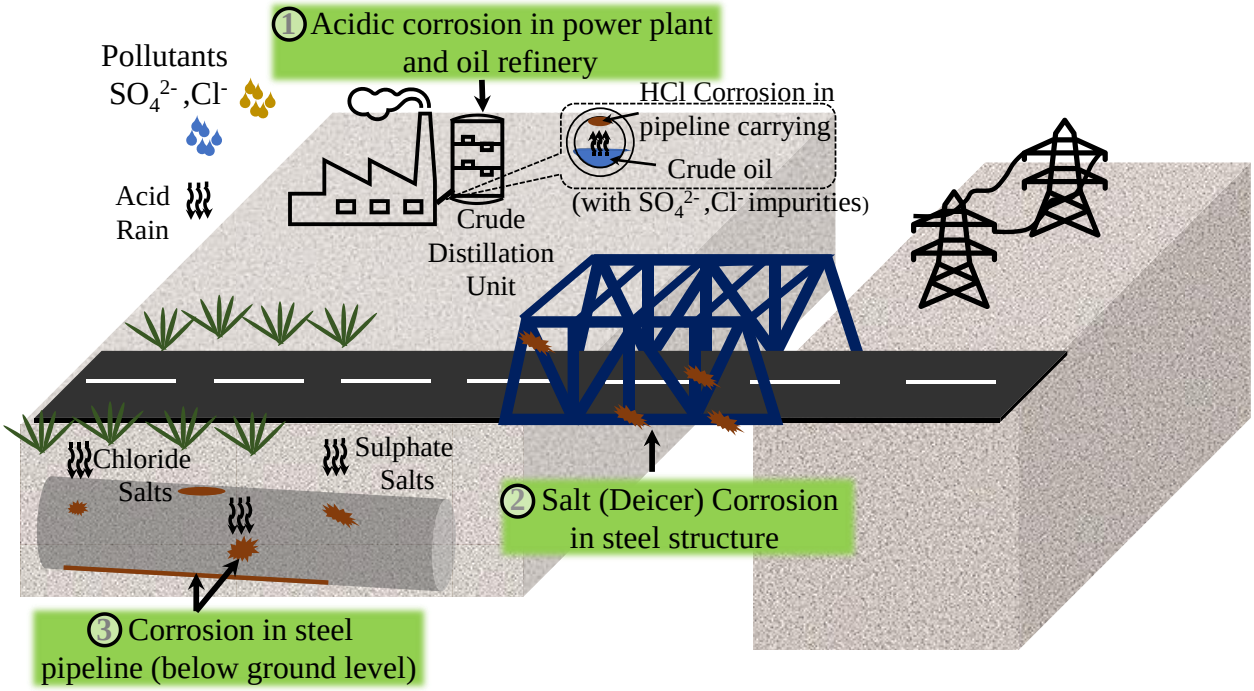


Figure 1.8. Graphical illustration of the corrosion commonly observed in civil infrastructure: 1 – power plants and oil refineries where acidic corrosion is initiated by the impurities transported by crude oil, 2 – bridges and other steel structures exposed to deicing salts and acidic rains, and 3 – underground pipelines exposed to various salts leaching through the soil.

1.3. Explainable Artificial Intelligence (XAI)

The emergence of sophisticated machine learning (ML) models and their proven ability to make accurate predictions by learning complex patterns from the data has gained a lot of attention in various domains in the past decade [39]. For example, tasks such as image recognition, biometrics, text mining, business analytics, etc. [39, 40]. However, such ML models are complicated in their structure and are difficult to comprehend, i.e., they act as a black box (BB), and their functioning is not easily understood. As shown in Figure 1.9, the transparency (ability to interpret) of the ML model is compromised as the model accuracy increases [41]. Trusting the decisions or predictions of the ML models may not be justifiable in the fields such as medicine, law, defense, etc., if the complete explanations are not provided [41]. To

circumvent the opaqueness of the ML models and obtain the information accountable for the model outcome, human-understandable knowledge extracting algorithms are necessary. Such algorithms or models are referred to as interpretable or explainable machine learning models (XAI) [41-43]. The aim of the XAI models is to deliver the rules or the output symbols along with the predictions such that the rationale behind the mappings of the BB model is revealed [41, 44]. More precisely, the representations of interpretations can be delivered in the form of feature summary, model parameters such as weights or coefficients, counterfactual explanations, etc. Note that the goal of XAI is only to provide an answer for the question "why a certain prediction is made?" and not verifying the correctness of the prediction. According to [45], the interpretable or XAI models are the one that enables human users to understand, trust and effectively manage the emerging generation of artificially intelligent partners. Another definition for XAI is provided by Arrieta et al. [41], which takes the expertise of an audience into consideration and is quoted as

"Given an audience, explainable AI is one that produces details or reasons to make its functioning clear or easy to understand."

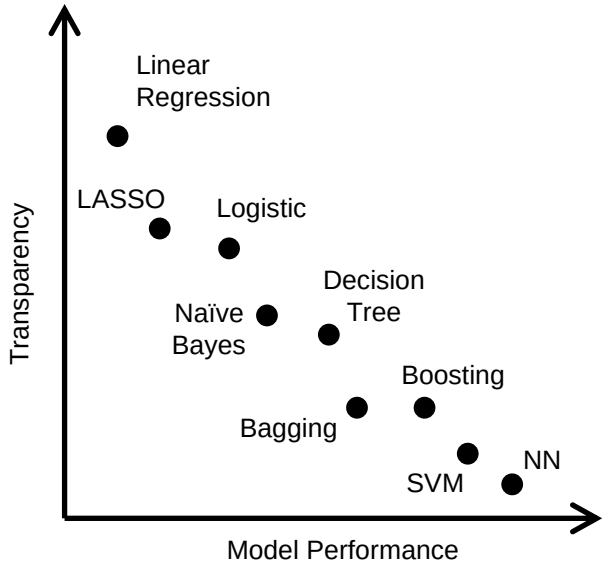


Figure 1.9. The trade-off between model transparency and model performance [46].

Interpretability of predictions from trained BB models can be achieved by either employing a set of algorithms that are intrinsically interpretable (e.g., linear regression, decision trees, rule fit, etc.) or using techniques such as visualization, local explanations, explanation by example, feature relevance, etc. (see Figure 1.10) [41, 47]. For example, the weights in the linear regression model and the split of the features in the decision tree model aids in inferring how the features are associated with the decision-making process. When the input features and their respective predictions obtained from the trained BB model are mapped to the whole training dataset by employing an intrinsically interpretable algorithm, then such models are referred to as global or surrogate models [48]. On the contrary, when the input features and their respective predictions obtained from the trained BB model are mapped to a single instance in the small region of interest (local neighborhood), then such models are referred to as local models [48]. For locally interpretability models, it is assumed that the predictions are linearly dependent on a subset of features in the local neighborhood rather than having complex dependencies among all the features and serves as a good approximation. According to Robnik-Sikonja et al. [49] a good

interpretable model should possess the following properties, namely, expressive power, translucency, portability, and algorithmic complexity; and an explanation by the model should possess properties such as accuracy, fidelity, consistency, stability, comprehensibility, certainty, importance, novelty, representativeness [48].

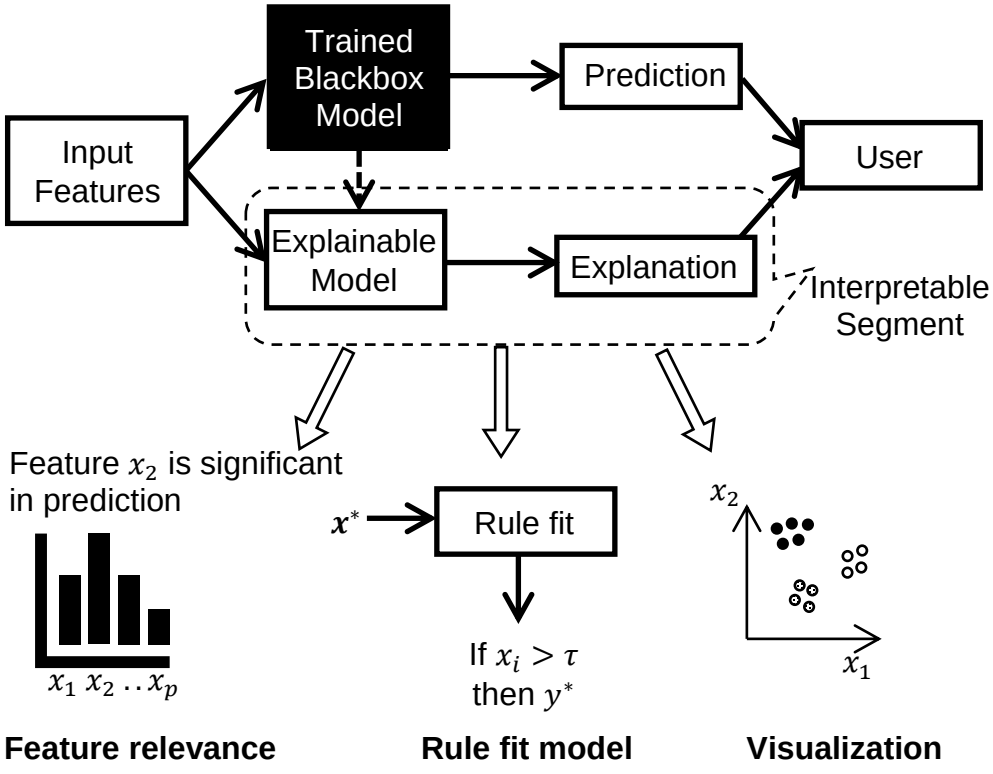


Figure 1.10. Interpretability pipeline for black-box models [41].

1.3.1. Significance of Interpretable Machine Learning

Integration of interpretable or explainable models with black-box models has the following significance [41, 42, 47]

1. Incorporating interpretability with BB models will aid in degubbaility and rectification of the deficiencies in the model i.e., it will allow finding a direction to fix the model such that misleading predictions can be avoided.

2. Interpretability will improve the robustness of the model by highlighting the adversarial perturbations that could impact the predictions.
3. While the current development of ML models focuses on performance, the search for understanding or acquiring new knowledge in the field of science through interpretability will be of practical importance i.e., they will act as a pre-requisite for new insights.
4. Providing explanation through interpretability models will result in gaining human trust, which will subsequently encourage the social acceptance and integration of the AI systems into daily lives.
5. Interpretability provides the means to verify the fairness of the model. In other words, bias in the model due to wrong parameterization or incompleteness of the problem can be detected. An explanation is complete when it allows the behavior of the system to be anticipated in diverse situations.

1.3.2. Sensitivity Analysis of Deep Neural Networks

Multilayer feedforward deep neural networks (FFDNN) are parameterized nonlinear models that approximate a mathematical mapping between the input features and the output target variables [50]. FFDNN are often treated as a black-box model due to their complex nature. In other words, it is difficult to interpret the predictions of FFDNN since the closed-form function is unknown. Nevertheless, the sensitivity analysis could be performed to understand the influence of each input feature on the target output [51]. Sensitivity analysis involves the examination of the change in the target output of the model when the input features are varied. According to Saltelli [52] the sensitivity analysis can either be local or global. While local sensitivity analysis assesses a particular point in input feature space, varying features one at a time to obtain a local response of the model to each feature, the global sensitivity analysis tries to

capture the entire feature space all at once, allowing multiple feature values to be explored simultaneously [53]. In general, local sensitivity analysis is preferred in practice since they provide elementary effects of input features on the model output. However, global sensitivity analysis may be required if the influence of feature interaction on the model output has to be determined [54]. Local sensitivity analysis involves the computation of the partial derivative of the target output with respect to the perturbed input feature [55]. Nevertheless, other measures such as range, variance, and average absolute deviation may also be employed [56]. Keeping in view that partial derivatives are the most commonly employed measure for performing sensitivity analysis in practice, this dissertation focuses on the pitfalls of the existing techniques, and a novel approach is developed.

1.4. Research Gaps

The research gaps identified for Research Phase I and Research Phase II are as follows.

Research Phase I

1. Pixel intensity alone may be insufficient for phase identification when there are more than two metallurgical phases in the microstructure.
2. Currently, there are no automatic methods that detect and quantify fracture types in structural steels.
3. Corrosion detection under ambient lighting conditions such as shadows, water wetting, and oil wetting have not been addressed yet.
4. The current corrosion detection techniques cannot eliminate the visual ambiguity and identify the corrosion source.

Research Phase II

5. Novel sensitivity methods are required to explain the predictions of deep neural networks.

1.5. Research Objectives

The objectives of Research Phase I and Research Phase II are as follows.

Research phase I

1. To identify the distinct metallurgical phases and evaluate their volume fractions in structural steels.
2. To identify the brittle and ductile fracture in structural steels and evaluate their area fractions.
3. To detect the corrosion in structural steels under varying illuminations, cast shadows, water wetting, and oil wetting conditions.
4. To identify the source of corrosion and eliminate the visual ambiguity during corrosion detection.

Research phase II

5. To evaluate numerical derivatives of the output in deep neural networks with respect to input features.
6. To formulate a feature attribution algorithm for identifying the features that are important in determining the output of the deep neural network.

1.6. Dissertation Organization

This dissertation is organized into nine chapters. Chapter 1 provides a brief overview of the background required for the current research and lists the specific research objectives. Chapter 2 describes the concept of image texture and demonstrates its efficacy in the

identification of the metallurgical phases in heat-treated ASTM A36 steel using ML algorithms. Chapter 3 highlights the need for rotational and gray-scale invariant textural features and extends the concept to the fractographic images wherein brittle and ductile type fractures are identified. Chapter 4 explores the various combinations of color spaces and configurations of multi-layer perceptron for corrosion detection in ambient lighting conditions. Chapter 5 describes the principle of the hyperspectral imaging system and reveals the efficacy of spectral features in the elimination of visual ambiguity and detection of various corrosion sources. Chapter 6 provides the review of explainable AI algorithms available in XAI literature. Chapter 7 proposes novel algorithms referred to as Complex-step derivative approximation (CSDA) and Complex-step feature attribution (CS-FA) for evaluating the first-order derivatives of the output of the neural network and performing sensitivity analysis, respectively. Chapter 8 demonstrates the efficacy of the proposed algorithms by employing real-world datasets and the hyperspectral imaging dataset. Note that while Chapter 2 to Chapter 5 are associated with the objectives of Research Phase I, Chapter 6 to Chapter 8 are associated with Research Phase II. Chapter 9 provides the conclusions of the current research and the possible future research directions.

1.7. List of Publications from Thesis

The following journal articles are produced during the dissertation tenure at NDSU.

1. Ravi Kiran, and **Naik, Dayakar L,** "Novel sensitivity method for evaluating the first derivative of the feed-forward neural network outputs." *Journal of Big Data* 8.1 (2021): 1-13
2. **Naik, Dayakar L.,** et al. "Detection of Corrosion-Indicating Oxidation Product Colors in Steel Bridges under Varying Illuminations, Shadows, and Wetting Conditions." Metals 10.11 (2020): 1439.

3. **Naik, Dayakar L.,** and Ravi Kiran. "Identification and characterization of fracture in metals using machine learning based texture recognition algorithms." Engineering Fracture Mechanics 219 (2019): 106618.
4. **Naik, Dayakar L,** Hizb Ullah Sajid, and Ravi Kiran, "Texture-Based Metallurgical Phase Identification in Structural Steels: A Supervised Machine Learning Approach". Metals (2019).
5. **Naik, Dayakar L**, and Ravi Kiran, "Hyperspectral Imaging for the Elimination of Visual Ambiguity in Corrosion Detection and Identification of Corrosion Sources". Structural Health Monitoring (Under Review).
6. **Naik, Dayakar L**, and Ravi Kiran, "A Novel Sensitivity-based Method for Feature Selection". Journal of Big Data (Under Review)

   Other A.I related Publications
7. Flores, P., Zhang, Z., Igathinathane, C., Jithin, M., **Naik, Dayakar L**., Stenger, J., ... & Kiran, R. Distinguishing seedling volunteer corn from soybean through greenhouse color, color-infrared, and fused images using machine and deep learning. Industrial Crops and Products, 161, 113223.
8. Zhang, Z., Flores, P., Igathinathane, C., **Naik, Dayakar L**., Kiran, R., & Ransom, J. K. "Wheat Lodging Detection from UAS Imagery Using Machine Learning Algorithms." Remote Sensing 12.11 (2020): 1838.
9. **Naik, Dayakar L**., and Ravi Kiran. "Data Mining and Equi-accident zones for US Pipeline Accidents." ASCE Journal of Pipeline Systems Engineering and Practice, 9(4) (2018): 1-28.

10. **Naik, Dayakar L.**, and Ravi Kiran. "Naïve Bayes classifier, multivariate linear regression and experimental testing for classification and characterization of wheat straw based on mechanical properties." Industrial Crops and Products 112 (2018): 434-448.

1.8. References

1. Khan A, Becker K. US Infrastructure: Challenges and Directions for the 21st Century: Routledge; 2019.
2. Engineers ASoC. 2017 infrastructure report card. 2017.
3. Wu H-C, Eamon CD. Strengthening of Concrete Structures Using Fiber Reinforced Polymers (FRP): Design, Construction and Practical Applications: Woodhead Publishing; 2017.
4. Chang PC, Flatau A, Liu S. Health monitoring of civil infrastructure. Structural health monitoring. 2003;2:257-67.
5. Nixon MS, Aguado AS. Feature extraction & image processing for computer vision: Academic Press; 2012.
6. Rich E, Knight K. Artificial intelligence. McGraw-Hill New York; 1991.
7. Troitsky MS. Planning and design of bridges: John Wiley & Sons; 1994.
8. Haas T. Are Reinforced Concrete Girder Bridges More Economical Than Structural Steel Girder Bridges?: A South African Perspective. Jordan Journal of Civil Engineering. 2014;159:1-15.
9. Sastri VS. Challenges in corrosion: costs, causes, consequences, and control: John Wiley & Sons; 2015.
10. Chen W-F, Duan L. Bridge engineering handbook: construction and maintenance: CRC press; 2014.

11. Wang R, Kawamura Y. An automated sensing system for steel bridge inspection using GMR sensor array and magnetic wheels of climbing robot. Journal of Sensors. 2016;2016.

12. Fan Z. Microstructure and mechanical properties of multiphase materials: University of Surrey; 1993.

13. Bales B, Pollock T, Petzold L. Segmentation-free image processing and analysis of precipitate shapes in 2D and 3D. Modelling and Simulation in Materials Science and Engineering. 2017;25:045009.

14. Hall E. The deformation and ageing of mild steel: III discussion of results. Proceedings of the Physical Society Section B. 1951;64:747.

15. Petch N. The influence of grain boundary carbide and grain size on the cleavage strength and impact transition temperature of steel. Acta metallurgica. 1986;34:1387-93.

16. Otsu N. A threshold selection method from gray-level histograms. IEEE transactions on systems, man, and cybernetics. 1979;9:62-6.

17. Sosa JM, Huber DE, Welk B, Fraser HL. Development and application of MIPAR™: a novel software package for two-and three-dimensional microstructural characterization. Integrating Materials and Manufacturing Innovation. 2014;3:10.

18. Clifton C, Bruneau M, MacRae G, Leon R, Fussell A. Steel structures damage from the Christchurch earthquake series of 2010 and 2011. Bulletin of the New Zealand Society for Earthquake Engineering. 2011;44:297-318.

19. Russo FM, Mertz DR, Frank KH, Wilson KE. Design and Evaluation of Steel Bridges for Fatigue and Fracture–Reference Manual. 2016.

20. Naik Dayakar L, Kiran R. Data Mining and Equi-Accident Zones for US Pipeline Accidents. Journal of Pipeline Systems Engineering and Practice. 2018;9:04018019.

21. El-Magd E, Gese H, Tham R, Hooputra H, Werner H. Fracture criteria for automobile crashworthiness simulation of wrought aluminium alloy components. Materialwissenschaft und Werkstofftechnik: Materials Science and Engineering Technology. 2001;32:712-24.

22. Adib A, Baptista C, Barboza M, Haga C, Marques C. Aircraft engine bleed system tubes: Material and failure mode analysis. Engineering Failure Analysis. 2007;14:1605-17.

23. Anderson TL. Fracture Mechanics: fundamentals and applications. 3rd ed. Boca Raton, FL: CRC Press; 2004.

24. Kosarevych RY, Student O, Svirs'ka L, Rusyn B, Nykyforchyn H. Computer analysis of characteristic elements of fractographic images. Materials Science. 2013;48:474-81.

25. Naik DL, Sajid HU, Kiran R. Texture-Based Metallurgical Phase Identification in Structural Steels: A Supervised Machine Learning Approach. Metals. 2019;9:546.

26. Gad AF. Practical Computer Vision Applications Using Deep Learning with CNNs.

27. García-Martín J, Gómez-Gil J, Vázquez-Sánchez E. Non-destructive techniques based on eddy current testing. Sensors. 2011;11:2525-65.

28. Pavlopoulou S, Staszewski W, Soutis C. Evaluation of instantaneous characteristics of guided ultrasonic waves for structural quality and health monitoring. Structural Control and Health Monitoring. 2013;20:937-55.

29. Sharma S, Mukherjee A. Ultrasonic guided waves for monitoring corrosion in submerged plates. Structural Control and Health Monitoring. 2015;22:19-35.

30. Nowak M, Lyasota I, Baran I. The test of railway steel bridge with defects using acoustic emission method. Journal of Acoustic Emission. 2016;33.

31. Cole P, Watson J. Acoustic emission for corrosion detection. Advanced Materials Research: Trans Tech Publ; 2006. p. 231-6.

32. Deraemaeker A, Reynders E, De Roeck G, Kullaa J. Vibration-based structural health monitoring using output-only measurements under changing environment. Mechanical systems and signal processing. 2008;22:34-56.

33. McCrea A, Chamberlain D, Navon R. Automated inspection and restoration of steel bridges—a critical review of methods and enabling technologies. Automation in Construction. 2002;11:351-73.

34. Doshvarpassand S, Wu C, Wang X. An overview of corrosion defect characterization using active infrared thermography. Infrared Physics & Technology. 2018.

35. Jahanshahi MR, Kelly JS, Masri SF, Sukhatme GS. A survey and evaluation of promising approaches for automatic image-based defect detection of bridge structures. Structure and Infrastructure Engineering. 2009;5:455-86.

36. Chen P-H, Chang L-M. Artificial intelligence application to bridge painting assessment. Automation in construction. 2003;12:431-45.

37. Chen P-H, Yang Y-C, Chang L-M. Automated bridge coating defect recognition using adaptive ellipse approach. Automation in Construction. 2009;18:632-43.

38. Shen H-K, Chen P-H, Chang L-M. Automated steel bridge coating rust defect recognition method based on color and texture feature. Automation in Construction. 2013;31:338-56.

39. Gevers T, Gijsenij A, Van de Weijer J, Geusebroek J-M. Color in computer vision: fundamentals and applications: John Wiley & Sons; 2012.

40. Koschan A, Abidi M. Digital color image processing: John Wiley & Sons; 2008.

41. Došilović FK, Brčić M, Hlupić N. Explainable artificial intelligence: A survey. 2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO): IEEE; 2018. p. 0210-5.

42. Samek W, Wiegand T, Müller K-R. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. arXiv preprint arXiv:170808296. 2017.

43. Arrieta AB, Díaz-Rodríguez N, Del Ser J, Bennetot A, Tabik S, Barbado A, et al. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion. 2020;58:82-115.

44. Gilpin LH, Bau D, Yuan BZ, Bajwa A, Specter M, Kagal L. Explaining explanations: An overview of interpretability of machine learning. 2018 IEEE 5th International Conference on data science and advanced analytics (DSAA): IEEE; 2018. p. 80-9.

45. Gunning D, Stefik M, Choi J, Miller T, Stumpf S, Yang G-Z. XAI—Explainable artificial intelligence. Science Robotics. 2019;4.

46. Murdoch WJ, Singh C, Kumbier K, Abbasi-Asl R, Yu B. Definitions, methods, and applications in interpretable machine learning. Proceedings of the National Academy of Sciences. 2019;116:22071-80.

47. Gladkovskaya O, Greaney P, Gun'ko YK, O'Connor GM, Meere M, Rochev Y. An experimental and theoretical assessment of quantum dot cytotoxicity. Toxicology Research. 2015;4:1409-15.

48. Morocho-Cayamcela ME, Lee H, Lim W. Machine learning for 5G/B5G mobile and wireless communications: Potential, limitations, and future directions. IEEE Access. 2019;7:137184-206.

49. Molnar C. Interpretable Machine Learning; 2020.

50. Carvalho DV, Pereira EM, Cardoso JS. Machine learning interpretability: A survey on methods and metrics. Electronics. 2019;8:832.

51. Robnik-Šikonja M, Kononenko I. Explaining classifications for individual instances. IEEE Transactions on Knowledge and Data Engineering. 2008;20:589-600.

52. Zhang Z, Beck MW, Winkler DA, Huang B, Sibanda W, Goyal H. Opening the black box of neural networks: methods for interpreting neural network models in clinical applications. Annals of translational medicine. 2018;6.

53. Nourani V, Fard MS. Sensitivity analysis of the artificial neural network outputs in simulation of the evaporation process at different climatologic regimes. Advances in Engineering Software. 2012;47:127-46.

54. Campolongo F, Saltelli A, Cariboni J. From screening to quantitative sensitivity analysis. A unified approach. Computer Physics Communications. 2011;182:978-88.

55. Dresch JM, Liu X, Arnosti DN, Ay A. Thermodynamic modeling of transcription: sensitivity analysis differentiates biological mechanism from mathematical model-induced effects. BMC systems biology. 2010;4:1-11.

56. Hsieh N-H, Reisfeld B, Bois FY, Chiu WA. Applying a global sensitivity analysis workflow to improve the computational efficiencies in physiologically-based pharmacokinetic modeling. Frontiers in pharmacology. 2018;9:588.

57. Morio J. Global and local sensitivity analysis methods for a physical system. European journal of physics. 2011;32:1577.

58. Cortez P, Embrechts MJ. Using sensitivity analysis and visualization techniques to open black box data mining models. Information Sciences. 2013;225:1-17.

2. IDENTIFICATION OF METALLURGICAL PHASES IN STRUCTURAL STEEL USING GLCM TEXTURAL FEATURES [1]

2.1. Introduction

It is well known that the macroscopic mechanical properties of any material are governed by its underlying microstructure [1]. Most of the engineering metallic alloys like dual-phase steels, α-β brass, α-β titanium, etc., possess multi-phase polycrystalline microstructures [2]. Such microstructures are characterized by determination of the grain sizes, distinct phases, and their volume and morphology [3]. Under different mechanical and thermal manufacturing and operating conditions, these microstructural features undergo changes resulting in modified bulk properties of the metal [4]. Analysis of such information results in the establishment of a relationship between microstructural features and the bulk material properties that will guide engineers design components for specialized applications (ex: Hall-Petch relation [5, 6]). In general, material characterization techniques such as X-ray, neutron and electron diffraction, light optical microscopy, and electron and ion beam microscopy are employed to investigate and quantify the microstructural features of metals at various length scales [1]. Some of these techniques are time-consuming and expensive, and hence researchers often resort to light optical microscopy for performing tasks such as metallurgical phase identification and evaluation of grain sizes. The images obtained from the light-optical microscope are then analyzed manually following the standard protocols provided by the ASTM standards E114 [7] and E562 [8].

[1] This chapter is based on the paper "Texture-Based Metallurgical Phase Identification in Structural Steels: A Supervised Machine Learning Approach". *Metals*. 2019; 9(5):546. https://doi.org/10.3390/met9050546. The material in this chapter was co-authored by Dayakar Naik Lavadiya (DNL), Ravi Kiran Yellavajjala (RK) and Hizb Ullah Sajid (HUS). Contributions of authors are as follows: Conceptualization, D.L.N. and R.K.; Formal analysis, D.L.N.; Funding acquisition, R.K.; Investigation, R.K.; Methodology, D.L.N., H.U.S. and R.K.; Project administration, R.K.; Resources, H.U.S. and R.K.; Software, D.L.N.; Supervision, R.K.; Validation, D.L.N.; Writing—original draft, D.L.N.; Writing—review & editing, D.L.N. and R.K.

However, this process is labor intensive and a subjective process prone to poor repeatability and interpretation of results [9]. Therefore, automated digital image processing-based techniques are developed in recent years to overcome these issues and accurately quantify the microstructural features for better design of engineering components.

Image segmentation is a digital image processing technique that is widely used in the fields of engineering, medicine, food science, remote sensing etc. to identify the distinct regions/objects in an image that possess distinguishable visual characteristics or features [10]. Grayscale level, color, contrast, spectral values or textural features are some examples of such distinguishing features [11]. In general, two types of approaches are employed for segmentation of images namely discontinuity approach and similarity approach [12]. While discontinuity approach involves computation of abrupt changes or discontinuity of some object (ex. edges) in the image to identify distinct regions, similarity approach involves extraction and a one-to-one comparison of similar features (ex. pixel intensity) for identification of distinct regions. Techniques of discontinuity approach include – Sobel operator [13], Laplacian of Gaussian (LoG) operator [14], Laplacian operator [15] and canny operator [15] and techniques of similarity approach include – histogram based thresholding [16], region splitting and merging [17], level-set [18], clustering, and water shedding [11]. Among these techniques, histogram-based thresholding or Otsu's method [16] is extensively used for image analysis or segmentation of microstructure [19]. In this technique, a threshold-based criterion is established from the multimodal histogram of pixel intensities which is used for the segmentation of distinct phases. Implementation of such a technique results in an accurate segmentation of metallurgical phases whose pixel intensity distribution are distinct (multimodal) and do not overlap significantly (see Figure 1.5). However, when there are multiple metallurgical phases whose pixel intensity

distribution overlaps with each other (see Figure 1.5), employing histogram-based thresholding may lead to misclassification of phases.

Automated image segmentation procedures have been proposed and developed in recent years by numerous researchers to identify and quantify microstructural features [20]. Zhang and Liu [21] implemented canny edge algorithm for identification of phases in Ti-6Al-4V titanium alloy which involved a series of preprocessing steps such as noise removal and uneven illumination. Campbell et al. [9] developed a watershed algorithm based technique with pre- and post-processing steps to segment the touching grains in titanium (Ti-6Al-4V) microstructure. An automated microstructural characterization software MiPAR® was developed by Sosa et al. [19] that includes a segmentation module built based on the thresholding methods. Other notable works which employed threshold-based technique include segmentation of ferritic-martensitic dual phase steel by Burikova et al. [22] and identification of bainite in Fe-C-Mo steel by Ontman et al [23]. In addition to these studies, artificial intelligence based image segmentation is also found in literature which includes clustering [24], neural networks [25], fuzzy logic [26] and support vector machines (SVM) [27] methods for identification of microstructures in various metals. A sophisticated image processing technique that accounts for additional distinguishing features is required for accurate phase identification in a microstructure with phases that have overlapping pixel intensities.

In this chapter the distinct metallurgical phases of heat treated ASTM A36 steels is identified based on the textural features and pixel intensities of individual phases. To this end, the microstructural images of metallographic specimens are acquired using an optical microscope and the textural features and pixel intensities of distinct metallurgical phases are extracted from gray level co-occurrence matrix (GLCM) of each phase. Supervised machine

learning classifiers are employed for identification of metallurgical phases and following four classifiers are employed for this purpose: (1) Naïve Bayes, (2) K-Nearest Neighbor, (3) Linear Discriminant Analysis, and (4) Decision Tree. All four classifiers are trained with the extracted textural features and pixel intensities (of distinct metallurgical phases) and then deployed to identify the unknown phases in the microstructure. The rest of the chapter is organized as follows: textural feature extraction method is explained in Section 2.2, brief overview of supervised machine learning classifiers is provided in Section 2.3, materials and methodology adopted in this study is described in Section 2.4, feature selection method is explained in Section 2.5 and the validation of results are discussed in Section 2.6, and a summary of the study is provided in Section 2.7.

2.2. Texture

An object is considered to possess texture if its appearance is composed of patterns defined by variations in brightness and color. Objects can either have natural or manmade textures or a combination of both. While wood, soil, grass, etc. are some examples of natural texture possessing objects, carpet, brick walls, concrete, etc. are some examples of humanmade objects with distinctive textures. Human vision perceives the texture of different objects by sensing the variations in brightness and color and hence can use this information to discriminate one object from the other. However, in computer vision, a set of metrics are required to quantify the texture of an object. These set of metrics are often referred to as image-textural features and are extracted from a given digital image through image processing techniques. Image textural features are widely used for segmentation of different objects in an image for object/image identification and/ or classification purposes.

Two methods are commonly employed for quantifying the image texture: 1) structural approach and 2) statistical approach. While the primitives or repetitive elements and their placement rules are obtained in structural approach to describe the texture, non-deterministic properties obtained from the distribution of grayscale levels of a region of an image is used in statistical approach [28]. These non-deterministic properties are referred to as textural features. The structural approach is more suitable for regular textural patterns (for example checkerboard patterns, carpet textures, etc.) as it considers the hierarchy of spatial arrangement of primitives and statistical approach is more suitable for arbitrary textures (for example sand, concrete, etc.). In the context of this study, a statistical approach is adopted owing to the unstructured visual pattern (non-repetitive pattern) exhibited by the steel microstructure to identify the constituent metallurgical phases. In this study, the metallurgical phases of ASTM A36 steel, namely ferrite, pearlite, and martensite are assumed to possess unique textures. The statistical features quantifying the textures of each metallurgical phase are determined in this study by employing the gray-level co-occurrence matrix [29]. A detailed description of the GLCM method and extraction of textural features is provided next.

2.2.1. Gray Level Co-occurrence Matrix (GLCM) and Textural Features

Let us consider an image domain ($\boldsymbol{\Omega}$) that consists of N_x and N_y number of pixels in x and y directions, respectively. Pixel (short form for "picture element") is the basic logic unit in a digital image which has a rectangular or a square shape and has a unique location attached to it. Location of a pixel in the domain $\boldsymbol{\Omega}$ is denoted by ω_{ij}, where $i = 1 \ldots N_y$ and $j = 1 \ldots . N_x$ represents the corresponding row and column numbers of the pixel image grid respectively. Each pixel in an image is associated with an intensity $I(\omega_{ij}) = I_{ij}$ where $I_{ij} \in \mathbb{Z}_+^{N_y \times N_x}$. In an 8-bit grayscale image, which is the case in the current study, any pixel can have $2^8 = 256$ intensity

levels ($N_g = 256$). In fact, for the problem at hand, it is not necessary to consider 256 intensity levels. Instead, 8 intensity levels or grayscale levels are sufficient. The process of converting 256 grayscale levels to 8 grayscale levels is referred to as quantization and in this study this operation is accomplished through an in-built command 'imquantize' available in MATLAB®. The original ($N_g = 256$) and modified/ quantized images ($N_g = 8$) of microstructure are presented in Figure 2.1. Using $N_g = 8$, instead of $N_g = 256$ will lead to substantial savings in computational time without loss of visual information as demonstrated in Figure 2.1.

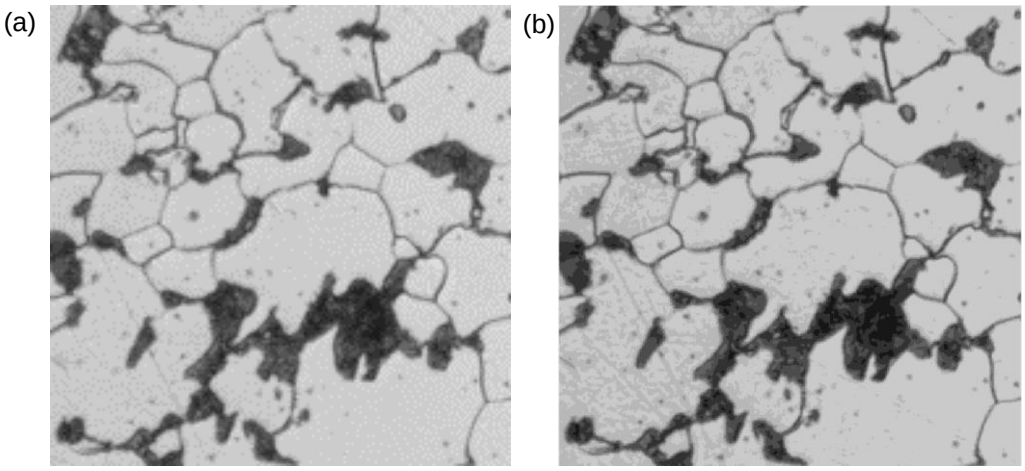


Figure 2.1. Illustration of (a) original image with 256 grey scale levels and (b) quantized image with 8 gray scale levels.

GLCM ($\boldsymbol{G}$) is a square matrix whose size is determined by the number of grayscales present in an image ($\boldsymbol{G} \in \mathbb{Z}_{+}^{N_g \times N_g}; N_g = 8$) and is independent of the number of pixels in the x and y directions in the image domain ($\boldsymbol{\Omega}$). Each element of the GLCM (G_{mn}) represents the count of pixel pairs that are separated by d number of pixels in the θ direction wherein one pixel has an intensity m and the other pixel has an intensity n. The mathematical definition of a typical element G_{mn} in a GLCM ($\boldsymbol{G}$) for $\theta = 0°$ or $180°$ is given as follows

$$G_{mn} = \#(\omega_{ab}, \omega_{ef} \in \Omega |\ a - e = 0, |b - f| = d, \big(I(\omega_{ab}), I(\omega_{ef})\big) = (m, n)\ or\ (n, m)) \tag{2.1}$$

where, $a, e \in \{1,2,3 \ldots . N_y\}$, $b, f \in \{1,2,3 \ldots . N_x\}$, $m, n \in \{1,2,3, \ldots N_g\}$, the condition $a - e = 0$ signifies the fact that pixels ω_{ab} and ω_{ef} are in the same row, i.e. $\theta = 0°$ or $180°$ and the condition $|b - f| = d$ has two consequences, 1) the pixels ω_{ab} and ω_{ef} are separated by d number of pixels and 2) the order of the pixels (bidirectional) do not have any impact on the value of G_{mn} ($\theta = 0°$ or $180°$). The definitions of GLCM for $\theta = 45°, 90°, 135°, 225°, 270°$ and $315°$ can be found elsewhere [29]. In the current study, the value of d and θ are fixed to be 1 and $0°/180^0$ respectively in order to estimate the GLCM for various metallurgical phases present in the microstructural image (ASTM A36 steel).

The GLCM ($\boldsymbol{G}$) evaluated using the Eq. (2.1) is referred to as unnormalized GLCM and a typical element G_{ij}^N in the normalized GLCM ($\boldsymbol{G}^N$) is defined as follows

$$G_{ij}^N(\theta, d) = \frac{G_{ij}(\theta, d)}{\sum_{i=1}^{8} \sum_{j=1}^{8} G_{ij}(\theta, d)} \tag{2.2}$$

Each element (G_{ij}^N) of the normalized GLCM is a measure of the joint probability occurrence of pixel pairs that are separated by distance d in the θ direction such that the grayscale levels of the first and second pixel match with row and column numbers of GLCM, respectively. Detailed examples for the evaluation of normalized and unnormalized GLCM is provided elsewhere [29].

As previously mentioned, the image texture is quantified by the textural features. In the statistical approach, textural features are the second order statistics extracted from the normalized GLCM ($\boldsymbol{G}^N$). In total, there are 19 textural features that can be extracted from the

GLCM which were proposed by various researchers in the past. Among the 19 textural features, 14 were proposed by Haralick et. al. [29] and 5 other textural features were proposed by Soh et. al. [30]. The mathematical definitions of these textural features are summarized in Table 2.1. While some of these textural features are easy to interpret (for example homogeneity, entropy, contrast, correlation), the rest of them are purely mathematical in nature and are difficult to interpret. In this study, for a given digital image domain $\boldsymbol{\Omega}$, textural features are evaluated at every pixel. In fact, a pixel does not possess texture. Hence, a window of $S \times S$ pixels, where S is the number of pixels ($S \leq N_x$ and N_y) is used to evaluate the textural features and these textural features are assigned to the pixel located at the center of this window. The calculated textural features are then used to classify a pixel in to one of the metallurgical phases. This classification is performed using machine learning algorithms which are discussed next.

Table 2.1. Textural features from Gray Level Co-occurrence Matrix (GLCM) [31].

Notation	Texture Feature	Equation
T_1	Auto correlation	$\sum_i \sum_j (i.j)p(i,j)$
T_2	Contrast	$\sum_i \sum_j \lvert i-j \rvert^2 p(i,j)$
T_3	Cluster prominence	$\sum_i \sum_j \left(i+j-\mu_x-\mu_y\right)^4 p(i,j)$
T_4	Cluster shade	$\sum_i \sum_j \left(i+j-\mu_x-\mu_y\right)^3 p(i,j)$
T_5	Dissimilarity	$\sum_i \sum_j \lvert (i-j) \rvert p(i,j)$
T_6	Energy	$\sum_i \sum_j p(i,j)^2$
T_7	Entropy	$-\sum_i \sum_j p(i,j).\log\left(p(i,j)\right)$
T_8	Homogeneity	$\sum_i \sum_j \frac{p(i,j)}{1+\lvert i-j \rvert}$
T_9	Maximum probability	$\max p(i,j)$
T_{10}	Sum of squares	$\sum_i \sum_j (1-v)^2 p(i,j)$
T_{11}	Sum average	$\sum_{i=2}^{2L} i.p_{x+y}(i)$
T_{12}	Sum entropy	$-\sum_{i=2}^{2L} p_{x+y}(i).\log\left(p_{x+y}(i)\right)$
T_{13}	Sum variance	$\sum_{i=2}^{2L} (i-T_{12})^2.p_{x+y}(i)$
T_{14}	Difference variance	$\sum_{i=0}^{L-1} i^2.p_{x-y}(i)$
T_{15}	Difference entropy	$-\sum_{i=0}^{L-1} p_{x-y}(i).\log\left(p_{x-y}(i)\right)$
T_{16}	Information measure of correlation I	$\frac{H_{XY}-H_{XY1}}{\max\left(H_X,H_Y\right)}$
T_{17}	Inverse difference normalized	$\sum_i \sum_j \frac{p(i,j)}{1+\frac{\lvert i-j \rvert}{L}}$
T_{18}	Inverse difference moment normalized	$\sum_i \sum_j \frac{p(i,j)}{1+\frac{\lvert i-j \rvert^2}{L^2}}$

Note: L denotes quantization levels of gray scale; $p(i,j)$ denotes the $(i,j)^{\text{th}}$ entry of co-occurrence probability matrix; $\sum_i(.)$ and $\sum_j(.)$ are $\sum_{i=1}^{L}(.)$ and $\sum_{j=1}^{L}(.)$ respectively; $\mu_x = \sum_{i=1}^{L}\sum_{j=1}^{L} i.p(i,j)$; $\mu_y = \sum_{i=1}^{L}\sum_{j=1}^{L} j.p(i,j)$; $v =$ mean value of $p(i,j)$. $p_{x+y} = \sum_{i=1}^{L}\sum_{j=1}^{L} p(i,j)$ for $i+j=k$; $p_{x-y} = \sum_{i=1}^{L}\sum_{j=1}^{L} p(i,j)$ for $\lvert i-j \rvert = k$; $p_x(i) = \sum_{j=1}^{L} p(i,j)$; $p_y(j) = \sum_{i=1}^{L} p(i,j)$; $H_X =$ Entropy of p_x; $H_Y =$ Entropy of p_y; $H_{XY} =$ Entropy of $p(i,j)$; $H_{XY1} = \sum_{i=1}^{L}\sum_{j=1}^{L} p(i,j).\log\left(p_x(i)p_y(j)\right.$.

2.3. Supervised Machine Learning

Supervised machine learning is a branch of machine learning (ML) that is used for performing classification and regression tasks on labeled data [32]. Labeled data is the data gathered from experiments or observations whose outcomes are known. The factors that govern the outcome of an observation or an experiment are referred to as descriptive features and the variable(s) that quantify the outcome is/ are referred to as response/target variable(s). The values

of descriptive features and response/target variables in a dataset can be either categorical (nominal/ ordinal) or numerical (discrete/ continuous). Supervised machine learning involves three important steps: 1) gathering the data; 2) training the machine learning algorithm, and 3) testing and deployment of the machine learning algorithm for the intended purpose. Data are the workhorses of machine learning algorithms and gathering of high quality and quantity of labeled data is the first step in supervised machine learning. The quality and quantity of labelled data are very crucial for improving the predictive power of ML algorithms. An elaborate discussion on the influence of the quality and quantity of data on the predictive power of ML algorithms can be found elsewhere [33]. The gathered data is partitioned into two datasets, namely training dataset and test dataset, at the beginning of the second step. Choosing a partition ratio of 80:20 is very common, where the numbers 80 and 20 represents the percentage of gathered data used for training and testing purposes, respectively. The other methods of partitioning the data and associated issues are discussed in reference [34]. Followed by the data partition, a machine learning algorithm is employed to learn the patterns, relationships and/ or dependencies from the obtained training dataset in the second step. The efficacy of the trained algorithm is then tested on the testing data (also called as validation dataset) in the third step. If the prediction accuracy is satisfactory, then the trained algorithm will be deployed for performing classification or regression tasks on the new data. From this point of discussion in this chapter, the term supervised machine learning will be replaced by machine learning for the sake of brevity. In the context of this study, machine learning based classification algorithms will be used to learn the textures of metallurgical phases of ASTM A36 steel and then will be deployed to classify them accurately into different phases. To this end, four machine learning algorithms are employed: (1) Naïve Bayes (NB) classifier, (2) K-Nearest Neighbors (K-NN) algorithm, (3) Linear

Discriminant Analysis (LDA) and (4) Decision Tree (DT) classifier. The coding of these algorithms is carried out in an in-house MATLAB® based machine learning software. The basic nomenclature employed for describing the data is provided next.

The master dataset is denoted by $\boldsymbol{D} \in \mathbb{R}^{p \times r}$, where p is the number of observations or in this context the number of gathered image domains of metallurgical phases and $r = q + 1$, where, q is the number of descriptive textural features ($q = 20 -$ pixel intensity along with 19 textural features); for definitions of textural features see Table 2.1. The last (r^{th}) column of $\boldsymbol{D}$ has the outcome of the experiment or the target variable which in this case is the metallurgical phase (ferrite, pearlite or martensite). Each of the p^{th} row of $\boldsymbol{D}$ is denoted by an instance vector $\boldsymbol{x}_j = (x_{j1}, x_{j2}, \dots, x_{jq}, x_{jr})$, where j takes the values from 1 to p, and $p = 735$ in this study. Here $x_{j1}, x_{j2}, \dots, x_{jq}$ are the descriptive textural features corresponding to the metallurgical phase identified by x_{jr}. In this study, 80% of the $p = 735$ number of observations are randomly chosen for training purposes and 20% are chosen for testing purposes. The instance vectors used for training purposes will be designated as $\boldsymbol{x}_j^*$ and the vectors used for testing (validation) purposes will be designated as $\boldsymbol{x}_j^{\#}$. Note that the range of j in the training and testing set is not same as that of the master dataset $\boldsymbol{D}$ and depends on the fractions of data used for training and testing purposes.

2.3.1. Naïve Bayes

Naïve Bayes is a probabilistic classifier that is derived from Bayes' theorem [35]. Bayes' theorem defines the conditional probability of an event A given event B as follows

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \tag{2.3}$$

where, $P(A|B)$ and $P(B|A)$ are conditional or posterior probabilities, $P(A)$ and $P(B)$ are prior probabilities. Using this definition, for any given i^{th} observation in the training dataset

$$P\left(x_{ir} = C_j | x_{i1}, x_{i2}, \dots, x_{iq}\right) = \frac{P\left(x_{i1}, x_{i2}, \dots, x_{iq} \middle| x_{ir} = C_j\right) \times P(x_{ir} = C_j)}{P(x_{i1}, x_{i2}, \dots, x_{iq})} \tag{2.4}$$

where, $C_{j=1,2,3}$ is the class label for the given descriptive textural features, where C_1, C_2 and C_3 are ferrite, pearlite and martensite, respectively. The joint conditional probability term in the numerator of Eq. (2.4) can be rewritten using the chain rule of probability [36, 37] as follows

$$\begin{aligned} &P\left(x_{i1}, \dots, x_{iq} \middle| x_{ir} = C_j\right) \\ &\quad = P\left(x_{i1} \middle| x_{ir} = C_j\right) \times P\left(x_{i2} \middle| x_{i1}, x_{ir} = C_j\right) \times \dots . \\ &\quad \times P\left(x_{iq} \middle| x_{i1}, \dots, x_{iq}, x_{ir} = C_j\right) \end{aligned} \tag{2.5}$$

Similarly, the denominator can also be written as

$$P\left(x_{i1}, x_{i2}, \dots, x_{iq}\right) = P(x_{i1}) \times P(x_{i2}|x_{i1}) \times \dots . \times P(x_{iq}|x_{i1}, x_{i2}, \dots, x_{iq-1}) \tag{2.6}$$

Substituting Eq. (2.5) and Eq. (2.6)in Eq. (2.4), the following expression is obtained.

$$\begin{aligned} &P\left(x_{ir} = C_j | x_{i1}, x_{i2}, \dots, x_{iq}\right) \\ &= \frac{P\left(x_{i1} \middle| x_{ir} = C_j\right) \times P\left(x_{i2} \middle| x_{i1}, x_{ir} = C_j\right) \times \dots . \times P\left(x_{iq} \middle| x_{i1}, \dots, x_{iq}, x_{ir} = C_j\right) \times P(x_{ir} = C_j)}{P(x_{i1}) \times P(x_{i2}|x_{i1}) \times \dots . \times P(x_{iq}|x_{i1}, x_{i2}, \dots, x_{iq-1})} \end{aligned} \tag{2.7}$$

Here, it is important to note that the denominator is independent of the class label (C_j) and will only serve as a normalizing constant. Hence, the denominator can be dropped without loss of any classification information. Now the left hand side of Eq. (2.7) represents a measure of conditional probability and hence denoted by $M\left(x_{ir} = C_j | x_{i1}, x_{i2}, \dots, x_{iq}\right)$

$$M\left(x_{ir} = C_j | x_{i1}, x_{i2}, \dots, x_{iq}\right)$$
$$= P\left(x_{i1} \middle| x_{ir} = C_j\right) \times P\left(x_{i2} \middle| x_{i1}, x_{ir} = C_j\right) \times \dots . \quad (2.8)$$
$$\times P\left(x_{iq} \middle| x_{i1}, \dots, x_{iq-1}, x_{ir} = C_j\right) \times P(x_{ir} = C_j)$$

With an increase in the number of descriptive features, the evaluation of conditional probabilities in Eq. (2.8) becomes computationally expensive. To circumvent this computational issue, the Naïve Bayes algorithm assumes that the descriptive features are conditionally independent. As a consequence of conditional independence, Eq. (2.8) can be rewritten as follows

$$M\left(x_{ir} = C_j | x_{i1}, x_{i2}, \dots, x_{iq}\right)$$
$$= P\left(x_{i1} \middle| x_{ir} = C_j\right) \times P\left(x_{i2} \middle| x_{ir} = C_j\right) \times \dots . \times P\left(x_{iq} \middle| x_{ir} = C_j\right) \times P(x_{ir} \quad (2.9)$$
$$= C_j)$$

The output class label $(C_{j=1,2,3})$ for a given instance $(x_{i1}^{\#}, x_{i2}^{\#}, \dots, x_{iq}^{\#})$ belongs to the class which maximizes the value of M. The posterior probabilities and the prior probability of a class will be evaluated from the training dataset and will be used directly in the above equation to find the class label (metallurgical phase) when the textural features for an unknown metallurgical phase are provided. It is important to note that the textural features of the metallurgical phases extracted in this study are continuous variables. Therefore, to evaluate the conditional probabilities of these textural features conditional probability density functions are used as proxy measures in the place of actual conditional probabilities in Eq. (2.9). The probability density functions that are considered in this study are normal distribution and Weibull distribution.

2.3.2. K-Nearest Neighbor

K-nearest neighbor (K-NN) classifier is a non-parametric, instance-based classifier which determines the class label of $\boldsymbol{x}^{\#}$ based on the assumption that the instances belonging to the

same class are found in the close proximity to each other when a consistent measure of proximity is employed. In other words, the class label of $\boldsymbol{x}^{\#}$ will be the same as that of the class label shared by its nearest neighboring instances/observations $\boldsymbol{x}_j^*$. In order to quantify the proximity between the training instance $\boldsymbol{x}_j^*$ and test instance $\boldsymbol{x}^{\#}$ and identify the nearest neighbor instances, generally a distance metric d is employed. Note that this distance d is different from the one that is used to represent the pixel distance in GLCM matrix evaluation. Although there are numerous distance metrics available (see [38]), Euclidean distance d is used in this study as it is most commonly used distance metric for continuous descriptive features which is defined as follows

$$d\left(\boldsymbol{x}_j^*, \boldsymbol{x}^{\#}\right) = \| \boldsymbol{x}_j^* - \boldsymbol{x}^{\#} \|_2 , \forall j = 1,2, \dots, \mathrm{p} \tag{2.10}$$

where $\| . \|_2$ is the l_2 norm and p is the number of instances in the training set.

The class label $x_{ir}^{\#}$ corresponding to an instance $\boldsymbol{x}^{\#}$ is then determined as the most frequent class label among the K nearest neighboring instances. In this study, a general rule of thumb, $K = \lceil \sqrt{\mathrm{p}} \rceil$ is used to estimate the value of K, where p is the number of training instances and the ceil operator $\lceil . \rceil$ rounds any positive number to the nearest integer which is greater than equal to the number on which the operator is used.

2.3.3. Linear Discriminant Analysis

Linear discriminant analysis (LDA) classifier employs a discriminant score function $L_j(\boldsymbol{x}^{\#})$ to predict the class labels of a given instance $\boldsymbol{x}^{\#}$. The discriminant score for each class is derived based on the Bayes' theorem provided in Eq. (2.3) and Eq. (2.4). Ignoring the denominator in the Eq. (2.4) as it is independent of the class label we get a measure of conditional probability which can be written as

$$M(x_{ir} = C_j | x_{i1}, x_{i2}, \dots, x_{iq}) = P(x_{i1}, x_{i2}, \dots, x_{iq} | x_{ir} = C_j) \times P(x_{ir} = C_j) \quad (2.11)$$

In fact, the class that maximizes the value of M for a given $\boldsymbol{x}$ is the class of $\boldsymbol{x}$ and the above equation is referred to as Bayes' classifier. However, evaluating the conditional probability in Bayes' classifier is challenging. In the case of linear discriminant analysis, all the instances that belong to a class C_j are assumed to be sampled from a multivariate normal distribution $\mathcal{N}(\boldsymbol{\Sigma}, \boldsymbol{\mu}_j)$, where, $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}_j$ are the covariance matrix and mean vector of all features in the instances that belong to class C_j. The probability density function is given as

$$f(\boldsymbol{x}|C_j) = \frac{1}{\sqrt{(2\pi)^q |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_j)' \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_j)\right)$$

By substituting $f(\boldsymbol{x}|C_j)$ in the place of $P(\boldsymbol{x}|C_j)$ as a proxy measure, we get

$$M(x_{ir} = C_j | x_{i1}, x_{i2}, \dots, x_{iq}) = \frac{1}{\sqrt{(2\pi)^q |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_j)' \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_j)\right) \times P(x_{ir} = C_j) \quad (2.12)$$

Note that $P(\boldsymbol{x}|C_j)$ can be replaced by its proxy value $f(\boldsymbol{x}|C_j)$ as we are interested in discrimination of instances in to classes and are not interested in evaluating the actual probabilities. By applying logarithm on both sides we get the discriminant score function for class C_j as

$$L_j(\boldsymbol{x}) = -\frac{1}{2}\log((2\pi)^q |\boldsymbol{\Sigma}|) - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_j)' \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_j) + \log P(C_j) \quad (2.13)$$

Noting that $\boldsymbol{\Sigma}^{-1}$ is symmetric, i.e., $\boldsymbol{x}' \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_j = \boldsymbol{\mu}_j' \boldsymbol{\Sigma}^{-1} \boldsymbol{x}$, we can simplify the discriminant score function as follows

$$L_j(\boldsymbol{x}) = -\frac{1}{2}\log((2\pi)^q |\boldsymbol{\Sigma}|) - \frac{1}{2}\boldsymbol{\mu}_j' \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_j - \frac{1}{2}\boldsymbol{x}' \boldsymbol{\Sigma}^{-1} \boldsymbol{x} + \boldsymbol{\mu}_j' \boldsymbol{\Sigma}^{-1} \boldsymbol{x} + \log P(C_j)$$

By ignoring the terms that are independent of the class (as they do not improve the discriminative power of the algorithm), we obtain the discriminant score function for a class as

$$L_j(\boldsymbol{x}) = -\frac{1}{2}\boldsymbol{\mu}_j'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_j + \boldsymbol{\mu}_j'\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \log P(C_j) \tag{2.14}$$

In the case of linear discriminant analysis, the covariance matrices ($\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$ and $\boldsymbol{\Sigma}_3$) for all classes are assumed to be equal. In order to better capture the variances in the available dataset, a pooled covariance matrix defined as

$$\boldsymbol{\Sigma}_{pl} = \frac{1}{\mathrm{p} - m}\sum_{i=1}^{m}(\mathrm{p}_i - 1)\boldsymbol{\Sigma}_i$$

is used in the place of $\boldsymbol{\Sigma}$ modifying the discriminant score function as follows

$$L_j(\boldsymbol{x}) = -\frac{1}{2}\boldsymbol{\mu}_j'\boldsymbol{\Sigma}_{pl}^{-1}\boldsymbol{\mu}_j + \boldsymbol{\mu}_j'\boldsymbol{\Sigma}_{pl}^{-1}\boldsymbol{x} + \log P(C_j) \tag{2.15}$$

To predict the class label $x_{jr}^{\#}$ of a given test instance $\boldsymbol{x}^{\#}$, the discriminant scores $L_{j=1:m}(\boldsymbol{x}^{\#})$ are first evaluated for all the m class labels and then the index of the class label j that yields maximum L_j value is assigned as the class label for the test instance $\boldsymbol{x}^{\#}$. In the context of this study, the class label $j \in \{C_1, C_2, C_3\}$.

2.3.4. Decision Tree

Decision tree classifier is a non-parametric classifier that employs a hierarchical tree structure to predict the class label of an instance $\boldsymbol{x}^{\#}$. This tree structure is obtained as a result of recursive partitioning of the training dataset which yields a class label as an outcome at the end [36]. Basic decision tree architecture consists of a root node at the top, intermediate nodes in between and leaf nodes at the bottom. Each node (root and intermediate) in a tree represents a feature, each descending branch represents a criterion or a decision rule and the leaf nodes at the bottom represent the final outcomes or the classification labels. In this study, an ID3 (Iterative

Dichotomiser 3) decision tree algorithm [39] is used and the continuous textural features are treated according to the procedure employed in C4.5 algorithm [40] to create a decision tree. This algorithm employs entropy and information gain measures for partitioning the data in to a decision tree. Entropy is an information measure that characterizes the (im)purity or uncertainty of collected observations. Given a column vector of class labels $\boldsymbol{s}$ i.e. r^{th}column of training dataset, the entropy of the distribution of class labels denoted by $H(\boldsymbol{s})$ is evaluated as

$$H(\boldsymbol{s}) = -\sum_{i=1}^{m} P_i \log_2 P_i \tag{2.16}$$

where, P_i is the probability that an observation belongs to a class label C_i. Entropy is zero when all the classification labels belong to same class and is 1 when the classification labels are equally proportioned in to all classes. In order to evaluate the root node, the unique values of all textural features are sorted in to ascending order. The entropy reduction is calculated for every feature choosing each of its unique values. The feature ($\boldsymbol{x}_j = (x_{1j}, x_{2j}, \dots x_{kj}, \dots, x_{pj})$, p is the number of observations) and the unique value of that feature (x_{kj}) about which the partition is made that maximizes the information gain are taken as the root node and determining the corresponding decision rules for the first two branches of the decision tree. The information gain is defined as the expected reduction in entropy caused by partitioning the class label vector $\boldsymbol{s}$ with respect to the given j^{th} attribute vector $\boldsymbol{x}_j = (x_{1j}, x_{2j}, \dots x_{kj}, \dots, x_{pj})$ whose unique values are already sorted in to ascending order, where x_{kj} is the partition value. The information gain is defined as

$$U\left(\boldsymbol{s}, \boldsymbol{x}_j, x_{kj}\right) = H(\boldsymbol{s}) - \frac{|\boldsymbol{s}_l|}{|\boldsymbol{s}|} H(\boldsymbol{s}_l) - \frac{|\boldsymbol{s}_h|}{|\boldsymbol{s}|} H(\boldsymbol{s}_h) \tag{2.17}$$

where, $\boldsymbol{s}_l$ and $\boldsymbol{s}_h$ are subsets of $\boldsymbol{s}$ given as $\boldsymbol{s}_l = \{x_{1j}, x_{2j}, \ldots x_{kj}\}$ and $\boldsymbol{s}_h = \{x_{k+1j}, x_{k+2j}, \ldots x_{pj}\}$ and $|*|$ in this context is the cardinality of the subsets. While the first term on right hand side of the above equation represents the entropy of distribution of class labels before data partition, the second and the third terms represent the expected decrease in entropy after partitioning the data using attribute $\boldsymbol{x}_j$ about x_{kj} value. The intermediate nodes and their branches are generated similarly but by excluding the features that were already assigned to nodes previously. This procedure is carried out until each of the branches result in a class label as the output. Decision trees are prone to overfitting and hence tree pruning is recommended [41]. This is specifically true when there is large number of irrelevant and dependent features. As a feature selection algorithm is used in a pre-processing step, tree pruning is not performed in this study. The ID3 algorithm based on the training data provides a set of decision rules and a given instance $\boldsymbol{x}^{\#}$ from the test, dataset is taken through these decision rules to obtain its unknown class label $x_{ir}^{\#}$.

2.4. Methodology

The main objective of this study is to identify the distinct metallurgical phases present in the ASTM A36 steel using supervised machine learning classifiers. Following step-by-step procedure is adapted in the current study to accomplish this objective (see Figure 2.2): (1) acquisition of microstructural images of heat treated ASTM A36 steel metallurgical specimens, (2) splitting of the acquired images into training image set and test image set, (3) extraction of textural features for known metallurgical phases from training images and (4) extraction of textural features for unknown metallurgical phases from test images, (5) application of feature selection algorithm to select the most relevant textural features, (6) training the classifier with the selected relevant textural features, and (7) predicting the metallurgical phases in test image using a trained classifier. In this section, a detailed description of each step is provided.

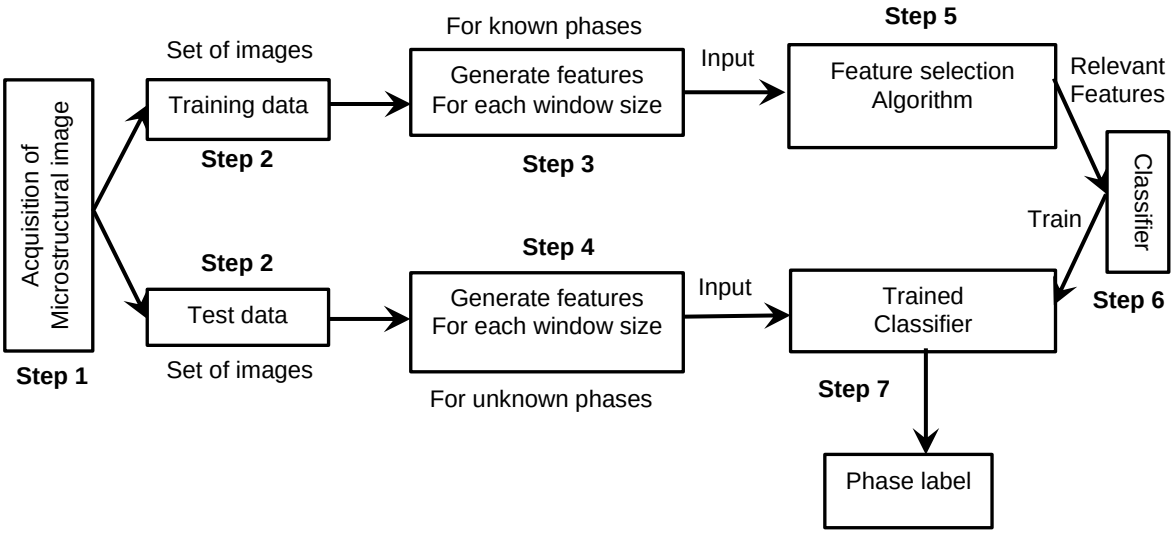


Figure 2.2. Flowchart of methodology for metallurgical phase identification in ASTM A36 steel using supervised machine learning.

Nine ASTM A36 metallographic specimens cooled from different elevated temperatures are chosen in this study. Of these nine specimens, six were heated to temperatures 500°C, 600°C, 700°C, 800°C, 900°C and 1000°C followed by air cooling, 2 specimens were heated to temperatures of 900°C and 1000°C followed by water cooling, and one specimen was extracted from as-received steel. Further details about the heat treatment of these specimens can be found elsewhere [42]. It is important to note that the heat treatment temperatures considered in this study are chosen to obtain various phase compositions in ASTM A36 steel. The air-cooled specimens have a ferrite-pearlite microstructure and the water-cooled specimens possess martensite ferrite microstructure. All the nine metallographic specimens are then examined under the Amscope® optical microscope at 50× magnification to acquire the microstructural images. In total, 45 images are acquired which includes five images each from all nine different metallographic specimens. Note that to obtain these five images from each metallographic specimen five different locations are chosen on the specimen.

In the second step, the images acquired for each specimen are divided into two sets of images, training image set and test image set. Out of the five images acquired for every metallographic specimen, three images are allocated to the training image set and the other two images are allocated to the test image set and this assignment was done randomly. By repeating this exercise for all 9 specimens, 27 images in total are generated for the training image set, and 18 images are generated for the test image set. In the third step, the textural features of each metallurgical phase are extracted from the images available in the training image set and a dataset $\boldsymbol{D}$ is generated. The generated dataset $\boldsymbol{D}$ consists of 735 number of data points in total, which includes 315 number of data points corresponding to ferrite, 270 number of data points corresponding to pearlite and 150 number of data points corresponding to martensite. Lower number of data points for pearlite and martensite can be attributed to the lesser number of images available for water cooled specimens. To extract the textural features of a metallurgical phase from an image, an in-house MATLAB® code is built in this study. This code allows the user to choose pixels randomly from an image that may correspond to any of the metallurgical phases. A schematic of pixel selection locations adopted in this study is shown in Figure 2.3. The textural features for each of these selected pixels are then extracted using GLCM explained in Section 2.2. To generate dataset $\boldsymbol{D}$ consisting of 735 number of data points, this exercise is repeated for all the 27 images present in the training image set. As mentioned in Section 2.2, a pixel does not possess a texture and hence a window size of $S \times S$ pixels is used to evaluate the textural features which are then assigned to the pixel located at the center of this window. As the ideal window size is not known a priori, following five window sizes are considered in this study:61×61, 81×81, 101×101, 121×121 and 161×161 pixels. The textural features are computed for all the five window sizes.

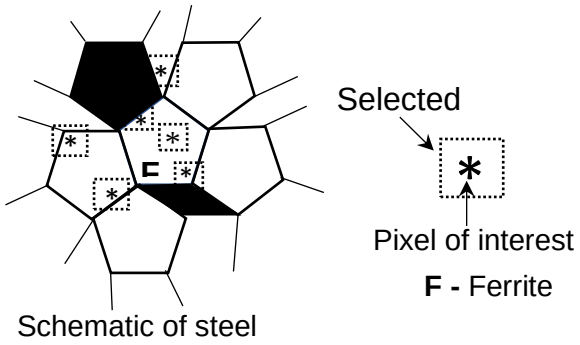


Figure 2.3. Schematic of pixel locations selected for extraction of textural features.

In the fourth step, the textural features of unknown metallurgical phases in the test images are evaluated using the same procedure described in the third step. These test images are selected from the 18 images that are present in the testing image set and not used for training purpose. However, in this step, every pixel of the test image is selected automatically by the MATLAB® code, because the metallurgical phases of these pixels are unknown and should be classified into one of the known phases. The dataset generated in this step is called as the test dataset. In the fifth step, feature selection is performed on the dataset ***D*** to obtain the subset of most relevant textural features. In the sixth step, the dataset ***D*** consisting of only most relevant textural features is provided as input to the classifier chosen in this study and the machine learning algorithm is trained. In the seventh and final step, the test dataset is given as input to the trained algorithm and unknown phase labels are identified. The flowchart of this methodology is illustrated in Figure 2.2.

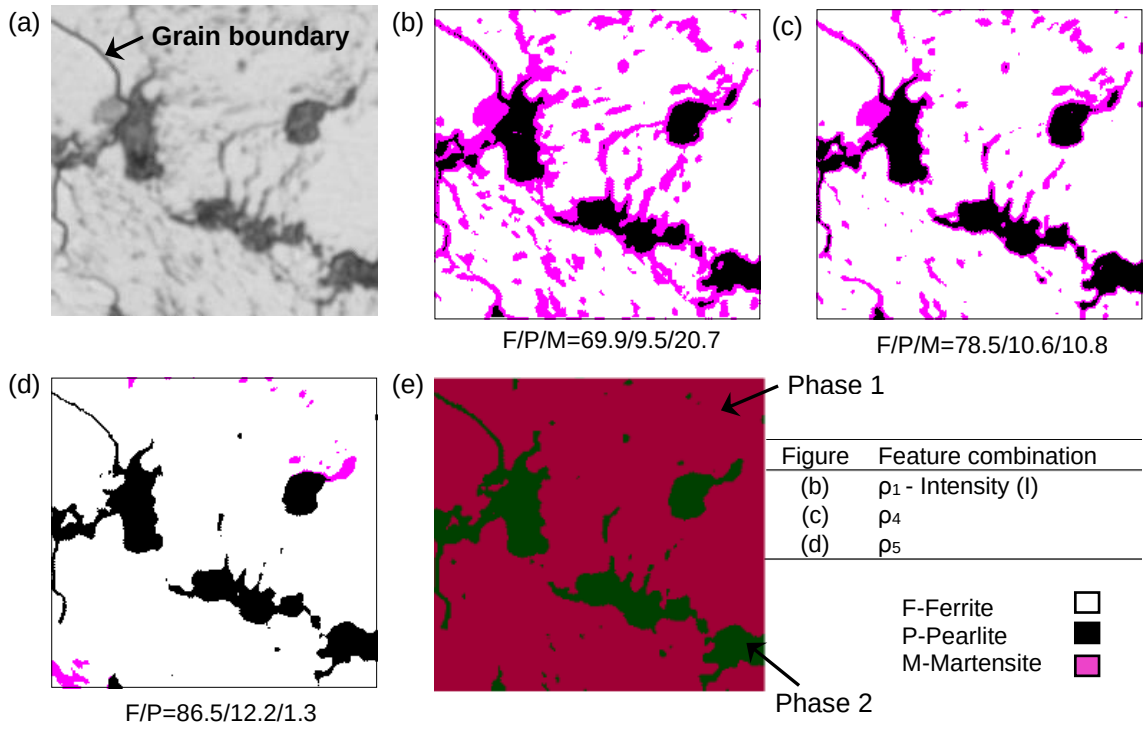


Figure 2.4. Numerical example illustrating the importance of selection of most relevant textural features (a) original image (b) only 'pixel intensity' (c) 'pixel intensity' and top 3 textural features (d) 'pixel intensity' and top 4 textural features and (e) segmentation in imageJ. It is observed from subfigure (d) that the accuracy of classifying metallurgical phases has increased with addition of more number of relevant textural features.

2.5. Feature Selection

Feature selection is the process of choosing a subset of features from the entire 20 textural features that are ideally necessary and sufficient for predicting the target variable (metallurgical phase) [43]. Feature selection eliminates the redundant and irrelevant features from the evaluated 20 textural features which otherwise will adversely impact the performance of a given machine learning classifier. In this study, feature selection is carried out using a filter approach that primarily involves two steps: (1) ranking of features and (2) selecting a subset of top-ranked relevant features. A brief description of these two steps is provided below.

2.5.1. Feature Ranking

Feature ranking is the process of assigning ranks to the descriptive features based on the scores that are estimated from one of the following: information measure, distance, similarity, consistency, statistical measures etc. [44]. These estimated scores represent the correlation/association/relevance between the feature and a target variable or class label. Higher the score, higher is the rank and stronger is the correlation. In this study, ReliefF algorithm [43, 45] is employed to estimate the ranking of textural features. This algorithm calculates a proxy statistic for each textural feature, called feature weight ($W_{i=1:q}$), which is used to estimate the relevance of the textural feature to the target variable or metallurgical phase [44]. To determine the feature weight W_i, ReliefF algorithm employs an iterative updating scheme for weights which is executed z number of times (z is the user defined parameter, $z \in \mathbb{Z}_+$) in the following three steps: (1) an instance $\boldsymbol{x}_{t=1:p}$ is sampled at random from the dataset $\boldsymbol{D}$ without replacement in the first step; (2) k (is a user defined parameter, $k \in \mathbb{Z}_+$) number of nearest instances of $\boldsymbol{x}_t$ are determined from each class in the second step and; (3) weights of each textural feature is estimated and updated using Eq. (2.18) in the third step. The set of k nearest instances that are identified in the second step are referred as nearest-hit instances $\boldsymbol{h}_j$, if the class label of k instances are same as that of the class label of $\boldsymbol{x}_t$, and are referred as nearest-miss instances $\boldsymbol{m}_j$ if the class label of the k instances differ from that of the class label of $\boldsymbol{x}_t$, where j takes the values from 1 to k. Based on the nearest-hit and nearest-miss instances identified in the second step, this algorithm rewards or penalizes the textural features by updating their weights using Eq. (2.18) which assigns higher weights to the features that are strongly correlated to the target variable when compared to the irrelevant features. The number of iterations z is an

arbitrary integer and is generally chosen to be $\lceil\sqrt{p}\,\rceil$. The equation to evaluate the weights of the feature is expressed as

$$W_i = W_i^{old} - \sum_{j=1}^{k} \frac{\text{diff}(i, x_{ti}, h_{ji})}{zk} + \sum_{C_o \neq \text{class of } \boldsymbol{x}_t} \frac{\text{P}(C_o)}{1 - \text{P}(\text{class of } \boldsymbol{x}_t)} \sum_{j=1}^{k} \frac{\text{diff}\left(i, x_{ti}, m_{ji}(C_o)\right)}{zk} \tag{2.18}$$

where, W_i on the left hand side is the updated weight of the textural feature i, W_i^{old} on right hand side is the initial or previous weight of the textural feature i, z is the number of iterations, x_{ti}, h_{ji}, m_{ji} are the values of textural feature i corresponding to the instance $\boldsymbol{x}_t$, near-hit instance $\boldsymbol{h}_j$, near-miss instance $\boldsymbol{m}_j$ respectively, C_o is the class label of an instance $\boldsymbol{m}_j$ which is not the same as that of class label of $\boldsymbol{x}_t$, P(C_o) is the prior probability of the class C_o and both diff(i, x_{ti}, h_{ji}) and diff(i, x_{ti}, m_{ji}) evaluates the difference between the normalized values of i^{th} feature expressed as

$$\text{diff}(i, x_{ti}, h_{ji}) = \frac{|x_{ti} - h_{ji}|}{\max(i) - \min(i)} \text{ and, } \text{diff}(i, x_{ti}, m_{ji}) = \frac{|x_{ti} - m_{ji}|}{\max(i) - \min(i)}$$

A more detailed description of the algorithm can be found elsewhere [43, 45]. The dataset $\boldsymbol{D}$ generated in Section 2.4 is given as an input to the ReliefF algorithm and the rank of the descriptive features are obtained (see Table 2.2). This procedure is repeated for all five different window sizes that are considered in this study.

Table 2.2. Feature ranking based on ReliefF algorithm.

Rank	61x61	81x81	101x101	121x121	161x161
1	Intensity	Intensity	Intensity	Intensity	Intensity
2	Maximum probability	Maximum probability	Maximum probability	Maximum probability	Maximum probability
3	Auto-correlation	Sum of squares	Sum of squares	Cluster shade	Sum of squares
4	Sum of squares	Auto-correlation	Auto-correlation	Inverse correlation	Auto-correlation
5	Entropy	Sum variance	Sum variance	Sum of squares	Cluster shade
6	Sum variance	Energy	Inverse correlation	Auto-correlation	Sum variance
7	Cluster shade	Cluster shade	Cluster shade	Energy	Sum average
8	Sum average	Sum average	Energy	Sum variance	Inverse correlation
9	Inverse difference moment	Sum entropy	Sum average	Inverse difference normalized	Inverse difference normalized
10	Sum entropy	Entropy	Inverse difference normalized	Sum average	Energy

2.5.2. Selection of Feature Subset

Followed by a ranking of textural features, a subset of relevant textural features is selected in the second step of the feature selection process. In this study, as the number of relevant features is not known a priori, a trial and error approach is used. In this approach, a particular combination of relevant textural features is chosen in each trial and the resulting misclassification error for each classifier (from the selected combination) is evaluated. The combination of relevant textural features that minimized the misclassification error is then selected as the most relevant subset of textural features in this study. To select a combination of relevant textural features during each trail, following procedure is adapted: in the first trial only one relevant descriptive feature is selected that has the highest relevance index or rank of 1 (see Table 2.4); in the second trial the next relevant descriptive feature with rank 2 is included along with the previous one; in the third trail the third relevant feature is included with the previous two descriptive features and this procedure is repeated for each trial. In other words, for each trial, the next most relevant descriptive feature is added successively. Note that these combinations are found to differ for different window sizes.

2.5.3. Performance Assessment of Classifier

To evaluate the misclassification error or assess the performance of the classifier, dataset $\boldsymbol{D}$ is split into two subsets – one subset ($\boldsymbol{S}_1$) containing 80% of the observations and the other subset ($\boldsymbol{S}_2$) containing rest 20% of the observations. To obtain the subset $\boldsymbol{S}_1$, the observations available in the dataset $\boldsymbol{D}$ are randomly sampled without replacement using the 'datasample' function available in MATLAB®. To avoid the bias between class labels, the sampling was carried out such that 80% of observations from each metallurgical phase was chosen from dataset $\boldsymbol{D}$. The rest of the observations are grouped in to subset $\boldsymbol{S}_2$. Here, the subset $\boldsymbol{S}_1$ is referred to as training dataset (for the classifier) and the subset $\boldsymbol{S}_2$ is referred to as validation dataset. In other words, the classifier is first trained using the subset $\boldsymbol{S}_1$ and then deployed to predict the class labels on subset $\boldsymbol{S}_2$. As the class labels in subset $\boldsymbol{S}_2$ are already known, the performance of classifier on a subset $\boldsymbol{S}_2$ can be assessed by cross validating with the known class labels. For this, the count of both correctly and incorrectly classified class labels are obtained and summarized in the form of a matrix called confusion matrix (see Table 2.3).

Table 2.3. Confusion matrix ($\boldsymbol{C}$).

		Predicted class label			
		C_1	C_2	$\cdots$	C_m
Actual class label	C_1	c_{11}	c_{12}	$\cdots$	c_{1m}
	C_2	c_{21}	c_{22}	$\cdots$	c_{2m}
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	C_m	c_{m1}	$\cdots$	$\cdots$	c_{mm}

C_1 – Ferrite

C_2 – Pearlite

C_3 – Martensite

A confusion matrix ($\boldsymbol{C}$) is a square matrix of size $m \times m$, where m is the number of class labels or metallurgical phases and each element c_{ij} of the matrix represents the frequency of instances from the validation dataset that are assigned class j by the classifier which in reality belongs to class i [46]. In other words, a confusion matrix provides the summary of correct and

incorrect classifications predicted by the classifier. Here $i, j \in \{C_1, C_2, C_3\}$ where C_1, C_2 and C_3denotes ferrite, pearlite and martensite respectively. While the summation of each row of the confusion matrix (see Table 2.3) represents the number of instances actually belonging to class i in the validation dataset, the summation of each column of the confusion matrix (see Table 2.3) represents the number of instances that are assigned to class i by the classifier.

Accuracy (A) is often used to quantify the predictive power of classifiers. It is the ratio of a total number of observations whose class labels are correctly predicted to the total number of observations present in the validation dataset. Mathematically, A is defined as

$$A = \frac{\sum_{i=1}^{m} c_{ii}}{\sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}} \times 100\% \tag{2.19}$$

However, the accuracy estimated from the above equation may be misleading when the dataset is imbalanced, i.e., the number of data points corresponding to each class label is not the same [47]. The training dataset $\boldsymbol{D}$ consists of 43.7% of data points corresponding to ferrite, 37.5% of data points corresponding to pearlite and 20.8% of data points corresponding to martensite. To assess the performance of such imbalanced datasets often F-measure (F_m) is used instead of accuracy (A). F-measure (F_m) is the harmonic mean of two other accuracy measures, namely precision (O) and recall (R). Precision is defined as the ratio of the number of observations whose class label i is correctly predicted by the classifier to the total number of observations that are assigned to the class i by the classifier and recall is defined as the proportion of observations of class i (in the validation dataset) that are correctly predicted as class i by the classifier [48] . Provided a confusion matrix $\boldsymbol{C}$, the precision with which a class i instance is classified is the ratio of number of correctly classified class i instances to the total

number of instances that are assigned to class i (i.e., the summation of corresponding column elements $\sum_{j=1}^{m} c_{ji}$) by the classifier. The recall of a class i is evaluated as the ratio of correctly classified number of instances with class label i to the total number of instances that actually belong to class i (i.e., the summation of corresponding row elements $\sum_{j=1}^{m} c_{ij}$). The overall precision (O) and overall recall (R) are then computed as the average of the precision and recall of all classes, respectively and are given as follows [48]

$$\mathrm{O} = \frac{1}{m}\sum_{i=1}^{m}\frac{c_{ii}}{\sum_{j=1}^{m} c_{ji}} \text{ and } \mathrm{R} = \frac{1}{m}\sum_{i=1}^{m}\frac{c_{ii}}{\sum_{j=1}^{m} c_{ij}}$$

Here, m represents the number of class labels. While overall precision and overall recall are also used as the measures of performance assessment for classifiers, F-measure (F_m) combines the trade-off between both overall precision and overall recall [47] and is evaluated as follows

$$\mathrm{F}_m = \frac{2 \times \mathrm{O} \times \mathrm{R}}{\mathrm{O} + \mathrm{R}} \times 100\% \tag{2.20}$$

2.6. Results

The results of textural feature ranking and the subsets of most relevant textural features for all classifiers is presented in this section. In addition to this, the performance assessment of all four classifiers for the selected subset of relevant textural features is summarized. A numerical example to demonstrate the importance of choosing relevant textural features is also provided in this section. Finally, the unknown metallurgical phases are identified in test images of ASTM A36-500AC, ASTM A36-900AC and ASTM A36-900WC for the sake of visual validation.

2.6.1. Feature Ranking

The textural features of the dataset $\boldsymbol{D}$ obtained in Section 2.4 are ranked using ReliefF algorithm and the top 10 relevant features are provided in Table 2.2. From Table 2.2, it is clear that the rank of the textural features depend on the choice of window size i.e. rank of the textural features changes with the change in window size. The change in the order of the feature rank can be attributed to the fact that the image texture is not similar to the one that is obtained when the window size is increased. While employing smaller window sizes fails to capture the texture of the metallurgical phase at the given magnification, larger windows will include too much redundant information that will result in a texture which might not be the true representation of a metallurgical phase. However, it is interesting to note that the 'pixel intensity' and 'maximum probability' remained as the top relevant features for all five window sizes considered in this study. While the pixel intensity is perceivable to the human eye as the relative brightness/ grayscale, 'maximum probability' is a statistical measure evaluated from GLCM matrix and has no physical meaning. Mathematically, 'maximum probability' texture feature is the maximum value of the joint probability of quantized grayscale level values for the pair of pixels that are separated by distance 'd' and are oriented at an angle 'θ'. Pixel intensity is undoubtedly a very important feature to identify the metallurgical phase. Indeed, many commercial thresholding methods solely rely on pixel intensity. The robustness of the method proposed in this study relies on adding textural features which may not be perceivable by an ordinary human eye but can be measured mathematically and can be used by a ML algorithm to classify/ identify unknown metallurgical phases. In addition to the above mentioned features, 'auto-correlation' and 'sum of squares' are found to be the next two relevant textural features for all window sizes except for the window size 121×121. A careful observation of next 6 relevant textural features reveals that

‘cluster shade’, ‘sum variance’, ‘sum average’ and ‘energy’ are among the top 10 textural features for all window sizes. Note that most of these textural features are purely mathematical in nature and hard to interpret visually.

2.6.2. Feature Subset Selection

The subset of the most relevant textural features obtained by adapting the procedure explained in the previous section is provided in Table 2.4. From Table 2.4, it is observed that the combinations of textural features differed for all five window sizes (see Table 2.2). This can be attributed to the fact that order of the ranks of the textural features change with the change in window size as explained in previous section. In this study, an ideal window size and the subset of relevant textural features (see Table 2.4) that will maximize the performance of each classifier is determined. Accuracy and F-measure are evaluated to assess the performance of each classifier and the results are provided in Table 2.5 to Table 2.8. From Table 2.5 to Table 2.8, it is clear that the window size of 161×161 yielded higher accuracy (>97%) and F-measure (>97%) for all four classifiers – Naïve Bayes, K-NN, LDA, and decision tree. Besides the accuracy and F-measure, Table 2.5 to Table 2.8 also provides the confusion matrices. Note that the window sizes larger than 161×161 are not considered in this study for two reasons that are already mentioned in previous section i.e. (1) larger window sizes will include too much redundant information and, (2) will increase the computational time and cost. From the confusion matrices provided in Table 2.5 to Table 2.8, following two insights can be drawn: (1) martensite phase was often misclassified into ferrite when only ‘pixel intensity’ was considered and (2) misclassification was reduced when relevant textural features are also considered along with the pixel intensity. It is clearly evident from the pixel intensity histogram provided in Figure 1.5 that the majority of pixel intensities belonging to ferrite and martensite phase are overlapped when compared to

pearlite phase alone. For this reason, the ferrite and martensite phases were observed to be often misclassified. However, with addition of textural features to the pixel intensity, the misclassification is observed to reduce for all the classifiers because the textural features are distinct for each metallurgical phase and do not overlap. Among all the four classifiers that are considered in this study, Naïve Bayes, LDA and decision tree classifiers are observed to exhibit more or less the same robustness with respect to the misclassifications i.e. they exhibited more or less same confusion matrices with various subset of textural features that were considered. However, K-NN classifier was observed to misclassify more number of martensitic phase pixels in to ferrite phase when compared to other classifiers. With addition of relevant textural features, the misclassification was observed to be reduced for K-NN classifier.

In conjunction with the ideal window size, a combination of textural features that resulted in higher F-measure for all four classifiers are also determined and provided in Table 2.9. The ideal window size and the combination of textural features determined for Naïve Bayes classifier is – 161×161 and ρ_3; for K-NN classifier is – 161×161 and ρ_5 ; for LDA classifier is – 101×101 and ρ_6 and; for decision tree classifier is – 161×161 and 'All features'. Although LDA classifier was observed to yield a higher accuracy for 161×161 window size, it is observed that there are other combinations of window sizes and textural features which yielded similar accuracy. Considering the confusion matrices provided in Table 2.7, a window size of 101×101 is chosen in this study as it is observed to produce a more reliable classification of metallurgical phases along with the feature combination ρ_6 (see 3rd row of Table 2.4). Unlike Naïve Bayes, K-NN and LDA classifier, no specific feature selection is carried out for decision tree classifier in the current study. Decision tree classifier is a low bias-high variance classifier and is sensitive to the small fluctuations in the training dataset. While the inclusion of redundant and irrelevant features

will decrease the performance of a high bias-low variance classifier (e.g. Naïve Bayes and LDA), it will only decrease the performance of a decision tree classifier when there is significant amount of noise in the training and test data [49], which is not the case in the current study i.e. training and test data did not have significant amount of noise. Therefore, no feature selection was performed for decision tree classifier. Here bias represents the error resulting from the erroneous assumptions made in building a machine learning classifier and variance represents the sensitivity to small fluctuations in the training dataset.

Table 2.4. Combinations of features for each window size.

Window size	**Combination**					
	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	ρ_6
161X161	Intensity	ρ_1+T_9	ρ_1+ T_{10}	ρ_2+ T_1	ρ_3+ T_4	ρ_4+ T_{13}
121X121	Intensity	ρ_1+T_9	ρ_1+ T_4	ρ_2+ T_{16}	ρ_3+ T_{10}	ρ_4+ T_1
101X101	Intensity	ρ_1+T_9	ρ_1+ T_{10}	ρ_2+ T_1	ρ_3+ T_{13}	ρ_4+ T_{16}
81X81	Intensity	ρ_1+T_9	ρ_1+ T_{10}	ρ_2+ T_1	ρ_3+ T_{13}	ρ_4+ T_6
61X61	Intensity	ρ_1+T_9	ρ_1+ T_1	ρ_2+ T_{10}	ρ_3+ T_7	ρ_4+ T_{13}

Table 2.5. Performance of Naïve Bayes classifier for different combinations of textural features.

Size	***C* for** ρ_2			$\mathbf{F_m}$	***C* for** ρ_3			$\mathbf{F_m}$	***C* for** ρ_4			$\mathbf{F_m}$	***C* for** ρ_5			$\mathbf{F_m}$	***C* for** ρ_6			$\mathbf{F_m}$
161	0.976	0.000	0.024		0.992	0.000	0.008		0.968	0.016	0.016		0.810	0.016	0.175		0.849	0.016	0.135	
	0.009	0.954	0.037	95.22	0.000	1.000	0.000	97.70	0.000	0.991	0.009	95.62	0.000	1.000	0.000	91.55	0.009	0.991	0.000	90.32
	0.067	0.000	0.933		0.067	0.033	0.900		0.033	0.067	0.900		0.000	0.017	0.983		0.050	0.050	0.900	
Acc.		95.91				97.62				96.26				91.50				91.16		
121	0.976	0.008	0.016		0.849	0.016	0.135		0.881	0.000	0.119		0.913	0.000	0.087		0.929	0.008	0.063	
	0.000	0.991	0.009	94.69	0.000	1.000	0.000	93.00	0.009	0.981	0.009	92.61	0.009	0.991	0.000	94.29	0.000	1.000	0.000	95.79
	0.133	0.017	0.850		0.017	0.000	0.983		0.033	0.017	0.950		0.050	0.000	0.950		0.033	0.000	0.967	
Acc.		95.58				93.19				93.19				94.89				96.26		
101	0.968	0.000	0.032		0.952	0.000	0.048		0.944	0.008	0.048		0.913	0.000	0.087		0.944	0.000	0.056	
	0.000	0.972	0.028	95.05	0.000	0.981	0.019	94.69	0.000	0.991	0.009	92.59	0.000	1.000	0.000	94.10	0.000	1.000	0.000	95.99
	0.067	0.017	0.917		0.067	0.017	0.917		0.133	0.033	0.833		0.033	0.033	0.933		0.033	0.017	0.950	
Acc.		95.92				95.60				93.88				94.89				96.59		
81	0.960	0.000	0.040		0.968	0.000	0.032		0.952	0.008	0.040		0.881	0.008	0.111		0.849	0.000	0.151	
	0.000	0.963	0.037	95.18	0.000	0.981	0.019	94.99	0.000	1.000	0.000	94.71	0.000	0.981	0.019	90.24	0.000	0.991	0.009	91.21
	0.033	0.017	0.950		0.067	0.033	0.900		0.083	0.033	0.883		0.100	0.033	0.867		0.017	0.050	0.933	
Acc.		95.92				95.92				95.58				91.49				91.84		
61	0.984	0.000	0.016		0.952	0.000	0.048		0.921	0.000	0.079		0.865	0.000	0.135		0.889	0.000	0.111	
	0.000	0.963	0.037	94.09	0.000	0.981	0.019	93.85	0.000	1.000	0.000	92.61	0.000	0.981	0.019	92.46	0.000	0.991	0.009	92.75
	0.100	0.033	0.867		0.117	0.000	0.883		0.017	0.117	0.867		0.033	0.000	0.967		0.017	0.050	0.933	
Acc.		95.24				94.89				93.88				92.86				93.54		

Note: The term "window size" is replaced by **Size** whose dimensions are denoted by S instead of $S \times S$ pixels; **Acc.** here denotes the accuracy.

Table 2.6. Performance of K-NN classifier for different combinations of textural features.

Size	***C* for** ρ_2			$\mathbf{F_m}$	***C* for** ρ_3			$\mathbf{F_m}$	***C* for** ρ_4			$\mathbf{F_m}$	***C* for** ρ_5			$\mathbf{F_m}$	***C* for** ρ_6			$\mathbf{F_m}$
161	0.944	0.000	0.056		0.968	0.000	0.032		0.976	0.000	0.024		0.992	0.000	0.008		0.960	0.000	0.040	
	0.000	0.944	0.056	91.05	0.000	0.954	0.046	93.83	0.000	0.972	0.028	94.57	0.009	0.981	0.009	97.23	0.000	0.972	0.028	94.71
	0.133	0.017	0.850		0.083	0.017	0.900		0.100	0.017	0.883		0.067	0.000	0.933		0.083	0.000	0.917	
Acc.		92.52				94.89				95.58				97.62				95.58		
121	0.968	0.000	0.032		0.968	0.000	0.032		0.984	0.000	0.016		0.960	0.000	0.040		0.968	0.000	0.032	
	0.000	0.972	0.028	93.35	0.000	0.972	0.028	94.21	0.000	0.944	0.056	93.79	0.009	0.981	0.009	94.36	0.000	0.963	0.037	92.95
	0.150	0.000	0.850		0.117	0.000	0.883		0.117	0.000	0.883		0.117	0.000	0.883		0.150	0.000	0.850	
Acc.		94.56				95.24				94.89				95.24				94.22		
101	0.944	0.000	0.056		0.968	0.000	0.032		0.976	0.000	0.024		0.952	0.000	0.048		0.976	0.000	0.024	
	0.000	0.963	0.037	90.94	0.000	0.963	0.037	93.78	0.000	0.963	0.037	91.18	0.009	0.963	0.028	92.31	0.000	0.972	0.028	93.75
	0.183	0.000	0.817		0.100	0.017	0.883		0.233	0.000	0.767		0.150	0.000	0.850		0.150	0.000	0.850	
Acc.		92.52				94.89				92.86				93.54				94.89		
81	0.952	0.000	0.048		0.968	0.000	0.032		0.976	0.000	0.024		0.976	0.000	0.024		0.944	0.000	0.056	
	0.000	0.954	0.046	92.22	0.000	0.981	0.019	93.30	0.000	0.972	0.028	92.85	0.000	0.954	0.046	92.07	0.000	0.963	0.037	92.72
	0.117	0.017	0.867		0.150	0.017	0.833		0.167	0.017	0.817		0.183	0.000	0.817		0.117	0.000	0.883	
Acc.		93.54				94.56				94.22				93.54				93.88		
61	0.968	0.000	0.032		0.960	0.000	0.040		0.976	0.000	0.024		0.976	0.000	0.024		0.976	0.000	0.024	
	0.000	0.935	0.065	90.90	0.000	0.954	0.046	90.41	0.000	0.972	0.028	93.71	0.000	0.972	0.028	93.75	0.009	0.981	0.009	96.42
	0.183	0.000	0.817		0.217	0.000	0.783		0.133	0.017	0.850		0.150	0.000	0.850		0.067	0.000	0.933	
Acc.		92.52				92.18				94.89				94.89				96.94		

Note: The term "window size" is replaced by **Size** whose dimensions are denoted by S instead of $S \times S$ pixels; **Acc.** here denotes the accuracy.

Table 2.7. Performance of LDA classifier for different combinations of textural features.

Size	***C* for** ρ_2			$\mathbf{F_m}$	***C* for** ρ_3			$\mathbf{F_m}$	***C* for** ρ_4			$\mathbf{F_m}$	***C* for** ρ_5			$\mathbf{F_m}$	***C* for** ρ_6			$\mathbf{F_m}$
161	0.952	0.000	0.048		0.984	0.000	0.016		0.897	0.000	0.103		0.992	0.000	0.008		0.968	0.000	0.032	
	0.000	0.944	0.056	95.04	0.000	0.963	0.037	96.73	0.000	0.972	0.028	92.89	0.019	0.981	0.000	97.79	0.009	0.972	0.019	96.49
	0.017	0.000	0.983		0.033	0.000	0.967		0.050	0.000	0.950		0.050	0.000	0.950		0.033	0.000	0.967	
Acc.		95.58				97.28				93.54				97.96				96.94		
121	0.921	0.000	0.079		0.944	0.000	0.056		0.960	0.000	0.040		0.960	0.000	0.040		0.968	0.000	0.032	
	0.000	0.963	0.037	93.51	0.000	0.963	0.037	94.96	0.000	0.972	0.028	96.46	0.009	0.991	0.000	97.33	0.000	0.991	0.009	97.17
	0.050	0.000	0.950		0.033	0.000	0.967		0.017	0.000	0.983		0.017	0.000	0.983		0.033	0.000	0.967	
Acc.		94.22				95.58				96.94				97.62				97.62		
101	0.952	0.000	0.048		0.897	0.000	0.103		0.976	0.000	0.024		0.944	0.000	0.056		0.960	0.000	0.040	
	0.000	0.954	0.046	94.05	0.000	0.963	0.037	93.03	0.000	0.963	0.037	95.49	0.009	0.963	0.028	95.93	0.000	0.991	0.009	97.23
	0.067	0.000	0.933		0.033	0.000	0.967		0.067	0.000	0.933		0.000	0.000	1.000		0.017	0.000	0.983	
Acc.		94.89				93.54				96.26				96.26				97.62		
81	0.944	0.000	0.056		0.968	0.000	0.032		0.960	0.000	0.040		0.984	0.000	0.016		0.944	0.000	0.056	
	0.000	0.944	0.056	94.71	0.000	0.981	0.019	97.21	0.000	0.981	0.019	97.28	0.000	0.954	0.046	96.78	0.000	0.981	0.019	95.69
	0.017	0.000	0.983		0.017	0.000	0.983		0.000	0.000	1.000		0.017	0.000	0.983		0.033	0.000	0.967	
Acc.		95.24				97.62				97.62				97.28				96.26		
61	0.952	0.000	0.048		0.944	0.000	0.056		0.921	0.000	0.079		0.944	0.000	0.056		0.913	0.000	0.087	
	0.000	0.926	0.074	93.89	0.000	0.954	0.046	94.61	0.000	0.981	0.019	94.66	0.000	0.981	0.019	96.14	0.009	0.981	0.009	95.30
	0.033	0.000	0.967		0.033	0.000	0.967		0.033	0.000	0.967		0.017	0.000	0.983		0.000	0.000	1.000	
Acc.		94.56				95.24				95.24				96.60				95.58		

Note: The term “window size” is replaced by **Size** whose dimensions are denoted by S instead of $S \times S$ pixels; **Acc.** here denotes the accuracy.

Table 2.8. Performance of decision tree classifier.

Size	***C* for all features**			**F_m**
161	0.976	0.000	0.024	
	0.009	0.954	0.037	97.39
	0.067	0.000	0.933	
Acc.		97.39		
121	0.976	0.008	0.016	
	0.000	0.991	0.009	91.11
	0.133	0.017	0.850	
Acc.		91.02		
101	0.968	0.000	0.032	
	0.000	0.972	0.028	96.07
	0.067	0.017	0.917	
Acc.		96.02		
81	0.960	0.000	0.040	
	0.000	0.963	0.037	93.49
	0.033	0.017	0.950	
Acc.		93.20		
61	0.984	0.000	0.016	
	0.000	0.963	0.037	92.92
	0.100	0.033	0.867	
Acc.		92.82		

Note: The term "window size" is replaced by **Size** whose dimensions are denoted by S instead of $S \times S$ pixels; **Acc.** here denotes the accuracy.

Table 2.9. Window size and subset of features.

Classifier	**Window size**	**Feature combination**
Naïve Bayes	161×161	ρ_3
K-NN	161×161	ρ_5
LDA	101×101	ρ_6
DT	161×161	All

2.6.3. Example Problem

A numerical example (microstructure) is provided in this section to demonstrate the importance of choosing most relevant textural features for the prediction of metallurgical phases. This microstructure (see Figure 2.4(a)) consists of two distinct metallurgical phases namely, ferrite and pearlite. K-NN classifier with an ideal window size of 161×161 pixels (see Table 2.9)

is chosen and classification is performed with the following textural features: (a) 'pixel intensity', (b) 'pixel intensity' and top 3 textural features, ρ_4 (see Table 2.4) and (c) 'pixel intensity' and top 4 textural features, ρ_5 (see Table 2.4). The results obtained for each of the selected set of relevant features is shown in Figure 2.4(b)-(d). From Figure 2.4 (b), it can be inferred that K-NN classifier predicts significant portion of martensite in the microstructure when only 'pixel intensity' is considered. This prediction is not true and this can be attributed to the fact that considerable portion of martensite pixel intensities overlapped with ferrite and pearlite which resulted in such a prediction (see Figure 1.5(b)). However, considering top three relevant textural features in addition to the 'pixel intensity' (i.e. ρ_4) resulted in reduction of martensite (see Figure 2.4(c)). Further addition of one more textural feature to ρ_4 resulted in elimination of martensite misclassification and grain boundaries from the microstructure (see Figure 2.4 (d)). The slight amount of martensite and grain boundaries observed in the final microstructure can be attributed to the error from the classifier. With this, it can be concluded that the addition of relevant textural features to 'pixel intensity' improves the prediction accuracy of the K-NN classifier. This can also be proved for other classifiers but is omitted to avoid repetition. In other words, 'pixel intensity' alone is inadequate for phase identification, especially when pixel intensity of different metallurgical phases overlap. Note that the further addition of features degrades the prediction accuracy of classifier as the features might be redundant or irrelevant [50]. In this study, ReleifF algorithm was employed to rank the features in the decreasing order of relevance. Hence, when more than first few relevant features are chosen, the prediction accuracies are observed to decrease as the low ranked features may be irrelevant.

For comparison purpose, the same microstructure is also processed in an image processing software ImageJ to identify the phases and segmented image as shown in Figure 2.4

(e). From Figure 2.4 (e) it is observed that two distinct phases are present in the segmented image of the microstructure. At this juncture it is important to note that this segmentation process requires the end-user to input the number of phases present in the microstructure which is not necessary for the procedure proposed in the manuscript. Based on the provided input and single threshold level, the distinct metallurgical phases are identified when ImageJ is used. Unlike the proposed method, this segmentation process fails to identify the grain boundary and categorizes it into pearlite i.e. comparatively less volume fraction of pearlite pixels are predicted by the trained classifier reducing the misclassification of grain boundaries into pearlite. This can be attributed to the similar pixel intensity exhibited by both grain boundary and pearlite.

2.6.4. Validation

In this study, three different microstructural images are randomly chosen from the test image set to validate the proposed texture based phase identification in a microstructure. These microstructural images correspond to ASTM A36-500 air cooled (500AC), ASTM A36-900 air cooled (900AC) and ASTM A36-900 water cooled (900WC) specimens which are shown in Figure 2.5 to Figure 2.7. The textural features were extracted using the procedure explained in Section 2.4 which served as the test data. Based on the combinations of textural features and ideal window sizes provided in Table 2.9, the metallurgical phases of all the three test images are identified and are shown in Figure 2.5 to Figure 2.7. From Figure 2.5 to Figure 2.7 it is observed that all the four classifiers were able to predict the metallurgical phases accurately for both air cooled (ferrite-pearlite) (Figure 2.5 and Figure 2.6) and water cooled (martensite and ferrite) microstructures (Figure 2.7). Besides the identification of metallurgical phases, the volume fractions of the phases are also evaluated for all four classifiers and are summarized in Table 2.10. The volume fraction of ferrite and pearlite predicted for (1) ASTM A36-500AC is 81.6%

and 17.7%, respectively, and (2) ASTM A36-900AC is 78.8% and 19.8%, respectively. For ASTM A36-900WC specimen the volume fractions of ferrite and martensite are predicted to be 36% and 64%, respectively.

Table 2.10. Volume fractions (%) of distinct metallurgical phases.

Classifier	Microstructure								
	500-AC			900-AC			900-WC		
	Ferrite	Pearlite	Martensite	Ferrite	Pearlite	Martensite	Ferrite	Pearlite	Martensite
NB	79.2	20.8	0	77	23	0	39	0	61
K-NN	81.2	18.2	0.6	76.9	20.6	2.5	37	0	63
LDA	81	16.8	2.2	80.2	18.8	1	33	0	67
DT	85	15	0	81.1	17.5	1.4	31	1.5	67.5

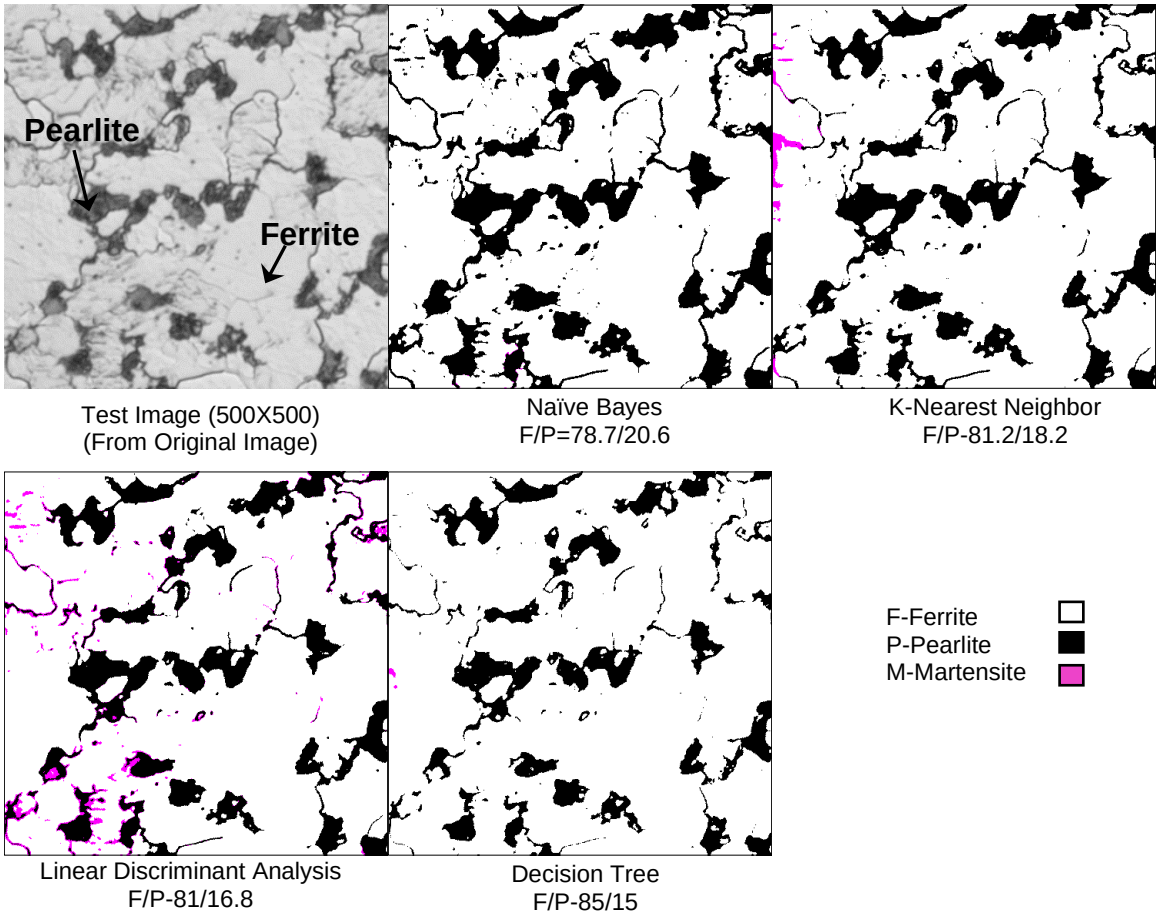


Figure 2.5. Microstructure of ASTM A36-500AC and machine learning based metallurgical phase identification. It is observed that all four classifiers predicted the phases accurately.

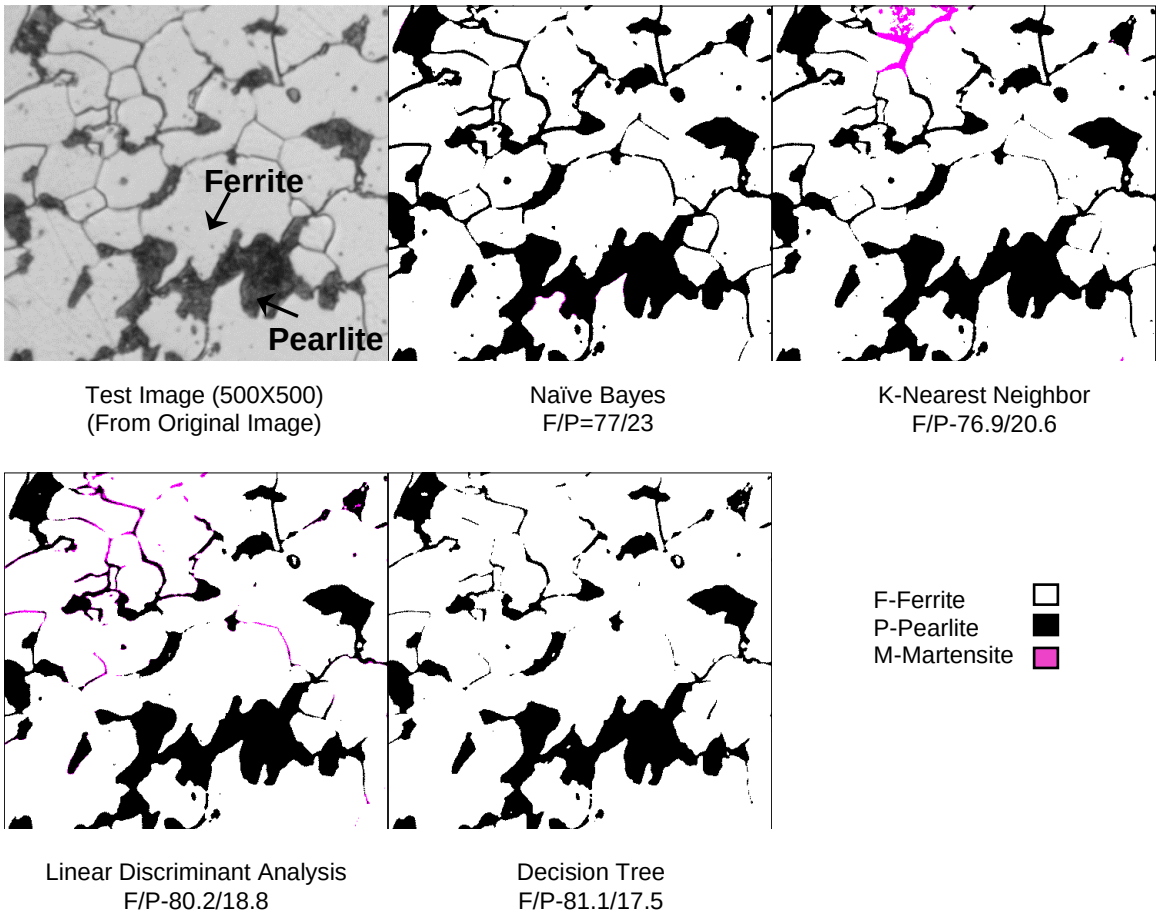


Figure 2.6. Microstructure of ASTM A36-900AC and machine learning based metallurgical phase identification. It is observed that all four classifiers predicted the phases accurately.

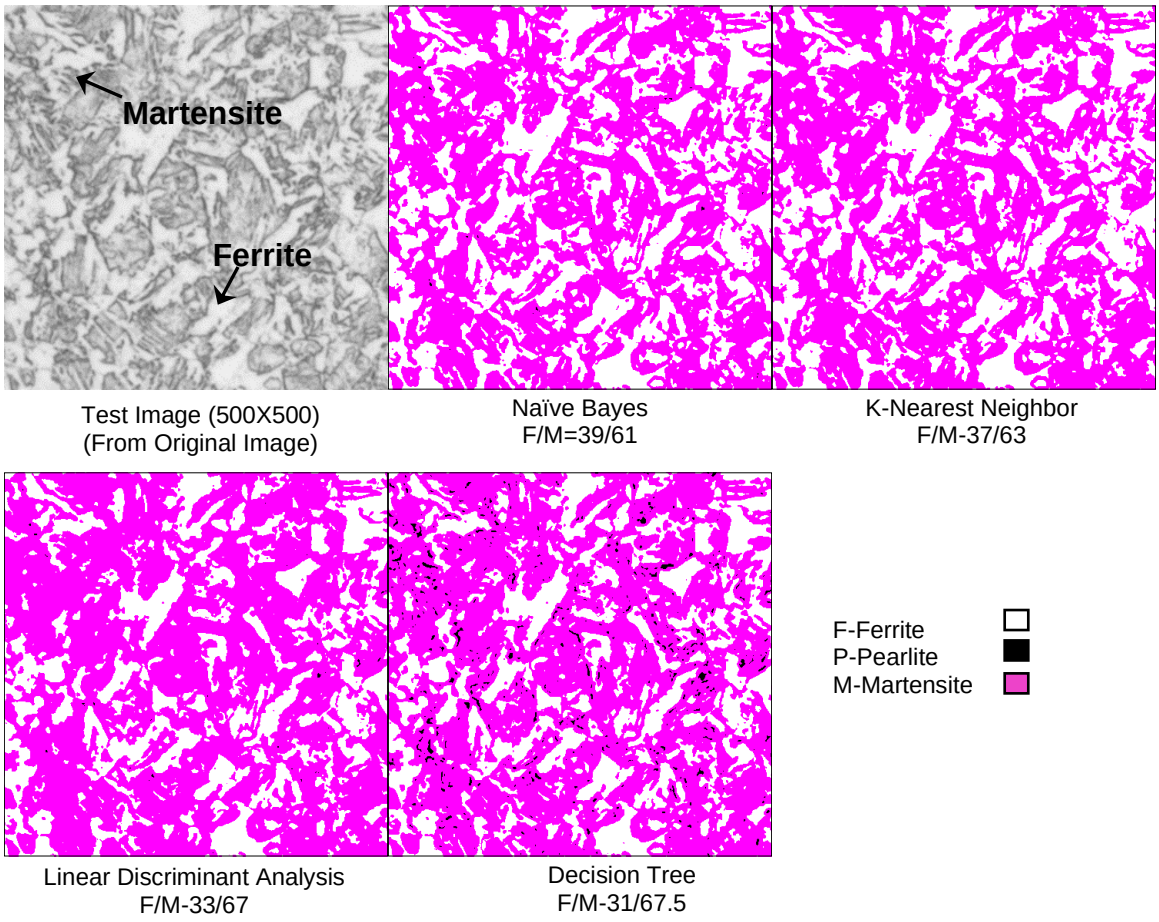


Figure 2.7. Microstructure of ASTM A36-900WC and machine learning based metallurgical phase identification. It is observed that all four classifiers predicted the phases accurately.

2.7. Summary and Recommendations

In this study, a supervised machine learning approach is proposed to identify the metallurgical phases in ASTM A36 heat treated steel, namely ferrite, pearlite and martensite. To identify or classify the metallurgical phases in the microscopic images, both the pixel intensities and textural features are extracted from the images for individual phases which are then used as the descriptive features for machine learning classifiers. To extract the textural features, GLCM of each metallurgical phase is evaluated and to perform the classification Naïve Bayes, K-NN, LDA and Decision Tree classifiers are employed. Microstructural images corresponding to nine different heat-treated metallographic specimens are acquired using optical microscope and

descriptive (textural) features are generated for all three metallurgical phases and stored in a dataset $\boldsymbol{D}$. As the ideal window size for extraction of textural features is not known a priori, window sizes of 61×61, 81×81, 101×101, 121×121 and 161×161 pixels are considered to extract the textural features which are allocated to the center pixel of each window. Feature selection is performed on dataset $\boldsymbol{D}$ using ReliefF algorithm and the most relevant features are obtained. Among the 20 descriptive features, 'pixel intensity', 'maximum probability', 'auto-correlation', 'sum of squares' 'cluster shade', 'sum variance', 'sum average' and 'energy' are found to be most relevant features for all five window sizes. The performance of all four classifiers are assessed and the ideal window size and a combination of most relevant features that minimized the classification error are determined. A numerical example is provided to demonstrate the importance of choosing the most relevant features and the validation of the proposed approach is carried out on three different microstructural images that are not the part of training data. Unlike the threshold based segmentation approach, the proposed approach avoids the misclassification of grain boundaries in to pearlite. Further, the proposed approach does not require the end-user to input the number of metallurgical phases present in the microstructure which is advantageous when investigating new microstructures.

Based on the current study, following two recommendations are provided: (1) sufficient number of data points must be acquired (under similar conditions) to train the classifier and (2) an optimal window size must be determined in conjunction with the subset of relevant features for accurate prediction of metallurgical phases.

2.8. References

1. Clemens, H., S. Mayer, and C. Scheu, Microstructure and Properties of Engineering Materials. Neutrons and Synchrotron Radiation in Engineering Materials Science: From Fundamentals to Applications, 2017.
2. Fan, Z., Microstructure and mechanical properties of multiphase materials: University of Surrey; 1993,
3. Bales, B., T. Pollock, and L. Petzold, Segmentation-free image processing and analysis of precipitate shapes in 2D and 3D. Modelling and Simulation in Materials Science and Engineering, 2017. 25(4): 045009.
4. Beddoes, J. and M. Bibby, Principles of metal manufacturing processes: Butterworth-Heinemann; 1999.
5. Hall, E., The deformation and ageing of mild steel: III discussion of results. Proceedings of the Physical Society. Section B, 1951. 64(9): 747.
6. Petch, N., The influence of grain boundary carbide and grain size on the cleavage strength and impact transition temperature of steel. Acta metallurgica, 1986. 34(7): 1387-1393.
7. Materials, A.S.f.T.a. ASTM E112-96 (2004) e2: Standard Test Methods for Determining Average Grain Size. 2004. ASTM.
8. Standard, A., Standard Test Method for Determining Volume Fraction by Systematic Manual Point Count. ASTM E562-08, 2008.
9. Campbell, A., et al., New methods for automatic quantification of microstructural features using digital image processing. Materials & Design, 2018. 141: 395-406.

10. Ayech, M.B.H. and H. Amiri. Texture description using statistical feature extraction. in Advanced Technologies for Signal and Image Processing (ATSIP), 2016.
11. Acharya, T. and A.K. Ray, Image processing: principles and applications: John Wiley & Sons; 2005.
12. Pakhira Malay, K., Digital image processing and pattern recognition: PHI Learning Private Limited; 2011.
13. Sobel, I., Camera models and machine perception. Computer Science Department, Technion; 1972.
14. Dougherty, G., Digital image processing for medical applications: Cambridge University Press; 2009.
15. Jahne, B., Digital image processing: Concepts, algorithms, and scientific aplications. Berlin, Heildelberg: Springer-Verlag, 1997. 570.
16. Otsu, N., A threshold selection method from gray-level histograms. IEEE transactions on systems, man, and cybernetics, 1979. 9(1): 62-66.
17. Jain, A.K., Fundamentals of digital image processing: Englewood Cliffs, NJ: Prentice Hall;1989.
18. Jiang, X., R. Zhang, and S. Nie, Image segmentation based on level set method. Physics Procedia, 2012. 33: 840-845.
19. Sosa, J.M., et al., Development and application of MIPAR™: a novel software package for two-and three-dimensional microstructural characterization. Integrating Materials and Manufacturing Innovation, 2014. 3(1): 10.
20. Bonnet, N., Some trends in microscope image processing. Micron, 2004. 35(8): 635-653.

21. Yang, D. and Z. Liu, Quantification of microstructural features and prediction of mechanical properties of a dual-phase Ti-6Al-4V alloy. Materials, 2016. 9(8): 628.

22. Buriková, K. and G. Rosenberg, Quantification of microstructural parameter ferritic-martensite dual phase steel by image analysis. Metal, 2009. 5: 19-21.

23. Ontman, A.Y. and G.J. Shiflet, Microstructure segmentation using active contours—Possibilities and limitations. JOM, 2011. 63(7): 44-48.

24. Coverdale, G., et al., Cluster analysis of the microstructure of colloidal dispersions using the maximum entropy technique. Journal of magnetism and magnetic materials, 1998. 188(1-2): 41-51.

25. Azimi, S.M., et al., Advanced Steel Microstructural Classification by Deep Learning Methods. Scientific reports, 2018. 8(1): 2128.

26. Prakash, P., V. Mytri, and P. Hiremath, Fuzzy Rule Based Classification and Quantification of Graphite Inclusions from Microstructure Images of Cast Iron. Microscopy and Microanalysis, 2011. 17(6): 896-902.

27. DeCost, B.L. and E.A. Holm, A computer vision approach for automated analysis and classification of microstructural image data. Computational materials science, 2015. 110: 126-133.

28. Haralick, R.M., Statistical and structural approaches to texture. Proceedings of the IEEE, 1979. 67(5): 786-804.

29. Haralick, R.M. and K. Shanmugam, Textural features for image classification. IEEE Transactions on systems, man, and cybernetics, 1973(6): 610-621.

30. Soh, L.-K. and C. Tsatsoulis, Texture analysis of SAR sea ice imagery using gray level co-occurrence matrices. CSE Journal Articles, 1999: 47.

31. Gómez, W., W. Pereira, and A.F.C. Infantosi, Analysis of co-occurrence texture statistics as a function of gray-level quantization for classifying breast ultrasound. IEEE transactions on medical imaging, 2012. 31(10): 1889-1899.
32. Camastra, F. and A. Vinciarelli, Machine learning for audio, image and video analysis: theory and applications: Springer; 2015.
33. Sessions, V. and M. Valtorta, The Effects of Data Quality on Machine Learning Algorithms. ICIQ, 2006. 6: 485-498.
34. James, G., et al., An introduction to statistical learning: Springer; 2013.
35. Kelleher, J.D., B. Mac Namee, and A. D'Arcy, Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies: MIT Press; 2015.
36. Aggarwal, C.C., Data classification: algorithms and applications: CRC Press; 2014.
37. Naik, D.L. and R. Kiran, Naïve Bayes classifier, multivariate linear regression and experimental testing for classification and characterization of wheat straw based on mechanical properties. Industrial Crops and Products, 2018. 112: 434-448.
38. Kononenko, I. and M. Kukar, Machine learning and data mining: Horwood Publishing; 2007.
39. Mitchell, T.M., Machine Learning: McGraw-Hill, Inc; 1997.
40. Quinlan, J.R., C4. 5: programs for machine learning: Elsevier; 2014.
41. Bramer, M., Principles of data mining: Springer; 2007.
42. Sajid, H.U. and R. Kiran, Influence of stress concentration and cooling methods on post-fire mechanical behavior of ASTM A36 steels. Construction and Building Materials, 2018. 186: 920-945.

43. Kira, K. and L.A. Rendell. The feature selection problem: Traditional methods and a new algorithm. in Aaai. 1992.

44. Urbanowicz, R.J., et al., Relief-based feature selection: introduction and review. Journal of biomedical informatics, 2018.

45. Robnik-Šikonja, M. and I. Kononenko, Theoretical and empirical analysis of ReliefF and RReliefF. Machine learning, 2003. 53(1-2): 23-69.

46. Müller, M.E., Relational Knowledge Discovery: Cambridge University Press; 2012.

47. Akosa, J. Predictive Accuracy: A Misleading Performance Measure for Highly Imbalanced Data. in Proceedings of the SAS Global Forum. 2017.

48. Sokolova, M. and G. Lapalme, A systematic analysis of performance measures for classification tasks. Information Processing & Management, 2009. 45(4): 427-437.

49. Ratanamahatana, C.A. and D. Gunopulos, Scaling up the naive Bayesian classifier: Using decision trees for feature selection. 2002.

50. Kohavi, R. and G.H. John, Wrappers for feature subset selection. Artificial intelligence, 1997. 97(1-2): 273-324.

3. IDENTIFICATION OF FRACTURE IN METALS USING LBP TEXTURAL FEATURES[2]

3.1. Introduction

Fracture in metals is one of the most important reasons behind the failure of engineering components and structures. Fracture in metals has led to catastrophic failures in steel buildings [1] and bridges [2], oil, and gas pipelines [3], automobiles [4] and aerospace structures [5]. Ductile fracture, brittle (cleavage/ transgranular) fracture and intergranular fracture are the most common types of fracture in metals under monotonic loading conditions [6]. A decision about the choice of a suitable damage model to simulate the failure depends on the type of fracture as the basic microscopic damage mechanisms leading to the fracture varies from one fracture type to another. Both ductile and brittle fracture zones are observed in structural steels [7-10], dual- and multiphase steels [11-13], aluminum [14, 15], and titanium [16]. The choice of damage model in the case where multiple fracture mechanisms are involved depends on 1) fracture initiation mechanism, 2) the dominant fracture mechanism and/ or 3) the conditions under which the fracture transitions from one type to another type. Although the fracture type can be determined based on the mechanical and microstructural properties of a metal and state of stress and strain, a visual inspection of fractographic images is often conducted to identify the fracture type of metal. Although simple, visual inspection is slow, prone to confirmation bias and cannot be used for quantitative analysis of fracture surfaces. For instance, the evaluation of dominant

[2] This chapter is based on the paper "Identification and characterization of fracture in metals using machine learning based texture recognition algorithms." Engineering Fracture Mechanics 219 (2019): 106618. https://doi.org/10.1016/j.engfracmech.2019.106618. The material in this chapter was co-authored by Dayakar Naik Lavadiya (DNL) and Ravi Kiran Yellavajjala (RK) .Contributions of authors are as follows: Conceptualization, D.L.N. and R.K.; Formal analysis, D.L.N.; Funding acquisition, R.K.; Investigation, R.K.; Methodology, D.L.N., and R.K.; Project administration, R.K.; Resources, R.K.; Software, D.L.N.; Supervision, R.K.; Validation, D.L.N.; Writing—original draft, D.L.N.; Writing—review & editing, D.L.N. and R.K..

fracture type and description of the visual characteristics of fracture surfaces is not possible through visual inspection. In this study, digital image processing using an illumination and rotation invariant texture measure in conjunction with a supervised machine learning algorithm is used for the quantitative analysis of fracture surfaces.

With the advent of new technology and an increase in computational resources, digital image processing is frequently being used in various disciplines. Specifically, it is used for automatic identification of the objects in an image to quantify aspects such as shape, dimensions, the spatial position of the target objects, etc. [17]. Digital image processing involves the use of algorithms or mathematical operations on the digital images to enhance the visual quality of the image or to extract useful visual information from the image [18]. When the extracted information corresponds to a specific object or a region in an image, the extracted information is referred to as features of the object, and the process of extracting such information is referred to as feature extraction. Features of the objects are also called as the descriptors which are used for the segmentation or classification of an image, i.e. identification of an object in a given image and categorization of the object into one of the known finite classes. Examples of some features that are used in identifying the objects include pixel intensity, shape, edges, color, texture, etc. Among these features, pixel intensity is more commonly used to perform the task of image segmentation. Pixel intensity is a measure of magnitude of the grayscale level of a pixel in a digital image. In the study performed by Kosarevych et al. [19], pixel intensity based multilevel thresholding technique was employed to perform the automatic segmentation of fractographic images of steel. A subset of histogram bins of pixel intensities corresponding to specific element or heterogeneity was extracted and was used as features to identify the elements of interest in the fractographic image.

The texture is another commonly used feature that possesses information about an object in an image. Unlike pixel intensity, which is associated with an individual pixel, the texture is associated with a region (a group of pixels) of the image. It aids in identifying the objects in a digital image that possess unique textures. An object is considered to possess a texture if its appearance is composed of repetitive visual patterns defined by variations in brightness and/ or color [20]. Examples of texture possessing objects include wood, grass, soil, concrete, etc. In the study conducted by Rodriguez et al. [21], textural features are employed to identify the type of failure from the fractographic images of metallic materials. Three types of failure were considered in their study, namely brittle, ductile, and fatigue. To extract the textural features of each failure type, gray level co-occurrence matrix (GLCM), fractal analysis, and texture energy laws were employed in their study. Similar studies were conducted by Dutta et al. [22] in which the fractographic images of AISI stainless steel were considered, and automatic characterization of the fracture surfaces was carried out. However, in their study, gray level run length matrix (GLRLM) was employed in addition to GLCM and fractal analysis to extract textural features. GLRLM was reported to be the most suitable method for predicting the failure type among the considered two methods. Textural feature extraction methods that are mentioned above are illumination/ grayscale and rotation variant, i.e., the extracted textural features may change with the change in the illumination and rotation of an image [23]. In reality, the fracture surfaces are quite uneven at microscales. Hence, the fractographic images obtained from scanning electron microscopy (SEM) consists of varying illumination levels across the image. Therefore, adopting an illumination/ grayscale and rotation invariant method to extract textural features may result in more accurate prediction when compared to other methods.

Local binary patterns (LBP) is a texture quantification algorithm which was proposed by Ojala et al. [24] and is a grayscale and rotation invariant method. Keeping in view the variations in the grayscale levels of the regions in the fractographic images, LBP textural feature extraction algorithm is implemented in this chapter to extract the textural features. These textural features are then used in conjunction with a supervised machine learning classifier to perform automatic identification of the fracture type in structural steel. The main contribution of this study is to propose a robust methodology that can fully automate the identification of the fracture type in metals by performing quantitative analysis of fractographs. The rest of the chapter is organized as follows: a brief overview of LBP is provided in Section 3.2, description of supervised machine learning and LDA algorithm is provided in Section 3.3, the methodology is described in Section 3.4, results are provided in Section 3.5, and summary of the study is provided in Section 3.6.

3.2. Local Binary Pattern (LBP) as Texture Descriptors

Consider a digital image of an object that is said to possess a certain texture. Let the domain of this image be denoted by $\boldsymbol{\Omega} \in \mathbb{Z}_{+}^{N_y \times N_x}$, where N_x and N_y represent the number of pixels in the x and y directions, respectively and $\mathbb{Z}_{+}$ is a set of positive integers (see Figure 3.1) [20]. Let $\boldsymbol{\omega} \in \mathbb{Z}_{+}^{n \times n}$ represent a subdomain of the domain $\boldsymbol{\Omega}$ (also called as local image) whose size is $n \times n$ pixels, where $n \geq 3$ is an odd number (see Figure 3.1). Note that $\boldsymbol{\omega}$ is obtained by partitioning $\boldsymbol{\Omega}$ into overlapping blocks of size $n \times n$ pixels such that center pixel of each $\boldsymbol{\omega}$ represents each pixel of the image $\boldsymbol{\Omega}$. Let R be the radius of the circle drawn on the local image $\boldsymbol{\omega}$ such that the center of the circle coincides with the center pixel of the local image $\boldsymbol{\omega}$ and let P be the number of pixels (also referred to as neighboring pixels) that lie on the perimeter of this circle (see Figure 3.1). If $g_0, g_1, \ldots, g_{P-1}$ represents the grayscale value of $1,2, \ldots, P$ neighboring

pixels that lie on the perimeter of the circle, respectively and, 0's/ 1's represent their respective encoded binary values, then the pattern of P-digit binary numbers (example, 11000110 for P=8) obtained in a specific sequence around the center pixel is referred to as the Local Binary Pattern (LBP) of $\boldsymbol{\omega}$. Attributed to the fact that the interpretation of a P-digit binary number becomes difficult when P assumes a large value, an integer I is introduced in the place of a P-digit binary number. For instance if a 8-digit binary pattern is considered, then 2^8 combinations of binary values are possible and each combination corresponds to a specific integer I. In practice, LBP is replaced by an integer I ($1 \leq I \leq 2^P$), which is referred to as LBP value and is unique for a specific P-digit binary pattern. To evaluate I, each binary digit of LBP or P-digit pattern is multiplied with a fixed weight $w_i = 2^{i-1}$ (see Figure 3.1) and the resulting values are added. Here i denotes the pixel position in the P-digit pattern and takes the values from 1 to P. To quantify the texture of an image $\boldsymbol{\Omega}$, the LBP values of all local images are first evaluated. These LBP values are then summarized in the form of a histogram, which is used to represent the texture of an image $\boldsymbol{\Omega}$. In what follows, a detailed derivation of LBP is provided to demonstrate its rotation and grayscale invariance [24].

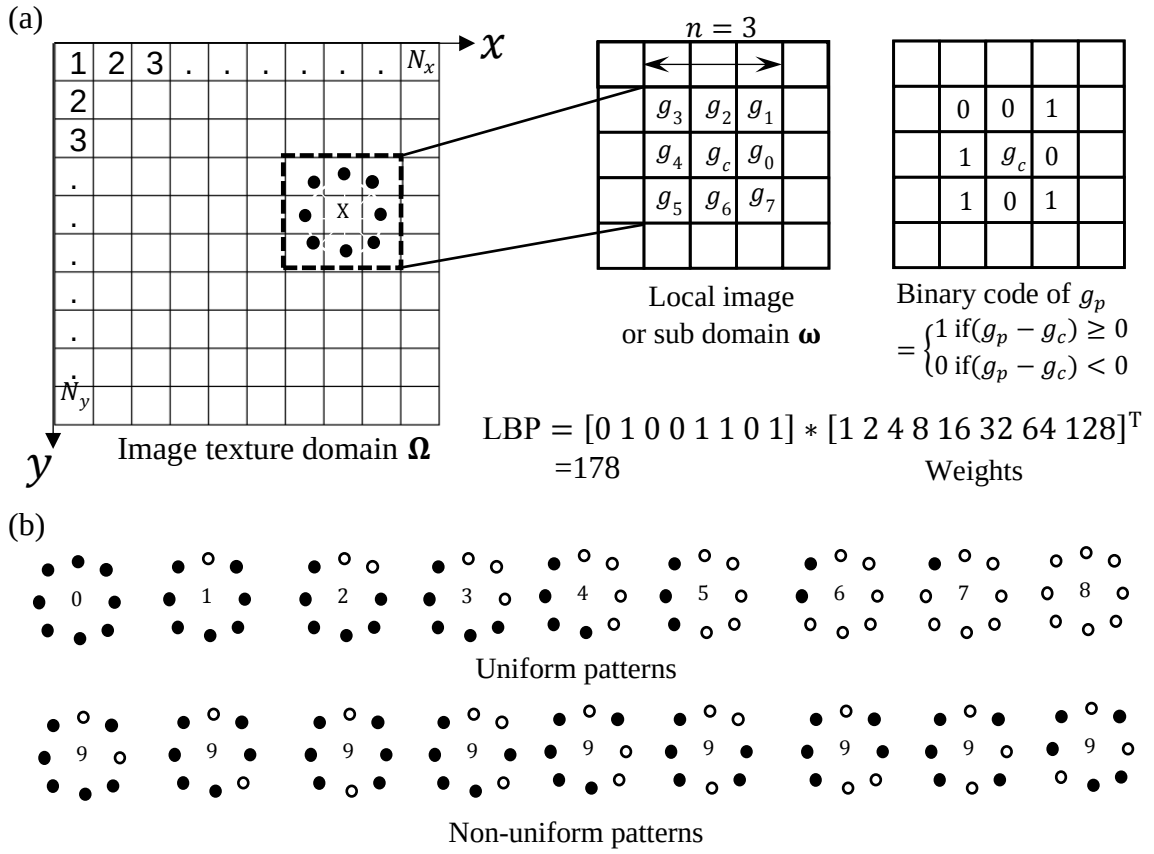


Figure 3.1. Illustration of local binary pattern (LBP) estimation and uniformity measures: (a) a schematic of local image **ω** with radius R=1 and neighboring pixels P=8 is shown and the order in which binary pattern is evaluated is also provided i.e. same as the order of $g_0, g_1, \ldots, g_7$ and, (b) rotation invariant uniform/non-uniform patterns with #0 representing bright spot, #4 representing edge and #8 representing dark spot/flat areas. (Note: black circles corresponds to 0's and white circles corresponds to 1's. Other non-uniform patterns can be found elsewhere [24]).

Let T represent the texture of the image **Ω** and g_c denote the grayscale value of the center pixel of any local image $\boldsymbol{\omega}$. T is then defined as the joint distribution of grayscale values $g_{p=0,1,\ldots,P-1}$ of P neighboring pixels around the center pixel g_c, which is expressed as

$$T = t(g_c, g_0, g_1, \ldots, g_{P-1}) \tag{3.1}$$

For achieving grayscale invariant texture measure, first g_c is subtracted from the grayscale values of its neighborhood pixels g_p and the following equation is obtained

$$T = t(g_c, (g_0 - g_c), (g_1 - g_c), \ldots, (g_{P-1} - g_c)) \tag{3.2}$$

Assuming that the difference $(g_p - g_c)$ is independent of the grayscale value of the center pixel g_c, Eq. (3.2) can be factorized as follows

$$T \approx t(g_c)t((g_0 - g_c), (g_1 - g_c), \dots, (g_{P-1} - g_c)) \tag{3.3}$$

Note that the exact independence is not guaranteed and factorized distribution in Eq. (3.3) is only an approximation [24]. With this, the texture of the image $\mathbf{\Omega}$ can be quantified by employing the joint difference distribution of grayscale levels and ignoring the distribution of grayscales of the central pixels which is expressed as follows

$$T \approx t((g_0 - g_c), (g_1 - g_c), \dots, (g_{P-1} - g_c)) \tag{3.4}$$

Attributed to the fact that the signed difference $(g_p - g_c)$ is not affected by the change in illuminance of the image, the above joint difference distribution is said to be invariant against grayscale level shifts. Further, to achieve the invariance with respect to the scaling or quantization of grayscale value, only signs of difference $(g_p - g_c)$ is considered instead of their exact values. Mathematically this is expressed as

$$T \approx t(s(g_0 - g_c), s(g_1 - g_c), \dots, s(g_{P-1} - g_c)) \tag{3.5}$$

where, $s(g_p - g_c)$ represents the sign of the difference $(g_p - g_c)$ and takes the value of 1 if difference of $(g_p - g_c)$ is ≥ 0 and takes the value of 0 if difference $(g_p - g_c)$ is < 0. This operation $s(g_p - g_c)$ encodes the grayscale values of P-neighboring pixels into binary values 0/ 1.

Assigning a factor 2^p as the weight to $s(g_p - g_c)$, Eq. (3.5) is transformed into unique LBP value that characterizes local image texture for the chosen P number of neighboring pixels

$$LBP = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \tag{3.6}$$

Note that the LBP value in Eq. (3.6) is the same as integer I that is mentioned before. At this juncture, it is important to note that the P neighboring pixels found on the perimeter of the circle around g_c may not fall exactly on the center of their respective pixels. In such cases the interpolated value of g_p is used. Details on performing interpolation can be found in the reference [24].

The LBP derived in Eq. (3.6) is not rotation invariant, i.e. the values of LBP change when the images are rotated. In other words, when the image is rotated, the position of grayscale values changes with rotation, however the weight w_i associated with each position remains the same. Therefore, to make this operator rotationally invariant, a uniformity measure U was introduced by Ojala et al [24]. This uniformity measure quantifies the number of spatial transitions of binary numbers observed in a P-digit binary pattern. These spatial transitions are the fundamental properties of a texture (see Figure 3.1(b)) and are invariant to rotation. The spatial transitions may quantify visual patterns such as bright spots, dark spots, edges of varying curvature, etc. For instance, if grayscale value of the center pixel is less than all its neighboring pixels, then $s(g_p - g_c)$ yields a value of one for all its neighboring pixels which is represented by a dark spot and/ or flat areas (no change in grayscale levels) [24]. A P-digit binary pattern is said to be uniform if it has at most two 0 to 1 or 1 to 0 transitions, and is said to be non-uniform if it has more than two transitions. For instance, 00000000, 00000001 and 00001000 are referred as uniform patterns as they have zero and two transitions (0-1 and 1-0) of binary numbers, respectively, and 01010010 is referred as the non-uniform pattern as it has six transitions. Based on the uniformity measure U, the LBP operator provided in Eq. (3.6) is modified as follows

$$LBP^{riu} = \begin{cases} \sum_{p=0}^{P-1} s(g_p - g_c) & \text{if } U(LBP) \leq 2 \\ P+1 & \text{otherwise} \end{cases} \quad (3.7)$$

where, the superscript riu denotes the rotational invariant uniform measure and

$$U(LBP) = |s(g_{P-1} - g_c) - s(g_0 - g_c)| + \sum_{p=1}^{P-1} |s(g_P - g_c) - s(g_{p-1} - g_c)|$$

In the above equation, while LBP^{riu} takes values from 0 to 8 for uniform patterns, it assumes the value of 9 for all non-uniform patterns (see Figure 3.1(b)).

In this study, the image texture of two different types of metal fractures is considered namely, brittle and ductile fractures (see Figure 3.2). From Figure 3.2, it can be deduced that the image textures of both brittle and ductile fracture are distinct and unique. If $\boldsymbol{\Omega}$ denotes the domain of an image with an unique texture, i.e. brittle/ductile fracture, then the histogram of LBP^{riu} values obtained from all the local images represents the texture of $\boldsymbol{\Omega}$. Unlike the LBP operator (see Eq. (3.6)) which results in the values ranging from 0 to 2^p, LBP^{riu} results in the values ranging from 0 to 9 (see Eq. (3.7)). Therefore, the histogram generated for brittle/ductile fracture consists of only ten bins. Each bin represents the uniformity measure U and are considered as the textural features or texture descriptors of the brittle and ductile fracture. More detailed discussion on the uniformity measures of brittle and ductile type fracture is provided in the results Section 3.5. To identify if the given image texture corresponds to the brittle or ductile type of texture based on the uniformity measure, a supervised machine learning algorithm is employed in this study. A brief description of supervised machine learning and the corresponding algorithm used for classification is provided next.

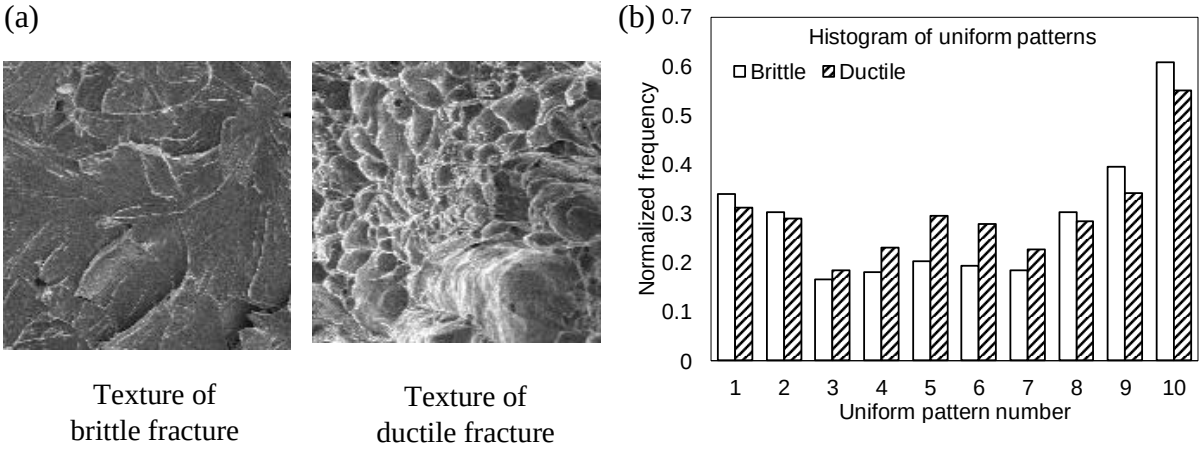


Figure 3.2. (a) Image texture of brittle and ductile fracture observed in fractographs of ASTM A992 and, (b) the histogram of uniform/non-uniform patterns for brittle and ductile fracture, where each bin is a textural feature.

3.3. Supervised Machine Learning Based Classification

Supervised machine learning is a branch of machine learning that is used to perform classification or regression by learning from a labeled dataset. A dataset is referred to as a labeled dataset when information about the outcome of an experiment, and the factors that govern the outcome is known. In machine learning terminology, the factors that govern the outcome are referred to as descriptive features, and the outcomes of an experiment/ observation are referred to as target variables. In the task of classification, the target variables are also referred to as class labels. Supervised machine learning is executed in three steps: (1) training the machine learning algorithm on the available (labeled) dataset, (2) testing the efficacy of the trained algorithm, and (3) deployment of the trained algorithm for the intended purpose. In the first step, the available dataset is partitioned into two datasets namely the training dataset and the test dataset, such that the training dataset consists of 80% of data points and the test dataset consists of the other 20% of the data points. Followed by the partitioning of available data, an algorithm is employed to learn the patterns, relationships, and/ or dependencies from the training dataset, and this step is referred to as the training step. Next, the efficacy of the trained algorithm

is verified on the test dataset. In other words, the class labels of the test dataset are predicted by the trained algorithms and then cross-validated with the available original labels. If the prediction accuracy is satisfactory, then the trained algorithm is deployed for performing classification task on new data in the third and final step.

The following mathematical notation is employed in the ensuing discussion about supervised machine learning algorithms [20]. Let $\boldsymbol{D} \in \mathbb{R}^{N \times r}$ represent the master dataset consisting of N number of row vectors and $r = q + 1$ number of column vectors. While q represents the number of descriptive features and r^{th} column represents the vector of class labels, N represents the number of experimental observations. Further, if m denotes the number of distinct class labels in the r^{th} column vector, then N_i represents the number of observations with class label i such that $\sum_{i=1}^{m} N_i = N$. Let each row of the N observations, also called as an instance, be denoted by a vector $\boldsymbol{x}_j = (x_{j1}, x_{j2}, \ldots, x_{jq}, x_{jr})$, where $j = 1$ to N and x_{jr} is the class label of j^{th} observation. Let $\boldsymbol{x}_j^*$ represent the j^{th} row vector of test dataset whose feature values $x_{j1}^*, x_{j2}^*, \ldots, x_{jq}^*$ are known but class label x_{jr}^* is unknown. Here it is important to note that the range of j in test dataset is different from the one used in the master dataset. With this, the implication of above notations in the context of current study is provided as follows: $q = 10$ refers to the number of uniform/non-uniform patterns extracted using local binary pattern ($LBP_{P,R}^{riu}$), m=2 refers to the number of fracture classes (brittle-1 and ductile-2), $N = 100$ refers to the number of brittle and ductile fracture images used for generating the master dataset $\boldsymbol{D}$, $\boldsymbol{x}_j$ refers to the vector of uniform patterns (along with class label) corresponding to j^{th} image, and $\boldsymbol{x}_j^*$ corresponds to the vector of uniform/ non-uniform patterns of test image whose fracture type has to be classified.

3.3.1. Linear Discriminant Analysis (LDA) Classifier

In this study, the linear discriminant analysis (LDA) classifier is employed to perform the task of classification. LDA is a linear classifier that employs a discriminant score function $L_j(\boldsymbol{x}^*)$ to identify or predict the class label of an instance $\boldsymbol{x}^*$ [25]. Here the subscript j indicates the label of the class C_j and the quantity L_j indicates the discriminant score corresponding to class C_j . This implies that the m number of class labels result in m number of discriminant scores designated as $L_1, L_2, \dots, L_m$. The class label that yields a maximum discriminant score is then assigned as the class label for the instance $\boldsymbol{x}^*$. In what follows, this section provides a detailed derivation of the discriminant score function L_j.

According to Bayes' theorem, if A and B are two events, then the likelihood of the event A occurring given that event B has already occurred is expressed as [26]

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \tag{3.8}$$

where, $P(A|B)$ and $P(B|A)$ are conditional or posterior probabilities, $P(A)$ and $P(B)$ are prior probabilities. For any given i^{th} observation in the training dataset, $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \dots, x_{iq}, x_{ir})$, the Eq. (3.8) can be rewritten as

$$P\left(x_{ir} = C_j | x_{i1}, x_{i2}, \dots, x_{iq}\right) = \frac{P\left(x_{i1}, x_{i2}, \dots, x_{iq} \middle| x_{ir} = C_j\right) \times P(x_{ir} = C_j)}{P(x_{i1}, x_{i2}, \dots, x_{iq})} \tag{3.9}$$

where, $C_{j=1,2}$ denotes the class label of the given textural features i.e. C_1 and C_2 are brittle fracture and ductile fracture, respectively.

Ignoring the denominator in the Eq. (3.9), as it is independent of the class label, we get a measure of conditional probability which can be written as

$$M\left(x_{ir} = C_j | x_{i1}, x_{i2}, \dots, x_{iq}\right) = P\left(x_{i1}, x_{i2}, \dots, x_{iq} \middle| x_{ir} = C_j\right) \times P(x_{ir} = C_j) \tag{3.10}$$

In fact, the class that maximizes the value of M for a given $\boldsymbol{x}$ is assigned as the class of $\boldsymbol{x}$ and the above equation is referred to as Bayes' classifier. However, evaluating the conditional probability in the right hand side of Eq. (3.10) is challenging. Therefore, in linear discriminant analysis, all the instances that belong to a class C_j are assumed to be sampled from a population with multivariate normal distribution $\mathcal{N}(\boldsymbol{\Sigma}, \boldsymbol{\mu}_j)$, where, $\boldsymbol{\Sigma} \in \mathbb{R}^{q \times q}$ and $\boldsymbol{\mu}_j \in \mathbb{R}^{1 \times q}$ are the population covariance matrix and population mean vector of all the features in the instances that belong to class C_j. This simplifies the evaluation of conditional probabilities shown in Eq. (3.10). Here it is important to note that the LDA classifier assumes the covariance matrix of all observations that belong to different class labels to be equal i.e. $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \cdots = \boldsymbol{\Sigma}_m = \boldsymbol{\Sigma}$.

For a multivariate normal distribution, the probability density function is given as

$$f(\boldsymbol{x}|C_j) = \frac{1}{\sqrt{(2\pi)^q |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_j)' \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_j)\right)$$

By substituting $f(\boldsymbol{x}|C_j)$ in the place of $P(\boldsymbol{x}|C_j)$ as a proxy measure, we get

$$\begin{aligned} &M(x_{ir} = C_j | x_{i1}, x_{i2}, \dots, x_{iq}) \\ &\quad = \frac{1}{\sqrt{(2\pi)^q |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_j)' \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_j)\right) \times P(x_{ir} = C_j) \end{aligned} \tag{3.11}$$

Note that $P(\boldsymbol{x}|C_j)$ can be replaced by its proxy value $f(\boldsymbol{x}|C_j)$ as we are interested in discrimination of instances in to classes and are not interested in evaluating the actual probabilities. By applying logarithm on both sides we get the discriminant score function for class C_j as

$$L_j(\boldsymbol{x}) = -\frac{1}{2}\log((2\pi)^q |\boldsymbol{\Sigma}|) - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_j)' \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_j) + \log P(C_j) \tag{3.12}$$

Noting that $\boldsymbol{\Sigma}^{-1}$ is symmetric, i.e., $\boldsymbol{x}' \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_j = \boldsymbol{\mu}_j' \boldsymbol{\Sigma}^{-1} \boldsymbol{x}$, we can simplify the discriminant score function as follows

$$L_j(\boldsymbol{x}) = -\frac{1}{2}\log((2\pi)^q|\boldsymbol{\Sigma}|) - \frac{1}{2}\boldsymbol{\mu}_j'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_j - \frac{1}{2}\boldsymbol{x}'\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \boldsymbol{\mu}_j'\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \log P(C_j)$$

By ignoring the terms that are independent of the class (as they do not improve the discriminative power of the algorithm), we obtain the discriminant score function for the j^{th} class as

$$L_j(\boldsymbol{x}) = -\frac{1}{2}\boldsymbol{\mu}_j'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_j + \boldsymbol{\mu}_j'\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \log P(C_j) \tag{3.13}$$

To better capture the variances in the available dataset, a pooled covariance matrix defined as

$$\boldsymbol{\Sigma}_{pl} = \frac{1}{\mathrm{N} - m}\sum_{i=1}^{m}(\mathrm{N}_i - 1)\boldsymbol{\Sigma}_i$$

is used in the place of $\boldsymbol{\Sigma}$ modifying the discriminant score function as follows

$$L_j(\boldsymbol{x}) = -\frac{1}{2}\boldsymbol{\mu}_j'\boldsymbol{\Sigma}_{pl}^{-1}\boldsymbol{\mu}_j + \boldsymbol{\mu}_j'\boldsymbol{\Sigma}_{pl}^{-1}\boldsymbol{x} + \log P(C_j) \tag{3.14}$$

In reality, the population covariance matrix ($\boldsymbol{\Sigma}$) and population mean vector ($\boldsymbol{\mu}_j$) are not known [27]. Therefore, the sample covariance matrix and sample mean vector are evaluated from the experimental observations that belong to each class and are used instead, as shown in Eq. (3.14). To predict the class label x_{jr}^* of a given test instance $\boldsymbol{x}^*$, the discriminant scores $L_{j=1:m}(\boldsymbol{x}^*)$ are evaluated for all the m class labels and the index of the class label j that yields maximum L_j value is then assigned as the class label for the test instance $\boldsymbol{x}^*$. In the context of this study, the class label$s$ are $\{C_1, C_2\}$, where C_1 and C_2 are brittle and ductile fractures, respectively.

3.4. Methodology

The methodology adopted in the current study to identify the type of fracture in fractographic images of structural steels essentially consists of following four tasks: (1)

acquisition of brittle and ductile fracture images, (2) extraction of the histogram of uniform/non-uniform patterns from brittle and ductile fracture images (building master dataset $\boldsymbol{D}$), (3) training of LDA classifier and, (4) deployment of the trained LDA classifier to identify the type of fracture in test images. Details of each of these tasks are described below.

To acquire a set of brittle and ductile fracture images, scanning electron microscope (SEM) images of fractured ASTM A992 structural steel surfaces (fractographs) are obtained in the first task. An image processing software ImageJ is then employed, and the regions of brittle and ductile regions are manually cropped from the SEM images. In total, one hundred cropped images (81×81 pixels) are obtained in this exercise, among which 50 images belong to brittle fracture, and the other 50 images belong to ductile fracture. Details of the SEM images that are used in the current study can be found elsewhere [28-31]. To extract the histogram of uniform/non-uniform patterns from all the brittle and ductile fracture images (master dataset $\boldsymbol{D}$), an in-built MATLAB® function 'extractLBPFeatures' is employed in the second task. In this study, a radius of R=1 and number of neighboring pixels P=8 (or sub-domain size, $n = 3$) is considered for extraction of the linear binary patterns i.e., uniformity measures. The extracted linear binary patterns are then used to train the LDA classifier in the third task. For this purpose, an in-house code is written in MATLAB®. In the fourth and final task, the trained LDA classifier is deployed to identify the brittle and ductile fracture regions in the test images that are not used for training purposes (see Figure 3.3). Test images are the fractographic images of fractured steel surface that were not considered for training the classifier. At this juncture, it is important to note that, unlike the training images, the test images have interspersed brittle and ductile fracture regions (see Figure 3.3). To perform the classification of the regions in the test images as ductile or brittle fracture, two approaches are employed in this study: (1) computationally inexpensive

block-wise approach, and (2) computationally intensive but accurate pixel-wise approach. Steps involved in both these approaches is described next.

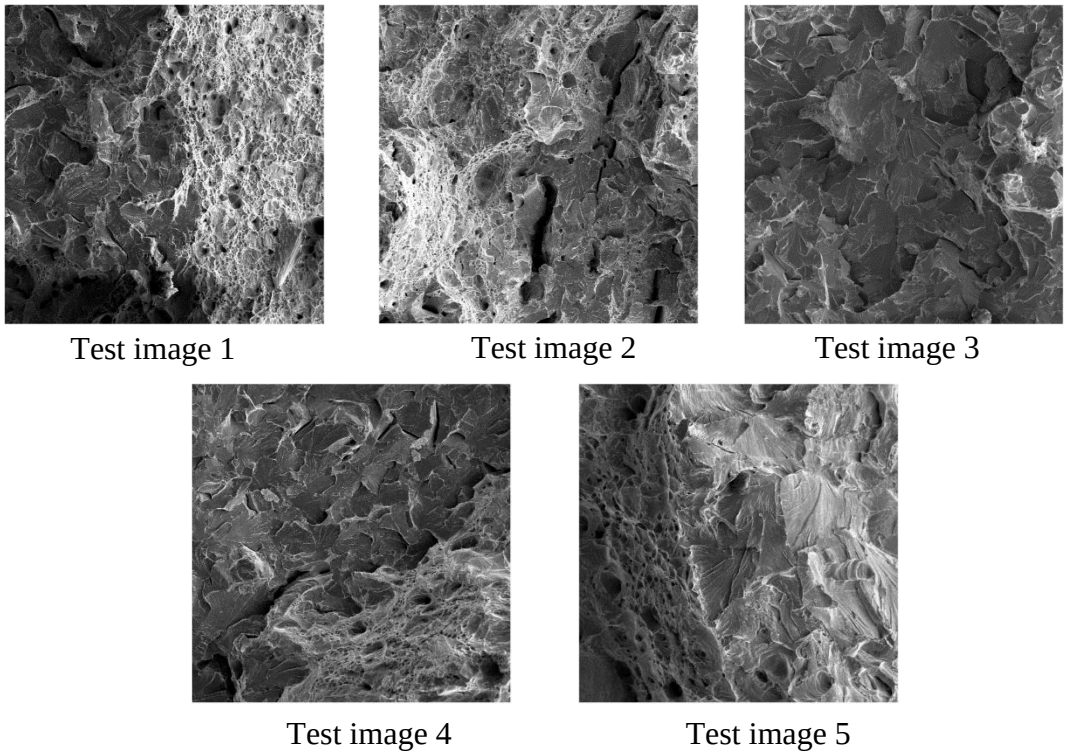


Figure 3.3. Fractographs of ASTM A992 steels on which trained linear discriminant analysis classifier is employed to identify the regions of brittle and ductile fractures.

Block- and pixel-wise approach are the two different approaches that are commonly employed to identify the distinct textured regions of an image consisting of more than one texture [32]. In the block-wise approach, an image is partitioned into several non-overlapping blocks of fixed size (81×81 pixels – same size as that of training images) and the textural features are extracted for each block. The class label of each block is then identified by providing their respective textural features as an input to the trained classifier. Note that the block sizes smaller and greater than 81×81 pixels are not considered in this study as they fail to capture the richness of the texture of the fracture type and include too much redundant information from neighboring regions of other fracture type, respectively [20]. Similar to the block-wise approach,

the pixel-wise approach also involves partitioning of an image into several blocks of fixed size (81×81 pixels). However, the difference between two approaches is that the partitioned blocks are overlapped with each other in the case of pixel-wise approach, i.e. the center pixel of two overlapping blocks are the adjacent pixels of the image that is being partitioned. After partitioning the image into overlapping blocks, the textural features are extracted for each block, and the trained classifier is deployed to identify the class label of each block. Note that the class label obtained for each block in the case of pixel-wise approach is assigned to the central pixel of the block. In this study, both block and pixel-wise approaches are implemented to identify the brittle and ductile fracture regions in the fractographic images, and the results are summarized in the next section.

3.5. Results

3.5.1. LBP Histogram of Brittle and Ductile Fracture Texture

In this section, the LBP histograms of both brittle and ductile fracture images are obtained and are compared to identify the most discriminating uniform/non-uniform circular patterns. In other words, a bin to bin comparison of the histogram is performed. For this purpose, two images – one image corresponding to brittle fracture and the other corresponding to ductile fracture, are arbitrarily selected from the dataset of brittle and ductile images (see Figure 3.2). The LBP histogram obtained for both these images using Eq.7 are shown in Figure 3.2(b). From Figure 3.2(b), it is clear that the occurrence of uniform patterns 3, 4, 5, and 6 are 27%, 45%, 44% and, 22% respectively higher for ductile fracture image texture when compared to the occurrence of same uniform patterns for brittle fracture image texture, and the occurrence of uniform pattern 8 and non-uniform pattern 9 are 15% and 10% higher for the brittle fracture image texture when compared to the occurrence of same patterns in the ductile fracture image texture. While the

uniform patterns 3, 4, 5 and, 6 represents the circular pattern of binary numbers in which digit 1 occurs consecutively 3, 4, 5 and, 6 times, respectively (i.e. for example 01110000, 01111000, 01111100 and 01111110), the uniform pattern 8 represents a circular pattern of binary numbers in which all encoded digits are 0's. Visually, pattern 3 represents an edge, patterns 4, 5 and 6 represents an edge with varying curvature and, pattern 8 represents a dark spot and flat areas with no change in grayscale values [24]. From the histogram of uniform and non-uniform patterns and, their visual interpretations provided above it can be deduced that ductile image texture has higher number of edges with varying texture (microscopic cup and cones) and brittle image texture has higher number of dark spots and non-uniform patterns (river like patterns), which is visually evident from Figure 3.2(a), and hence governs the identification of fracture type.

3.5.2. Performance Assessment of LDA

In this section, the performance of LDA classifier is assessed, i.e. the capability of a classifier to predict the class labels accurately is evaluated. For this purpose, the master dataset is partitioned into two subsets namely, training dataset ($\boldsymbol{S}_1$) and test dataset (validation dataset) ($\boldsymbol{S}_2$) [20]. As mentioned in Section 3.3, a partition ratio of 80:20 is used in this study. While the number 80 represents the percentage of observations randomly sampled from the master dataset to obtain training dataset ($\boldsymbol{S}_1$), the number 20 represents the remaining percentage of observations used as the validation dataset ($\boldsymbol{S}_2$). Followed by the partitioning of the master dataset, LDA classifier is trained on the training dataset ($\boldsymbol{S}_1$) and is later deployed to predict the class labels of the validation dataset ($\boldsymbol{S}_2$). The predicted class labels are cross-validated with the known class labels of the validation dataset and the summary of correct and incorrect classifications are provided in the form of a matrix referred to as confusion matrix ($\boldsymbol{C}$).

Confusion matrix is a $m \times m$ square matrix, where m represents the number of class labels and each element c_{ij} of $\boldsymbol{C}$ represents the frequency of instances from validation dataset that are assigned class label j by the classifier which in reality belongs to class label i [11]. With this, the performance of the classifier is then assessed by evaluating the prediction accuracy, which is defined as the ratio of total number of observations whose class labels are correctly identified to the total number of observations present in the validation dataset ($\boldsymbol{S}_2$). Mathematically it is expressed as

$$A_c = \frac{\sum_{i=1}^{m} c_{ii}}{\sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}} \times 100\%$$

Table 3.1. Confusion matrix for LDA.

		Predicted class label	
	Class Label	Brittle	Ductile
Actual class label	Brittle	0.94	0.06
	Ductile	0.07	0.93

In this study, the LDA classifier is observed to provide an accuracy of 94%, i.e. 94% observations of the test data are correctly classified. The confusion matrix summarizing the correct and incorrect misclassifications is provided in Table 3.1. From Table 3.1, it can be deduced that 6% of brittle fracture images are misclassified as ductile fracture images and 8% of ductile fracture images are misclassified as brittle fracture images.

3.5.3. Validation

In this section, the efficacy of the LBP technique to identify the type of fracture in fractographic images of structural steels with interspersed ductile and brittle fracture regions is demonstrated. For this purpose, five different fractographic images, also referred to as validation

images, are employed in this study (see Figure 3.3), and the histogram of uniform/non-uniform patterns are obtained. These images are acquired from [28-31] and are not used in the training of LDA. LDA classifier trained on the textural features extracted from 50 images of brittle fracture and 50 images of ductile fracture is then deployed, and the brittle and ductile fracture regions in the validation images are identified. To this end, both block- and pixel-wise approaches are employed for classification. The results obtained after classification are shown in Figure 3.4 to Figure 3.8. From Figure 3.4 to Figure 3.8, it can be observed that the prediction obtained through pixel-wise approach is more accurate when compared to the one that is obtained from block-wise approach. When the test images are partitioned into blocks in the block-wise approach, the blocks may either be completely occupied with individual brittle or ductile fracture texture or only a fraction of block may be occupied with brittle and ductile fracture texture. However, during the classification, if the chosen block consists of both brittle and ductile regions, then the trained classifier either assigns it as a brittle fracture or a ductile fracture depending on the visually dominant fracture type present in the block. With this, while a fraction of the pixels in the block is correctly classified, the rest of the pixels are misclassified. The area fraction of brittle and ductile fracture regions for all five images are evaluated and are provided in Table 3.2. The area fractions of brittle and ductile fracture regions predicted through block-wise approach are within 5% of the area fractions predicted by pixel-wise approach. This can be attributed to the relatively small size of the block and compensating misclassification errors in the case of the block-wise approach.

(a) (b)

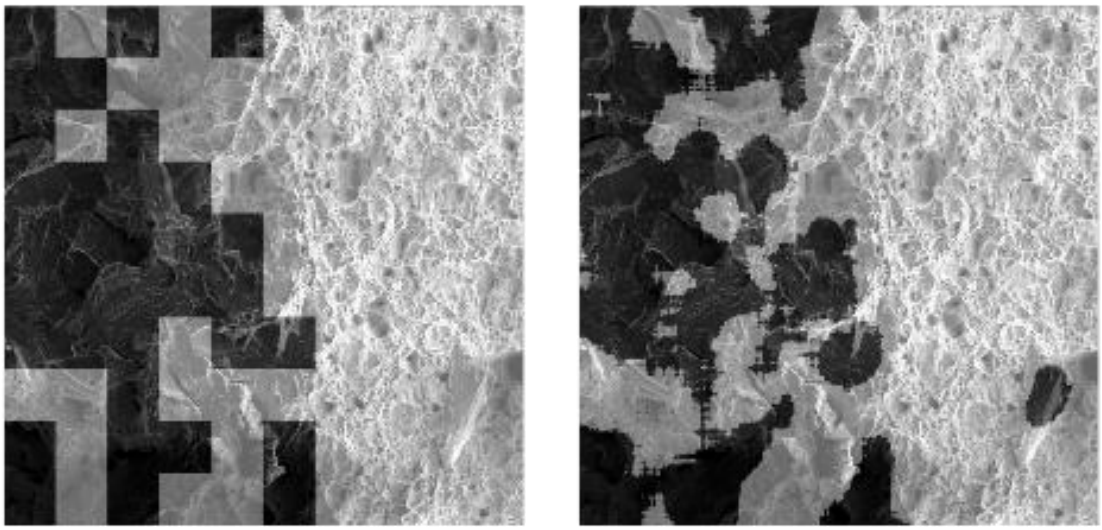


Figure 3.4. Brittle and ductile classification of test image 1: (a) block-wise classification, and (b) pixel-wise classification.

(a) (b)

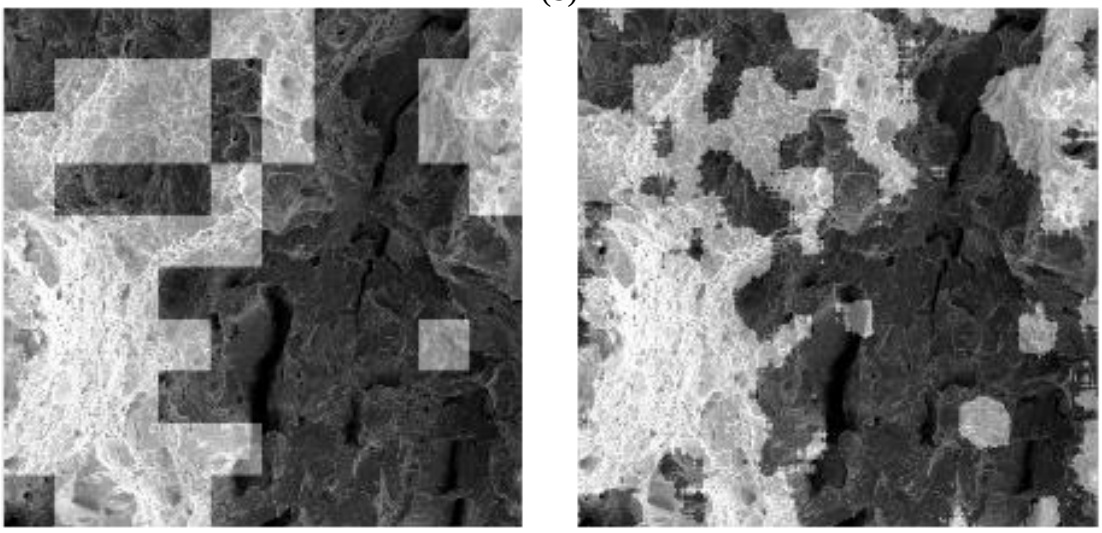


Figure 3.5. Brittle and ductile classification of test image 2: (a) block-wise classification, and (b) pixel-wise classification.

(a) (b)

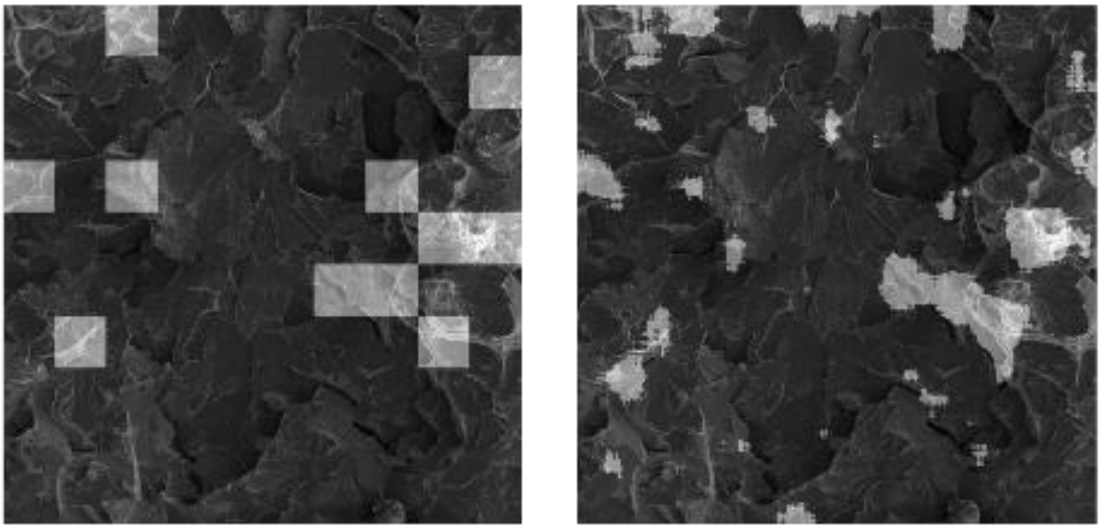


Figure 3.6. Brittle and ductile classification of test image 3: (a) block-wise classification, and (b) pixel-wise classification.

(a) (b)

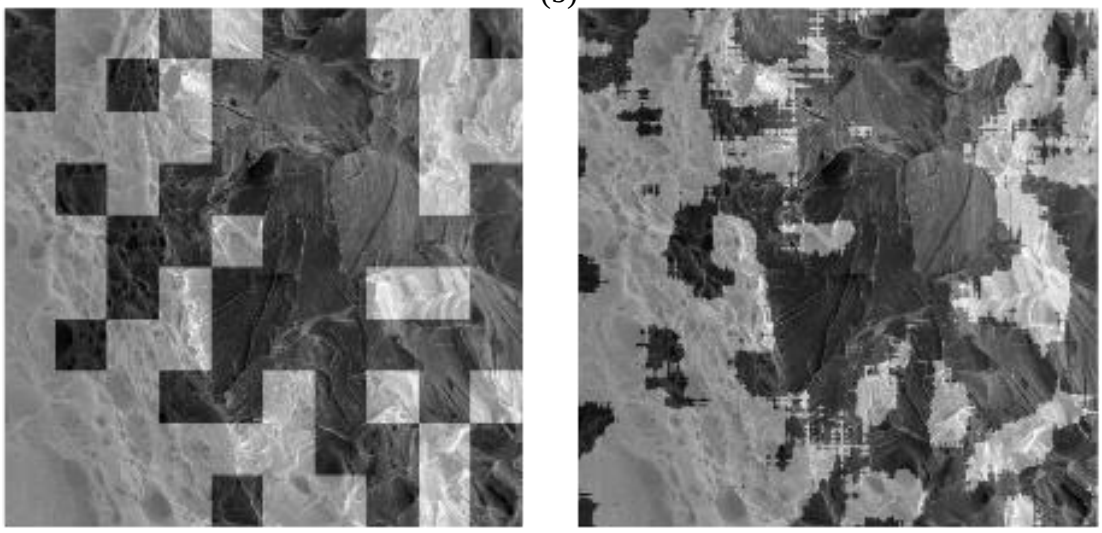


Figure 3.7. Brittle and ductile classification of test image 4: (a) block-wise classification, and (b) pixel-wise classification.

(a) (b)

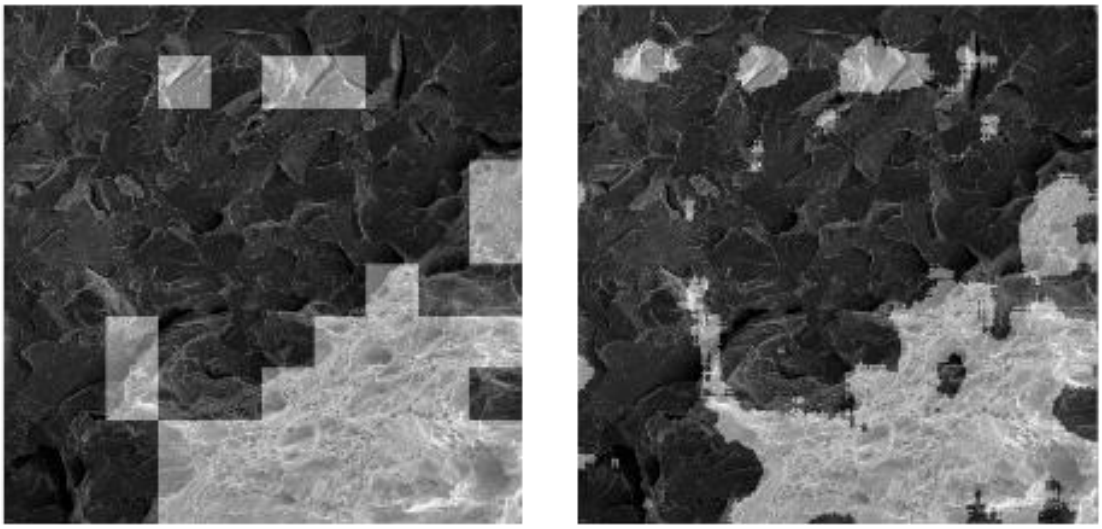


Figure 3.8. Brittle and ductile classification of test image 5: (a) block-wise classification, and (b) pixel-wise classification.

Table 3.2. Area fraction (%) of brittle and ductile fracture evaluated from block- and pixel-wise approaches.

Image No	Block-wise		Pixel-wise	
	Brittle	Ductile	Brittle	Ductile
1	33.4	66.6	35	65
2	57.1	42.9	58	42
3	90.4	9.6	89	11
4	53.8	46.2	52	48
5	71.6	28.4	70	30

3.6. Summary

The goal of this chapter is to automate the identification of the brittle and ductile fracture regions in fractographic images with varying grayscale. For this purpose, a textural feature extraction algorithm, LBP is employed, and a supervised machine learning classifier, LDA is employed. A set of brittle and ductile fracture images of structural steels are acquired, and their textural features are extracted. A master dataset ***D*** comprising of 50 observations of textural features for each fracture type is generated and provided as an input to train the LDA classifier. The performance of a trained LDA classifier is assessed prior to its deployment and an accuracy of 94% was observed. Five different fractographic images of structural steels, that are not a part

of the training data, are chosen and the efficacy of the proposed technique is demonstrated. To classify the type of fracture regions in a given test image, two types of approaches are used in this study, namely, block-wise classification and pixel-wise classification. Pixel-wise classification resulted in more accurate classification when compared to the block wise classification. However, block-wise classification is computationally inexpensive, and the area fractions obtained from both the methods are more or less the same. Note that this methodology can also be extended to identify fatigue fracture from fractographic images. Fatigue fractures are characterized by the presence of striations (high-cycle fatigue) [6] or elongated cup and cones (ultra-low cycle fatigue) [33] at microscale exhibiting a distinct texture. However, sufficient number of fatigue fracture training images are required to train ML algorithms to identify fatigue fracture in addition to the brittle and ductile fracture in fractographic images.

3.7. References

1. Clifton C, Bruneau M, MacRae G, Leon R, Fussell A. Steel structures damage from the Christchurch earthquake series of 2010 and 2011. Bulletin of the New Zealand Society for Earthquake Engineering. 2011;44:297-318.
2. Russo FM, Mertz DR, Frank KH, Wilson KE. Design and Evaluation of Steel Bridges for Fatigue and Fracture–Reference Manual. 2016.
3. Naik Dayakar L, Kiran R. Data Mining and Equi-Accident Zones for US Pipeline Accidents. Journal of Pipeline Systems Engineering and Practice. 2018;9:04018019.
4. El-Magd E, Gese H, Tham R, Hooputra H, Werner H. Fracture criteria for automobile crashworthiness simulation of wrought aluminium alloy components. Materialwissenschaft und Werkstofftechnik: Materials Science and Engineering Technology. 2001;32:712-24.

5. Adib A, Baptista C, Barboza M, Haga C, Marques C. Aircraft engine bleed system tubes: Material and failure mode analysis. Engineering Failure Analysis. 2007;14:1605-17.
6. Anderson TL. Fracture Mechanics: fundamentals and applications. 3rd ed. Boca Raton, FL: CRC Press; 2004.
7. Kanvinde AM. Micromechanical simulation of earthquake induced fracture in steel structures. Stanford, CA: Stanford University; 2004.
8. Wen H, Mahmoud H. New Model for Ductile Fracture of Metal Alloys. I: Monotonic Loading. Journal of Engineering Mechanics. 2016;142:04015088.
9. Jia L-J, Ge H. Ductile Crack Propagation under Monotonic Loading. Ultra-low-Cycle Fatigue Failure of Metal Structures under Strong Earthquakes: Springer; 2019. 71-95.
10. Sajid HU, Kiran R. Influence of stress concentration and cooling methods on post-fire mechanical behavior of ASTM A36 steels. Construction and Building Materials. 2018;186:920-45.
11. Uthaisangsuk V, Muenstermann S, Prahl U, Bleck W, Schmitz HP, Pretorius T. A study of microcrack formation in multiphase steel using representative volume element and damage mechanics. Computational Materials Science. 2011;50:1225-32.
12. Uthaisangsuk V, Prahl U, Bleck W. Modelling of damage and failure in multiphase high strength DP and TRIP steels. Engineering Fracture Mechanics. 2011;78:469-86.
13. Srivastava A, Ghassemi-Armaki H, Sung H, Chen P, Kumar S, Bower AF. Micromechanics of plastic deformation and phase transformation in a three-phase TRIP-assisted advanced high strength steel: Experiments and modeling. Journal of the Mechanics and Physics of Solids. 2015;78:46-69.

14. Papasidero J, Doquet V, Mohr D. Ductile fracture of aluminum 2024-T351 under proportional and non-proportional multi-axial loading: Bao–Wierzbicki results revisited. International Journal of Solids and Structures. 2015;69–70:459-74.

15. Luo M, Dunand M, Mohr D. Experiments and modeling of anisotropic aluminum extrusions under multi-axial loading–Part II: Ductile fracture. International Journal of Plasticity. 2012;32:36-58.

16. Giglio M, Manes A, Vigano F. Ductile fracture locus of Ti–6Al–4V titanium alloy. International Journal of Mechanical Sciences. 2012;54:121-35.

17. Pakhira Malay K. Digital image processing and pattern recognition: PHI Learning Private Limited; 2011.

18. Jayaraman S, Esakkirajan S, Veerakumar T. Digital Image Processing. New Delhi: Tata McGraw Hill Education Private Limited; 2009.

19. Kosarevych RY, Student O, Svirs'ka L, Rusyn B, Nykyforchyn H. Computer analysis of characteristic elements of fractographic images. Materials Science. 2013;48:474-81.

20. Naik DL, Sajid HU, Kiran R. Texture-Based Metallurgical Phase Identification in Structural Steels: A Supervised Machine Learning Approach. Metals. 2019;9:546.

21. Bastidas-Rodriguez M, Prieto-Ortiz F, Espejo E. Fractographic classification in metallic materials by using computer vision. Engineering Failure Analysis. 2016;59:237-52.

22. Dutta S, Das A, Barat K, Roy H. Automatic characterization of fracture surfaces of AISI 304LN stainless steel using image texture analysis. Measurement. 2012;45:1140-50.

23. Gad AF. Practical Computer Vision Applications Using Deep Learning with CNNs.

24. Ojala T, Pietikäinen M, Mäenpää T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on Pattern Analysis & Machine Intelligence. 2002:971-87.

25. Rencher AC. Methods of multivariate analysis: John Wiley & Sons; 2003.

26. Naik DL, Kiran R. Naïve Bayes classifier, multivariate linear regression and experimental testing for classification and characterization of wheat straw based on mechanical properties. Industrial Crops and Products. 2018;112:434-48.

27. Johnson RA, Wichern DW. Applied multivariate statistical analysis: Prentice hall Upper Saddle River, NJ; 2002.

28. Kiran R, Khandelwal K. A micromechanical model for ductile fracture prediction in ASTM A992 steels. Engineering Fracture Mechanics. 2013;102:101-17.

29. Kiran R, Khandelwal K. Experimental studies and models for ductile fracture in ASTM A992 steels at high triaxiality. Journal of Structural Engineering. 2013;140:04013044.

30. Kiran R, Khandelwal K. A triaxiality and Lode parameter dependent ductile fracture criterion. Engineering Fracture Mechanics. 2014;128:121-38.

31. Kiran R, Khandelwal K. A coupled microvoid elongation and dilation based ductile fracture model for structural steels. Engineering Fracture Mechanics. 2015;145:15-42.

32. Li J, Wang JZ, Wiederhold G. Classification of textured and non-textured images using region segmentation. Proceedings 2000 International Conference on Image Processing (Cat No 00CH37101): IEEE; 2000; 754-7.

33. Kiran R, Khandelwal K. A micromechanical cyclic void growth model for ultra-low cycle fatigue. International Journal of Fatigue. 2015;70:24-37.

4. DETECTION OF CORROSION-INDICATING OXIDATION PRODUCT COLORS IN STEEL BRIDGES UNDER VARYING ILLUMINATIONS, SHADOWS, AND WETTING CONDITIONS[3]

4.1. Introduction

Corrosion damage is found to play a vital role in the overall maintenance cost of the steel structures [1-3]. In the United States, the average annual cost of corrosion damage for steel bridges is estimated to be ~$10.15 billion [3]. Detection of corrosion in its early stages not only results in the reduction of maintenance costs but also increases the life of the structures [4]. Currently, either human inspection or non-destructive techniques like eddy current technique [5], ultrasonic inspection [6, 7], acoustic emission technique [8, 9], vibration analysis [10], radiography [11], thermography [12], optical inspection [13], etc. are employed to monitor and identify the corrosion damage in the steel structures. Although each of the above-mentioned techniques have their own advantages, the optical inspection technique is most commonly preferred owing to its simplicity and ease of interpretation.

In optical inspection, digital images of structures are first acquired on-site and are then analyzed using image processing techniques off-site to detect the corrosion. In recent times, various approaches have been proposed by researchers to detect the corrosion in steel structures using digital images [14-16]. Most of these approaches included either acquisition of grayscale images or a color image of the corroded steel structure under uniform illumination conditions

[3] This chapter is based on the paper "Detection of Corrosion-Indicating Oxidation Product Colors in Steel Bridges under Varying Illuminations, Shadows, and Wetting Conditions." Metals 10 (11) (2020): 1439. https://doi.org/10.3390/met10111439. The material in this chapter was co-authored by Dayakar Naik Lavadiya (DNL), Ravi Kiran Yellavajjala (RK), Genda Chen (GC) and Hizb Ullah Sajid (HUS). Conceptualization, R.K. and D.L.N.; formal analysis, D.L.N.; funding acquisition, R.K.; investigation, R.K., G.C.; methodology, D.L.N., H.U.S., R.K., and G.C.; project administration, R.K.; resources, R.K., and G.C.; software, D.L.N.; supervision, R.K., and G.C.; validation, D.L.N. and R.K.; writing—original draft, D.L.N.; writing—review & editing, D.L.N., R.K., and G.C.

(i.e., same time of the day, without shadows). Color is defined as a small portion of the electromagnetic spectrum that is visible to the human eye and covers wavelength in the range of 380nm to 740nm [17]. When compared to grayscale images, color images have more information i.e. chromaticity and luminosity [18]. Chromaticity refers to the combination of the dominant wavelength of the visible light (called as hue) reflected from the material surface and the purity (saturation) associated with it, and luminosity refers to the intensity of light per unit area of the light source. For identifying corroded portions in the color images the distinguishable features such as color [19], texture [20], and edge are extracted from the images. For instance, in the study conducted by Shen et al. [16] color components extracted from 19 different color spaces were considered. 'CIE La*b*' color space was reported to yield a satisfactory result. In another study conducted by Medeiros et al. [21], both color and textural features were included and linear discriminant analysis (LDA) classifier was employed to identify corrosion. Ranjan et. al. [22] proposed an edge-based corrosion identification wherein various edge filters were employed to detect boundaries between corroded and non-corroded regions in the images. Lee et al [23] performed a multivariate statistical analysis of three color channels Red (R), Green (G) and Blue (B) to identify the corrosion in steel bridges coated in blue paint.

Further Chen et. al. [14-16, 24] investigated the effect of artificially generated non-uniform illumination on corrosion detection. Three different approaches were proposed by the authors. In the first approach, a neuro-fuzzy recognition algorithm (NFRA) was implemented which automatically generates three optimal threshold values for later image thresholding artificial neural network (ANN) and a fuzzy adjustment system. In the second approach, the authors have investigated the use of 14 different color spaces for corrosion detection and proposed an adaptive ellipse to segment background coating and corrosion rust. 'CIE La*b*' was

identified as the best color space. In the third approach, the authors have integrated color image processing, Fourier transforms and Support Vector Machines (SVM) to identify the corrosion in the bridges with a red and brown color background. In another set of studies, both Ghanta et. al. [25] and Nelson et. al. [26] implemented a wavelet transforms based approach to identify corrosion in steel bridges and shipboard ballast tanks, respectively. Son et. al. [27] used 'HSV' color space and C4.5 decision tree algorithm to identify corrosion. It is important to note that, in reality, the illumination of natural daylight does not remain the same for the entire day. Moreover, the steel structures have self-shadows (shadows from the structural components) and oil/ water wetted spots (for example, bridges). In the recent study carried out by Liao et al. [28], both 'RGB' and 'HSV' color spaces were adopted in conjunction with the least squares-support vector machine (LS-SVM) based technique to identify corrosion under shaded area generated by natural light. However, the proposed approach had few limitations as reported by the authors: (1) the inefficiency of the approach to predict dark corroded areas and (2) the proposed threshold values may vary for other images that are not considered in their study. From a practical perspective, there is a need to develop a more robust technique that can be used to identify the corrosion in steel structures using images taken under varying illuminations, dark shadows, water, and oil wetting.

The aim of the current chapter is to detect corrosion in steel structures under ambient lighting conditions such as varying illuminations, shadows, and water and oil wetting. To this end, four different color spaces are employed, and a multi-layer perceptron (MLP) is configured and trained with the color features extracted from the lab generated corrosion images. Subsequently, the trained MLP is deployed on the field generated images (i.e. a steel bridge) and the corrosion is detected. Note that the scope of this study is only limited to the corrosion

identification on the surface of the steel and not the cross-section. The main emphasis of the current study is to determine the most suitable combination of color space and an MLP configuration from the laboratory-generated image dataset which can yield correct predictions in the case of images acquired in a real-world scenario. Rest of the manuscript is organized as follows: materials and methods used to generate images of corroded plates in the laboratory is described in Section 4.2, extracting the color features of corroded/non-corroded pixels and building a training, validation and test dataset is described in Section 4.3, details of MLP configuration is provided in Section 4.4, performance assessment and efficacy of the trained model is discussed in Section 4.5, and conclusions are provided in Section 4.6.

4.2. Laboratory Generated Corrosion Images

In this section, the procedure adopted for acquiring the lab generated corrosion images is described.

4.2.1. Accelerated Corrosion Tests and Image Acquisition

Six ASTM A36 structural steel plates (see Table 4.1 [29]) with dimensions 7.6 cm × 7.6 cm × 0.4 cm are subjected to accelerated corrosion. To this end, ASTM A36 structural steel plates are placed inside a salt spray chamber at an angle of 20° to the vertical and are continuously exposed to 3.5 wt. % sodium chloride (NaCl) solution mist for 12 hours. The corroded steel plates are then removed from the salt spray chamber and are gently cleaned with warm water to remove the excess salt traces and then air-dried. The detailed description of the accelerated corrosion protocol can be found elsewhere [30]. While the surfaces of two of the plates are completely exposed to corrosion (see Figure 4.1), the surfaces of rest of the plates are only partially exposed to corroded (Figure 4.2) i.e. random patches of corrosion are induced on the plate by preventing the interaction between the corrosive media and the surface with the help

of adhesive tapes. At this juncture, it is important to note that the plate surfaces completely exposed to corrosion are used for generating training and validation datasets and the plate surfaces partially exposed to corrosion are reserved for generating a test image database (see Section 4.3.4). Here on the plate surfaces completely exposed to corrosion will be referred to as fully corroded plates and the plate surfaces partially exposed to corrosion will be referred to as partially corroded plates. Details of generating training and test dataset are provided in Section 4.3.

A mobile camera with a digital resolution of 12MP (4032×3024) is employed to acquire the images of the lab corroded steel plates. Note that the images were acquired outdoors while the corroded plates are directly exposed to sunlight. As per the manufacturer specifications, the size of the sensor and the size of the image pixel are 1/3.6" and 1.0μm, respectively and the camera's angular field of view (FOV) is 45°. For image acquisition, the camera is mounted on a small tripod and is placed parallel to the target surface (corroded plate) at a fixed distance of 8.0 inches. The camera was then operated with a fixed setting of ISO 400 and a shutter speed of 1/350th of a second. In addition to this, the color tone was set to a standard color tone and white balance was set to 5500K (daylight). The images of the corroded steel plates are then acquired at different time intervals during the day i.e. 5:00 AM, 9:00 AM, 12:00 PM, 3:00 PM and 6:00 PM such that varying illuminations of daylight are captured in the images (see Figure 4.1 (a)-(b)). Furthermore, the images of fully and partially corroded plates under shadows, water wetting, and oil wetting are also acquired. Shadows, water wetting, and oil wetting are commonly observed in steel structures such as bridges, water tanks, bunkers, and silos. Shadows are formed by blocking the light falling on the structure by surrounding vegetation, constructions or the structural components that are being monitored. The water and oil wetting may occur due to rainfall and oil

leak from cargo, respectively. The images of fully and partially corroded plates with shadows are acquired by blocking the light falling on the plate with an opaque object (see Figure 4.1 (c)).

Table 4.1. ASTM A36 composition.

Composition	Symbol	Wt. %
Carbon	C	0.25-0.29
Iron	Fe	98
Copper	Cu	0.2
Manganese	Mn	1.03
Phosphorous	P	0.04
Silicon	Si	0.28
Sulfur	S	0.05

For acquiring the images of the fully and partially corroded plates wetted with water, the corroded steel plates are sprayed with the water using a spray jug. Enough distance between corroded plates and spray jug is maintained such that only a drizzle of fine droplets of water is sprayed on the plate and sufficient care is exercised to avoid the formation of bigger water droplets on the surface of the plate. A similar procedure is then adopted for acquiring the images of fully and partially corroded plates under oil wetting except that oil is used in the place of water as a wetting agent. The images of all the corroded steel plates acquired using the mobile camera are shown in Figure 4.1 and Figure 4.2. Note that the images are not directly used as the input to the MLP in this study i.e. the images are not directly considered as the training and validation datasets. Instead, color features of corroded and non-corroded pixels of the plate are extracted from the images to generate a training dataset and then used to train various MLP configurations. For instance, if the color features of 1000 corroded pixels are chosen from one image, then the training dataset is said to have 1000 observations. The color feature extraction process is described next.

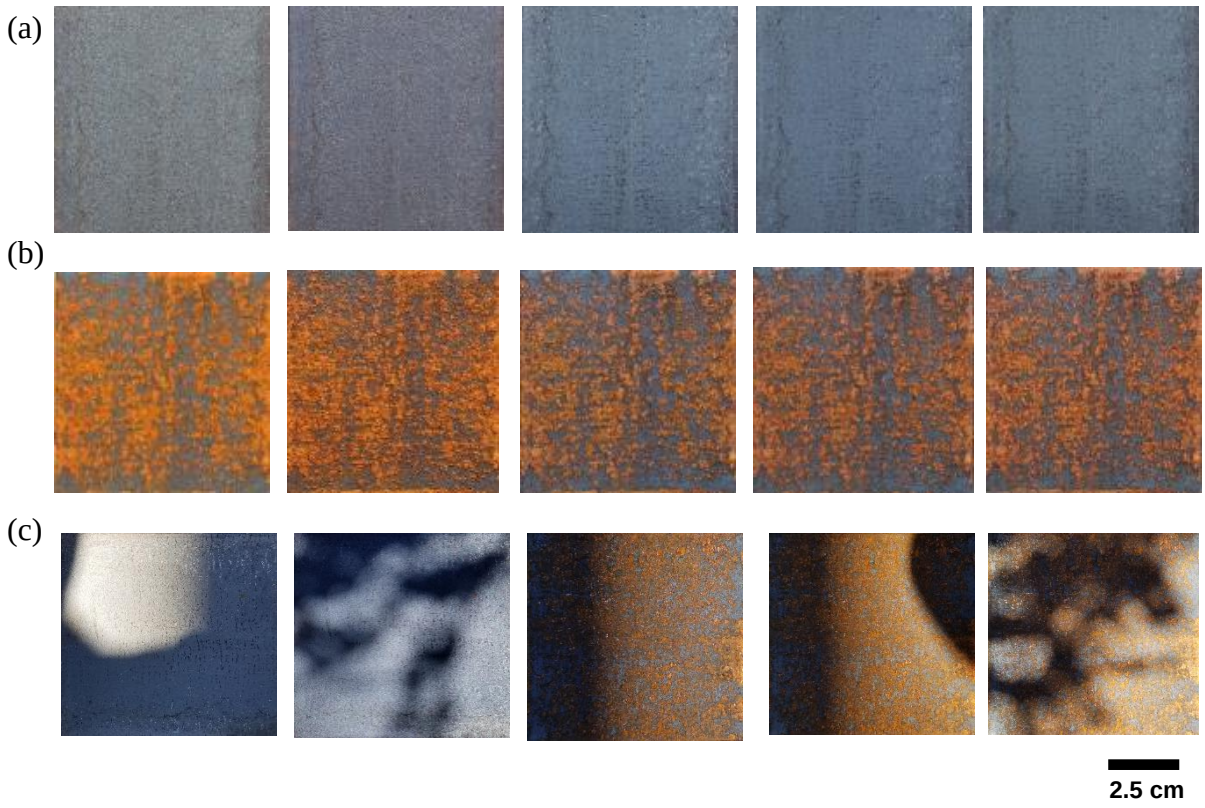


Figure 4.1. Non-corroded and corroded steel plates used for training purposes. (a) top row (left to right) includes images of non-corroded plates acquired at varying illuminations under natural daylight, (b) second row (left to right) includes images of corroded plates acquired under illuminations similar to that of non-corroded plates and, (c) third row (left to right) includes images of both corroded and non-corroded plates acquired under casted shadows.

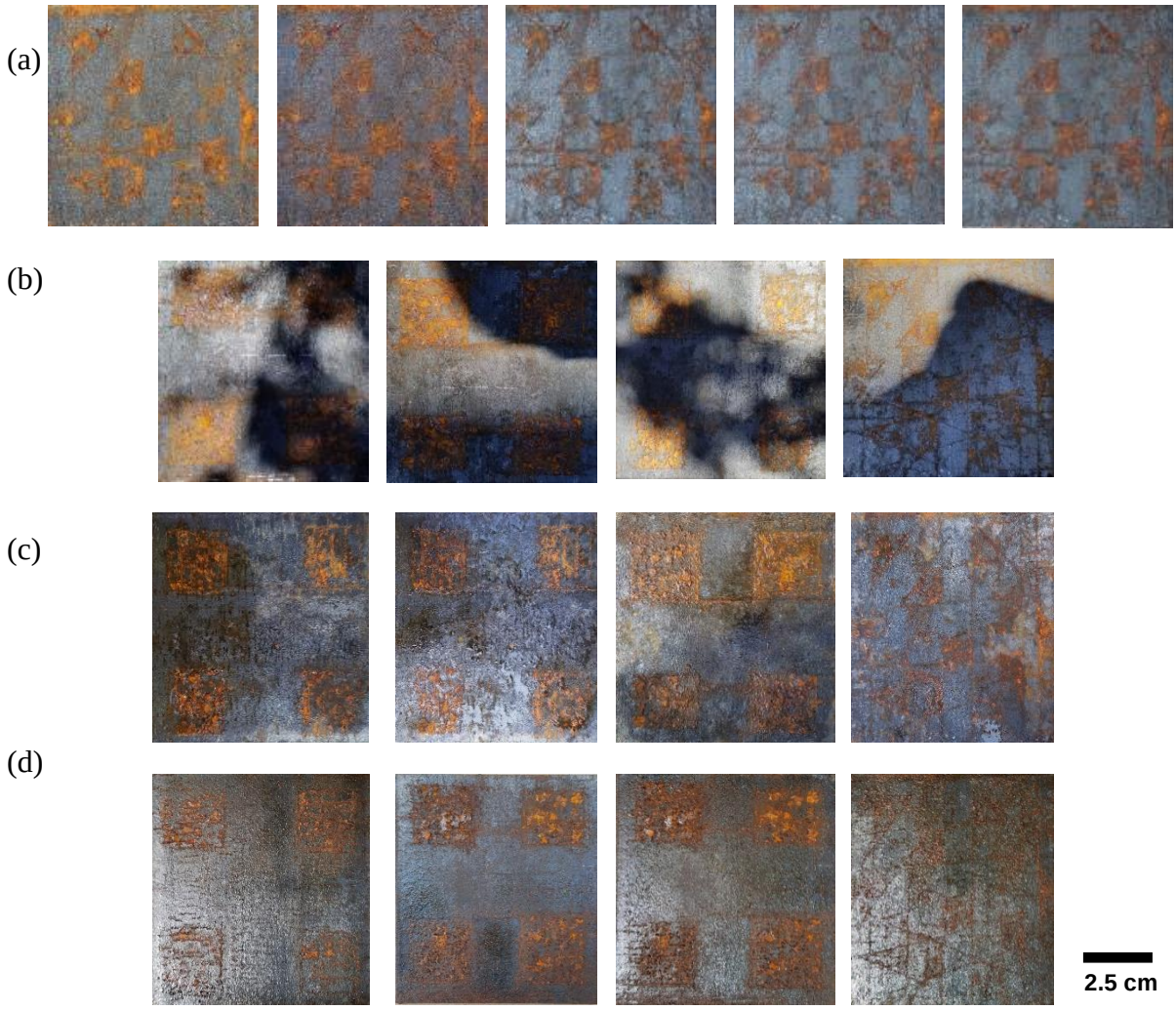


Figure 4.2. Images of partially corroded steel plates used for testing: (a) acquired at different illuminations of natural daylight, (b) shadows, (c) water wetting, and (d) oil wetting.

4.3. Color Feature Extraction and Dataset Generation

In this study, we hypothesize that the color features alone can be used to identify corrosion in steel bridges. Hence color features are extracted from the lab generated images to generate training, validation and test datasets that will be used to train, validate and demonstrate the performance of MLP, respectively. In this section, a brief overview of color spaces used in this study is provided and the process of generating training, validation and test datasets is described.

4.3.1. Color Spaces and Color Features

Color space is a mathematical abstraction introduced by *Commission Internationale de l'éclairage* (CIE) [31] to numerically express color as a tuple of numbers. It is regarded as the color coordinate system wherein each color feature is plotted along a coordinate axis. In this study, four-color spaces namely 'RGB' (also called as primary color space), 'rgb' (also called as normalized color space), 'HSV' (also called as perceptual color space) and 'CIE La*b*' (also called as uniform color space) are considered. A brief description of these four-color spaces is provided next.

'RGB' color space

In 'RGB' color space, the primary colors Red (R), Green (G) and Blue (B) are considered as the color features whose intensities range from 0 to 255. These three-color intensities when plotted in a three-dimensional Cartesian coordinate system with primary colors (R, G, B) as the x, y and z axis, respectively, forms the 'RGB' color space. In the 'RGB' color space, all possible colors are encompassed in a cube with dimensions $(255 \times 255 \times 255)$ in the first positive octant of the coordinate system (see Figure 4.3 (a)). Each point enclosed in this cube represents a unique color. While the point $(0,0,0)$ represents the black color, the point $(255, 255, 255)$ represents the white color. The points that fall on the line that joins origin $(0, 0, 0)$ and its diagonally opposite point $(255, 255, 255)$ represent different gray shades. Note that the color images obtained from the mobile or Digital Single-Lens Reflex (DSLR) cameras generally have intensities of R, G, and B as the pixel values. In the 'RGB' color space the chromaticity is coupled with the luminosity in the R, G, and B features and are sensitive to non-uniform illuminations.

'rgb' color space

Unlike 'RGB' color space, in 'rgb' color space the normalized intensities of primary colors Red, Green and Blue are considered as the color features. The magnitude of their intensities range from 0 to 1 and when plotted in a three-dimensional Cartesian coordinate system in which the normalized primary colors (r, g, b) are the x, y and z axis, respectively, each point represents a single unique color. In the 'rgb' color space, all possible colors are encompassed with-in the surface of a sphere in the first octant of the coordinate system where the radius of a sphere is 1 (see Figure 4.3 (b)). To determine the values of 'r', 'g' and 'b', from 'R', 'G', and 'B' Eq. 4.1is used. In 'rgb' color space the chromaticity and luminosity are decoupled from the r, g, and b features and, unlike 'RGB' color space the 'rgb' color space is insensitive to non-uniform illuminations [32].

$$r = \frac{R}{\sqrt{R^2 + G^2 + B^2}}; g = \frac{G}{\sqrt{R^2 + G^2 + B^2}}; b = \frac{B}{\sqrt{R^2 + G^2 + B^2}} \tag{4.1}$$

'HSV' color space

In 'HSV' color space, hue (H), Saturation (S) and Value (V) are considered as the color features. Hue is defined as the pure color, and its magnitude ranges from 0 to 1. Saturation is defined as the amount of impurity or white color added to hue and its magnitude also ranges from 0 to 1. Value is defined as the brightness or intensity of light, and its magnitude ranges from 0 to 255. On a cylindrical coordinate system these color features i.e. Hue, Saturation and Value, represents the angle (θ), radius (r) and vertical height (z), respectively. All the colors in the HSV color space fit in an inverted cone (see Figure 4.3(c)) [33]. Given, the intensities of R, G, and B, the magnitude of H, S, and V can be evaluated using Eq. 4.2 – 4.5. Similar to 'rgb' color space, the chromaticity and luminosity are decoupled in the 'HSV' color space and is robust to non-uniform illuminations. However, in 'HSV' color space the chromaticity is

represented as two separate features ‘Hue’ and ‘Saturation’. Moreover, ‘Hue’ is undefined when the intensities of R, G and, B are same.

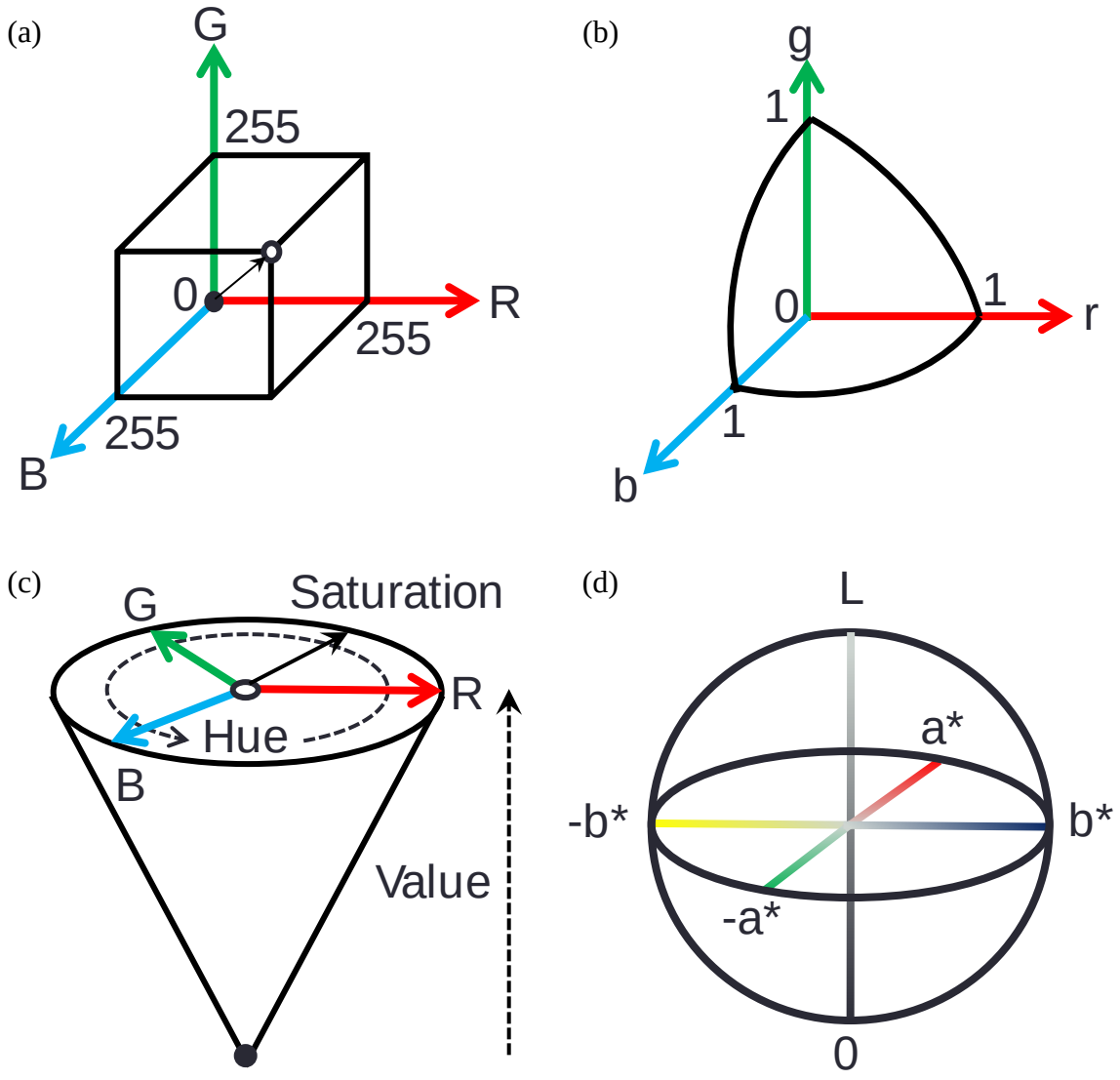


Figure 4.3. Color spaces in three-dimensional coordinate systems: (a) ‘RGB’, (b) ‘rgb’, (c) ‘HSV’ and (d) ‘CIE La*b*’.

Let $\alpha_{max} = \max(R, G, B)$; $\alpha_{min} = \min(R, G, B)$; $\delta = \alpha_{max} - \alpha_{min}$

Then

$$V = \alpha_{max} \tag{4.2}$$

$$S = \begin{cases} 0, & \delta = 0 \\ \dfrac{\delta}{\alpha_{max}}, & \delta \neq 0 \end{cases} \tag{4.3}$$

$$H' = \begin{cases} 5 + \dfrac{(\alpha_{max} - B)}{\delta} \; if \; \alpha_{min} = G \\ 1 - \dfrac{(\alpha_{max} - G)}{\delta} \end{cases} \alpha_{max} = R$$
$$\begin{cases} 1 + \dfrac{(\alpha_{max} - R)}{\delta} \; if \; \alpha_{min} = B \\ 3 - \dfrac{(\alpha_{max} - B)}{\delta} \end{cases} \alpha_{max} = G \tag{4.4}$$
$$\begin{cases} 3 + \dfrac{(\alpha_{max} - G)}{\delta} \; if \; \alpha_{min} = R \\ 5 - \dfrac{(\alpha_{max} - R)}{\delta} \end{cases} \alpha_{max} = B$$

$$H = \frac{H'}{6} \tag{4.5}$$

*'CIE La*b*' color space*

In 'CIE La*b*' color space, Lightness (L), and opponent colors (a* and b*) are considered as the color features. While L represents the intensity of light and its magnitude ranges from 0 (white) to 100 (black), a* and b* represents the opponent colors red-green and blue-yellow respectively and its magnitude ranges from -128 to +128. The negative most value of a*, i.e. -128, represents the red color and positive most value of a*, i.e. 128, represents the green color. Similarly, the negative most value of b*, i.e. -128, represents the blue color and positive most value of b*, i.e. 128, represents the yellow color. These three-color features when plotted in a three-dimensional Cartesian coordinate system with (a^*, b^*, L) as the x, y and z axis, respectively, forms the 'La*b*' color space. In the 'CIE La*b*' color space, all possible colors are encompassed in an ellipsoid (see Figure 4.3 (d)) [34]. Each point enclosed in the ellipsoid represents a unique color. Given, the intensities of R, G, and B, the magnitude of L, a* and b*can

be evaluated using Eq. 4.6. Similar to 'rgb' and 'HSV' color space, the chromaticity and luminosity are decoupled in 'CIE La*b*' color space and are insensitive to non-uniform illuminations. In addition to this, note that the 'CIE La*b*' color space is device independent and mimics the way humans perceive the colors [32].

$$L = 116f\left(\frac{Y}{Y_n}\right) - 16; a^* = 500\left(f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right);\ b^* = 200\left(f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right) \quad (4.6)$$

where,

$$\text{if } \frac{X}{X_n}, \frac{Y}{Y_n}, \frac{Z}{Z_n} \text{ are replaced by } 't' \text{ then}$$

$$f(t) = \begin{cases} \sqrt[3]{t} \text{ if } t > 0.00856 \\ 7.787t + \dfrac{16}{116} \text{ otherwise} \end{cases}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

The definitions for $X_n, Y_n, Z_n,\ R', G'$ and B' can be found in the cited reference [17, 47, 48].

4.3.2. Training Dataset

To obtain the training dataset the color features of corroded and non-corroded pixels are extracted from the lab generated images. Specifically, fully corroded steel plate images acquired under varying illuminations are considered for this purpose (see Section 4.2). For the extraction of color features the pixel information i.e. the intensity of primary colors R, G and B of the image is obtained in the MATLAB® and the Eq. (4.1) - (4.6) provided in previous section is used to determine the color features in the other color spaces. A total of 5000 instances are generated for the training dataset among which 50% of the instances belonged to the 'corrosion' class and the rest of them belonged to the 'non-corrosion' class. Labelling of the instances as corrosion/ non-corrosion is solely based on visual observation and judgement of the authors. As both 'Non-

corrosion' and 'Corrosion' exhibit distinguishable color for trained human vision, the reproducibility of labeling can be assured. However, a slight chance for subjectivity cannot be ruled out. The overall goal of this body of research is to automate the optical corrosion detection by addressing the challenges associated with identifying corrosion under ambient lighting conditions. Hence, no corrosion characterization tests were performed to cross-validate the corrosion/ non-corrosion pixels.

4.3.3. Validation Dataset

The validation dataset is used to validate a multi-layer perceptron that is trained using the color features extracted from lab acquired images of corroded plates under different illuminations (training dataset). The validation dataset is generated by extracting the color features of corroded and non-corroded pixels from the lab acquired images of corroded plates under shadows i.e. fully corroded steel plate images acquired under the shadow (see Figure 4.1 (c)). Evaluation of the performance of the trained MLP on the validation dataset is seen as the first line of validation in this study and is used to identify an appropriate combination of MLP configuration and color features that can be ultimately used to detect corrosion in more challenging real-world scenarios. Note that none of the data instances in the validation dataset are used for training purposes. A total of 2064 instances are generated for the validation dataset, among which 50% of instances belong to 'corrosion' class and the rest of them belong to 'non-corrosion' class.

4.3.4. Test Image Database

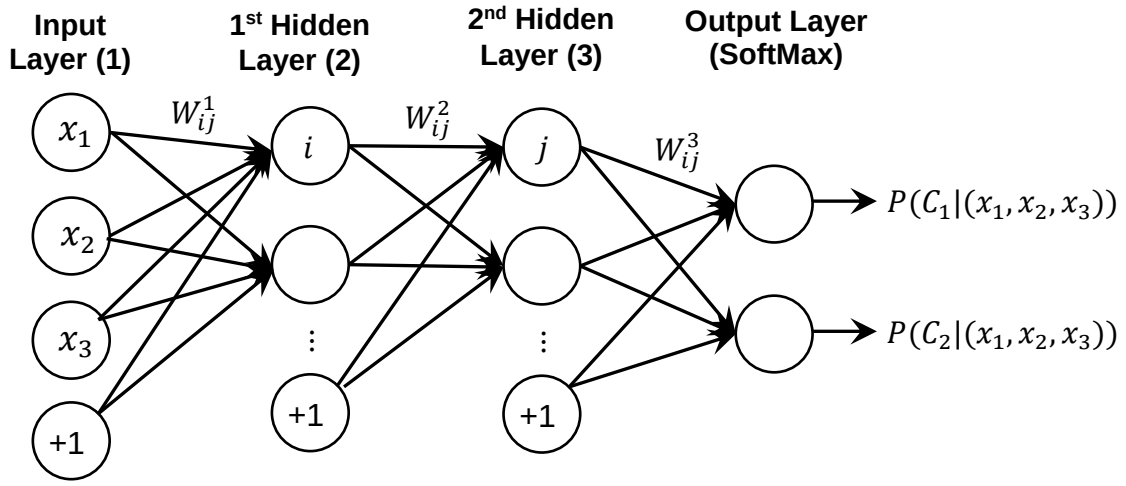
The test image database will be used to test and demonstrate the efficacy of the trained and validated MLP to detect corrosion in digital images that have dark shadows and are acquired by a different imaging sensor. In other words, the generalization capability of the trained MLP to

predict corrosion will be verified. The test image database generated herein includes the lab acquired images of partially corroded steel plates exposed to natural daylight illuminations (different from the ones used for training dataset), shadows and, water and oil wetting conditions and the images of a steel girder bridge located in Fargo-Moorhead (Minnesota) area acquired on-site. Note that the sensor employed for acquiring images in the laboratory and on-site bridge images are not the same. A Digital Single-Lens Reflex (DSLR) camera is used for acquiring the on-site bridge images. The DSLR camera has a resolution of 18MP (5184×3456) with a pixel size of 4.3µm and a 22.3mm x 14.9mm sensor size, and a sensor ratio of 3:2. The bridge images acquired on-site consist of the steel plate girders with naturally varying illumination and self-shadows. Especially, the images of the bottom side of the deck of the bridge had dark self-shadows. Note that in order to detect corrosion portions in the test images, the color features of each pixel in the test image have to be extracted first and then labeled by the trained MLP.

4.4. Multi-Layer Perceptron

A multi-layer feed-forward neural network, also referred to as multi-layer perceptron (MLP), is programmed in MATLAB® and is trained, validated and tested in this study. The mathematical underpinnings and detailed description of multi-layer feed-forward neural network can be found elsewhere [35-38]. The MLP employed herein receives the color features of a pixel (see Section 4.3.1) as an input and delivers its class label (corrosion/non-corrosion) as an output (see Figure 4.4). Since the suitable configuration of MLP is not known apriori, four different configurations are explored: (1) one hidden layer (HL) with 2 neurons (1st HL(2N)), (2) one hidden layer with 4 neurons (1st HL(4N)), (3) two hidden layers with 2 neurons in each layer (1st HL(2N)-2nd HL(2N)) and (4) three hidden layers with 50 neurons in first layer, 10 neurons in second layer and 4 neurons in third layer (1st HL(50N)-2nd HL(10N)-3rd HL(4N)). Note that the

selection of these configurations is based on thumb rules provided in the following references [39, 40]. Sigmoid function is chosen as an activation function for all the neurons in the MLP except the output layer wherein SoftMax function is used and the maximum number of epochs is fixed to 1000. The maximum number of epochs is not exceeded in any case during the training phase. Furthermore, the MLP is trained only once (until the weights converged), and no re-training is required when used on in-house generated validation dataset or test image database. The back-propagation algorithm is used for determining the weights of the MLP.



W_{ij}^l – Weight connecting the neuron i in the preceding layer l and the neuron j in the succeeding layer $(l + 1)$.

$P(C_1|(x_1, x_2, x_3))$ – Probability of the given instance (x_1, x_2, x_3) belonging to class C_1 i.e. 'Corrosion'.

$P(C_2|(x_1, x_2, x_3))$ – Probability of the given instance (x_1, x_2, x_3) belonging to class C_2 i.e. 'Non-Corrosion'.

Figure 4.4. Schematic of a multi-layer perceptron configuration for the classification of the labeled data. Note that the input features x_1, x_2 and x_3 represent three color features associated with 'RGB', 'rgb', 'HSV' and 'La*b*' color spaces respectively.

4.5. Results

In this section, the best combination of the color space and an MLP configuration that can detect corrosion accurately under ambient lighting conditions is determined, and its efficacy in

the real-world scenario is demonstrated. Test images described in Section 4.3.4 are employed for the purpose of demonstrating the efficacy i.e., the corroded portions in the lab generated images and steel bridge images are detected and provided.

4.5.1. Determining the Best Combination of Color Space and MLP

Sixteen combinations of MLP configurations and color spaces are assessed for determining the best combination i.e. the capability of each combination to predict the class labels (corrosion/ non-corrosion) accurately is evaluated. All the combinations are first trained with the training dataset (see Section 4.3.2) and then simultaneously deployed to predict the class labels in the validation dataset. The predicted class labels are cross-validated with the actual known class labels in the validation dataset and the summary of correct and incorrect classifications are provided in the form of a confusion matrix ($\boldsymbol{C}$). The confusion matrix is a square matrix of size $m \times m$, where m (=2) represents the number of class labels and the elements C_{ij} represents the frequency of instances from the validation dataset that are assigned class label j by the classifier which in reality they belong to class label i. Given the confusion matrix $\boldsymbol{C}$, the performance of each combination is assessed by evaluating the four metrics namely 'Accuracy', 'Precision', 'Recall' and 'F Measure'. 'Accuracy' is defined as the ratio of the total number of instances whose class labels are correctly identified to the total number of instances present in the validation dataset and is expressed as [20, 41, 42].

$$A = \frac{\sum_{i=1}^{m} C_{ii}}{\sum_{i=1}^{m} \sum_{j=1}^{m} C_{ij}} \times 100\% \tag{4.7}$$

'Precision (O)' is defined as the ratio of the number of observations whose class label i is correctly predicted by the classifier to the total number of observations that are assigned to the class i by the classifier, and 'Recall (R)' is defined as the proportion of observations of class i that are correctly predicted as class i by the classifier [41].

$$O = \frac{1}{m}\sum_{i=1}^{m}\frac{C_{ii}}{\sum_{j=1}^{m} C_{ji}} \times 100\% \quad R = \frac{1}{m}\sum_{i=1}^{m}\frac{C_{ii}}{\sum_{j=1}^{m} C_{ij}} \times 100\% \tag{4.8}$$

Here, m represents the number of class labels. While overall precision and overall recall are also used as the measures of performance assessment for classifiers, F-measure (F) combines the trade-off between both overall precision and overall recall and is evaluated as

$$F = \frac{2 \times O \times R}{O + R} \times 100\% \tag{4.9}$$

Table 4.2. Confusion matrix of the results predicted by various MLP configurations.

Color space	**Confusion matrix**							
	2N		**4N**		**2N-2N**		**50N-10N-4N**	
RGB	0.330	0.671	0.350	0.651	0.360	0.641	0.610	0.391
Acc.	**66.50**		**67.50**		**68**		**80.50**	
rgb	0.790	0.211	0.810	0.191	0.810	0.191	0.820	0.180
	0	1	0	1	0	1	0.03	0.97
Acc.	**89.50**		**90.50**		**90.50**		**89.50**	
HSV	0.660	0.340	0.680	0.320	0.700	0.300	0.600	0.400
	0.38	0.62	0.28	0.72	0	1	0.04	0.96
Acc.	**64**		**70**		**85**		**78**	
La*b*	0.390	0.611	0.510	0.491	0.540	0.461	0.310	0.691
Acc.	**69.50**		**75.50**		**77**		**65.50**	

Note: **Acc.** – Accuracy; Confusion matrix $\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$ where $c_{11}\ and\ c_{22}$ represents the correct predictions corresponding to class labels corrosion and non-corrosion respectively, $c_{12}\ and\ c_{21}$ represents the incorrect predictions corresponding to class labels corrosion and non-corrosion respectively.

The confusion matrix and 'Accuracy' evaluated for each combination of MLP configuration and color space are summarized in Table 4.2. From Table 4.2, it can be inferred that the color spaces 'RGB' and 'CIE La*b*' resulted in a higher fraction of misclassification of 'corrosion' class label when compared to 'rgb' and 'HSV' color space i.e. on an average 65% of the instances in the training dataset belonging to 'corrosion' class are misclassified as 'non-

corrosion' in the case of 'RGB' and 'CIE La*b*' color space (see also Figure 4.5). However, it is interesting to note that in the case of 'RGB' color space the accuracy improved when an additional hidden layer is added to the two-layer MLP configuration (see also Figure 4.5). The improvement in the accuracy of the 'RGB' color space may be attributed to the increased non-linearity or complexity in the decision boundary resulting from the addition of a hidden layer [43]. In other words, the decision boundary of a three-layer MLP maybe partitioning the training instances such that more fraction of the instances belonging to 'corrosion' class are on the same side of the boundary. Among the four-color spaces, the 'rgb' color space is found to yield a maximum prediction accuracy (91%). Despite the increase in the number of hidden layers the accuracy for the 'rgb' color space did not vary significantly (see Figure 4.5). To understand why the 'rgb' color space resulted in higher accuracy even without the addition of hidden layers, the color feature data is plotted as 2D scatter plots after dimensional reduction is performed on the training and validation datasets.

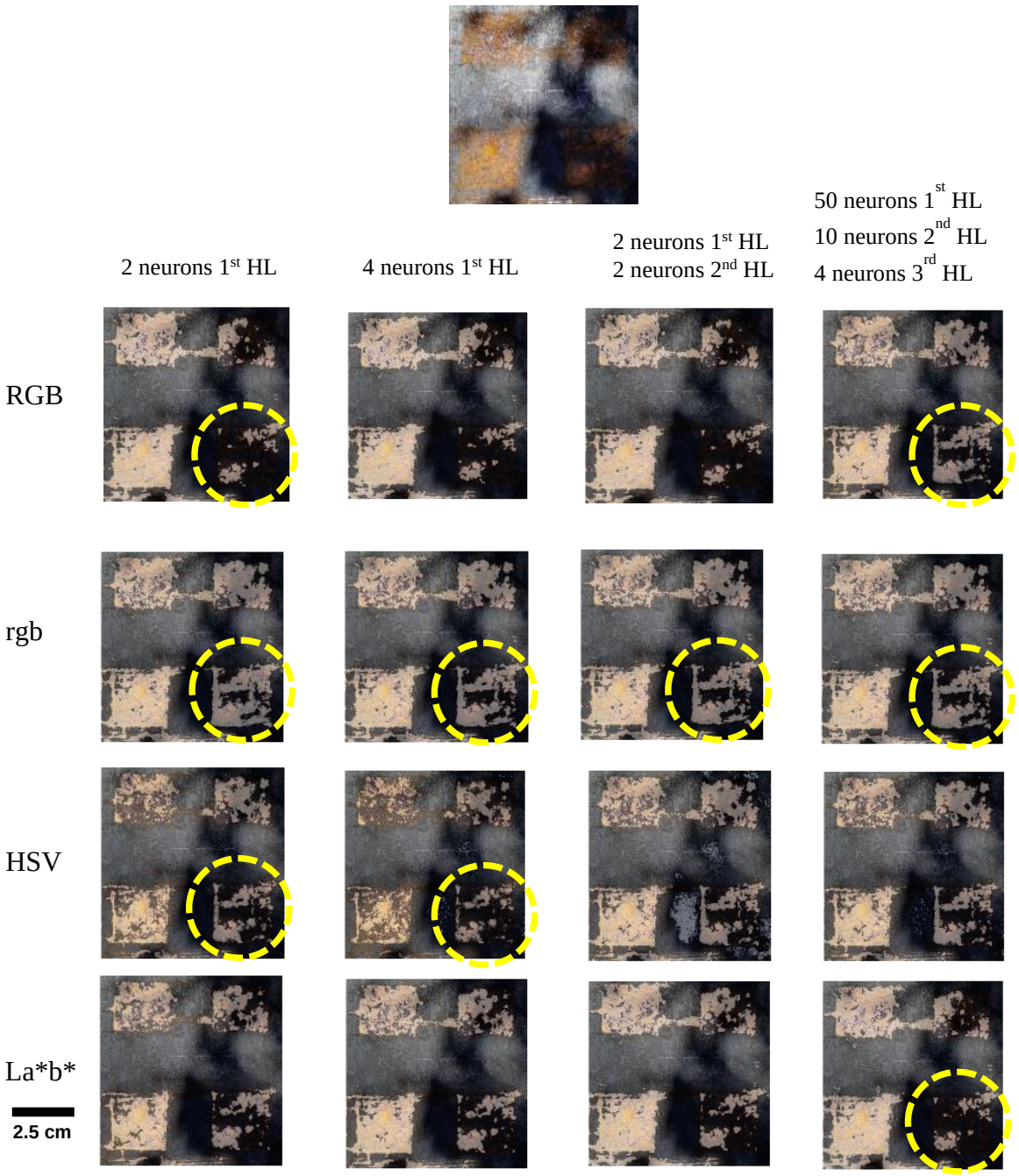


Figure 4.5. Prediction of corrosion in a test image (with shadow) using four different color spaces and four different ANN configurations. Markers for 'RGB' – accuracy improved for three-layer MLP; Markers for 'rgb' – accuracy remained almost same; Markers for 'HSV' – accuracy is poor for 2 and 4 neurons MLP configuration; Markers for 'CIE La*b*' – accuracy is poor for three-layer MLP configuration. *Note* HL – hidden layer.

In the context of the current study, the dimensional reduction will facilitate visualizing a three-dimensional data (color features) in a two-dimensional space. Visualizing data in two-

dimensional space will not only reveal the spatial distribution of instances belonging to ‘corrosion’ and ‘non-corrosion’ class but also aids in understanding the decision boundaries that can partition the instances with different class labels. Linear discriminant analysis (LDA) technique is used to perform dimensional reduction [44, 45]. The results obtained after dimensional reduction for all four-color spaces are shown in Figure 4.6. Figure 4.6 consists of a scatterplot of four isolated groups that are labeled as ‘Tr_Corrosion’, ‘Tr_Non-corrosion’, ‘Sh_Corrosion’ and ‘Sh_Non-corrosion’. Note that the groups labeled as ‘Tr_Corrosion’ and ‘Tr_Non-corrosion’ correspond to the instances with class labels ‘corrosion’ and ‘non-corrosion’, respectively that are obtained from the training dataset and the groups ‘Sh_Corrosion’ and ‘Sh_Non-corrosion’ correspond to the instances with class labels ‘corrosion’ and ‘non-corrosion’, respectively obtained from the validation dataset. A trained MLP is anticipated to predict the class labels correctly when the instances of the ‘Tr_Corrosion’ and ‘Sh_Corrosion’ group remain on the same side of the decision boundary. In other words, the color features of the corroded pixels obtained from shaded regions (validation dataset) and varying illuminations (training dataset) should be on the same side of the decision boundary. In the case of ‘rgb’ color space, it may be true that the higher number of instances from ‘Tr_Corrosion’ and ‘Sh_Corrosion’ always remained on the same side of decision boundary irrespective of change in the shape of the boundary. For the sake of illustrating this point, the pseudo decision boundaries that may be resulting from two different MLP configurations are plotted in Figure 4.6 (b). From Figure 4.6 (b) it can be understood that the instances partitioned by both the pseudo decision boundaries remains more or less similar.

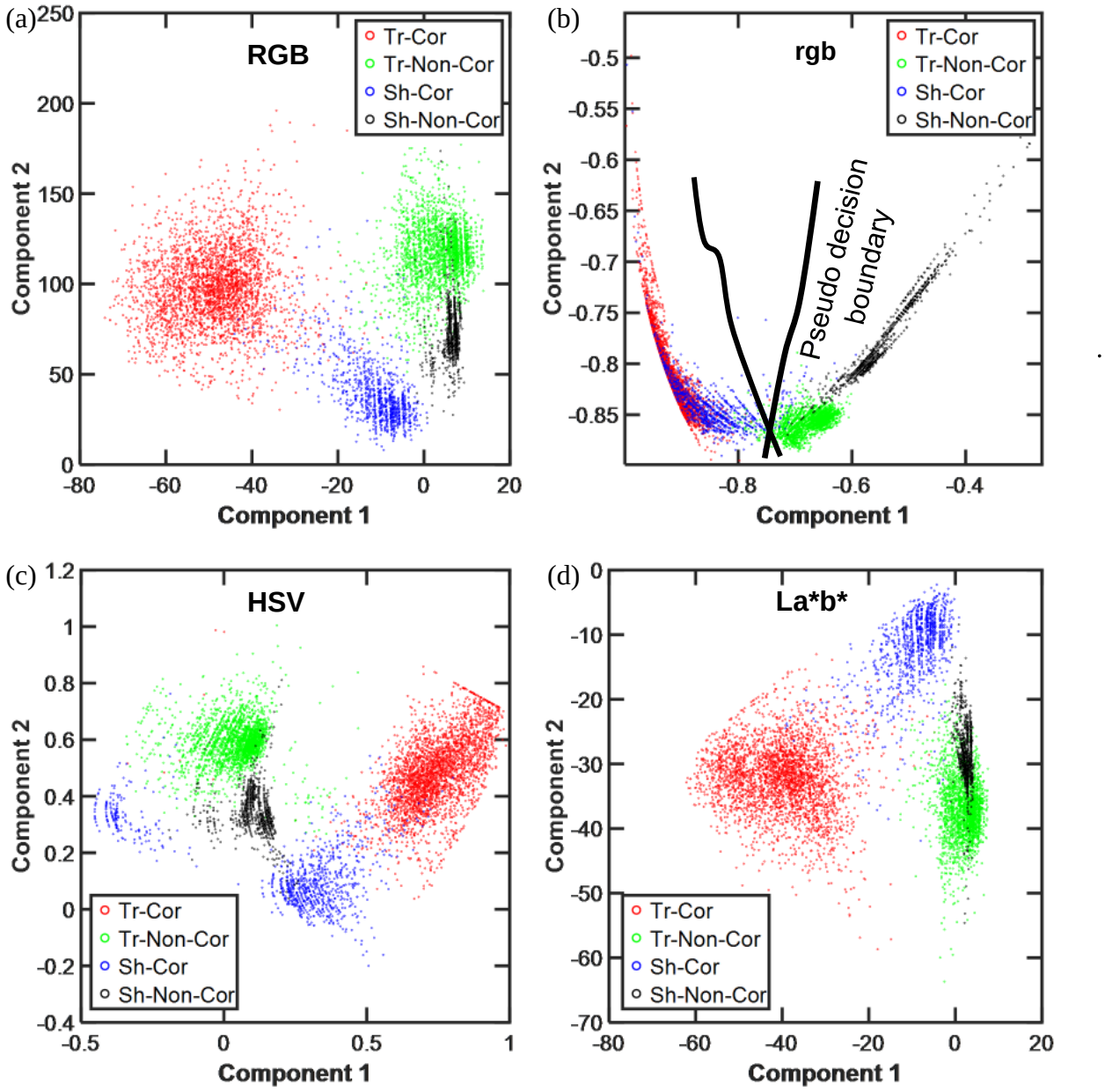


Figure 4.6. Dimensional reduction using LDA. Training dataset encompassing 4 class labels namely corrosion (Tr-Cor), non-corrosion (Tr-Non-Cor), corrosion in shadow (Sh-Cor) and non-corrosion (Sh-Non-Cor) in shadow are visualized in a 2-dimensional space. (a) RGB, (b) rgb, (c) HSV and (d) La*b*.

Besides 'Accuracy', the other performance metrics 'Precision', 'Recall' and 'F-measure' are also evaluated and provided in Table 4.3. For most of the combinations of MLP and color spaces the 'Precision' value is 100%. This can be attributed to the zero false positive value of the

‘Corrosion’ class label i.e. no ‘Non-corrosion’ class labels are incorrectly predicted as ‘Corrosion’. However, the ‘Recall’ and ‘F-measure’ value varied for different combinations. Note that the ‘Recall’ with a higher magnitude is preferred instead of ‘Accuracy’ for choosing the best combination of MLP and color space. ‘Recall’ measures the ability of a model to predict the actual ‘Corrosion’ class label as ‘Corrosion’ which is highly desired. Based on the assessment of ‘Recall’ values for all the combinations i.e., a single hidden layer with 4 neurons (1st HL (4N)) MLP configuration with ‘rgb’ color space is chosen as the best combination in this study.

Table 4.3. Performance metrics of various MLP configurations.

Color space		**Performance Metrics (%)**			
		Accuracy	**Recall**	**Precision**	**F-Measure**
‘RGB’	2N	66.5	33	100	49.62
	4N	67.5	35	100	51.85
	2N-2N	68	36	100	52.94
	50N-10N-4N	80.5	61	100	75.78
‘rgb’	2N	89.5	79	100	88.27
	4N	90.5	81	100	89.50
	2N-2N	90.5	81	100	89.50
	50N-10N-4N	89.5	81	100	90.11
‘HSV’	2N	64	66	63	64.47
	4N	70	68	70	68.99
	2N-2N	85	70	100	82.35
	50N-10N-4N	78	60	100	75.00
‘La*b*’	2N	69.5	39	100	56.12
	4N	75.5	51	100	67.55
	2N-2N	77	54	100	70.13
	50N-10N-4N	65.5	31	100	47.33

4.5.2. Detection of Corrosion in Lab Generated Test Images

An MLP configuration consisting of single hidden layer with 4 neurons (1st HL (4N)) trained on ‘rgb’ color features and is deployed to detect corroded portions in the lab generated test images and the results obtained are shown in Figure 4.7 to Figure 4.10. Note the corroded

portions are represented as a bright mask in the Figure 4.7(b), Figure 4.8(b), Figure 4.9(b) and Figure 4.10 (b). For qualitative comparison, the ground truth images are also provided along with the corrosion detected images (see Figure 4.7(a), Figure 4.8(a), Figure 4.9(a) and Figure 4.10(a)). While Figure 4.7 and Figure 4.8 consists of test images acquired under varying illuminations and cast shadows, respectively, Figure 4.9 and Figure 4.10 consists of test images acquired under water and oil wetting, respectively. From Figure 4.7 to Figure 4.10, it is evident that the trained MLP detects corrosion accurately in the test images that are acquired under varying illuminations, shadows, water wetting, and oil wetting conditions. Despite the presence of light shadows and dark shadows found in Figure 4.8 and the wetting conditions found in Figure 4.9 and Figure 4.10, the single hidden with 4 neurons (4N) MLP trained with 'rgb' color features predicted corrosion accurately. However, in the case of water and oil wetted test images some specular reflections are observed and it is highly unlikely to detect corrosion under such a scenario. Specular reflections are the mirror-like reflections commonly observed on smooth surfaces of bodies where the angle of incidence of light is equal to the angle of reflection [46].

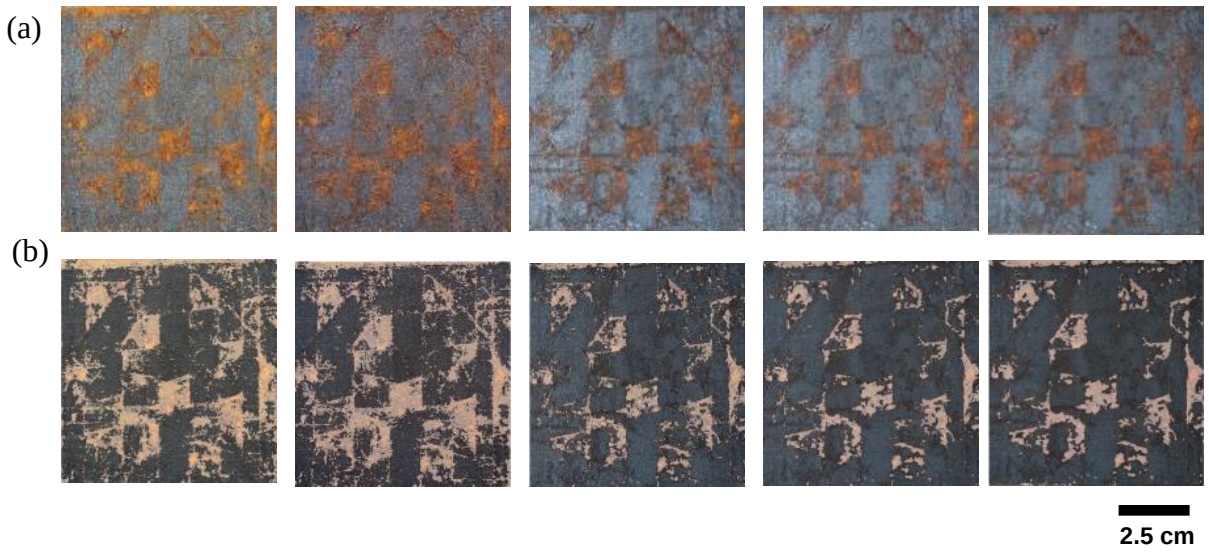


Figure 4.7. Test images of partially corroded steel plates acquired at different illuminations of natural daylight. (a) ground truth images, (b) MLP-based corrosion prediction.

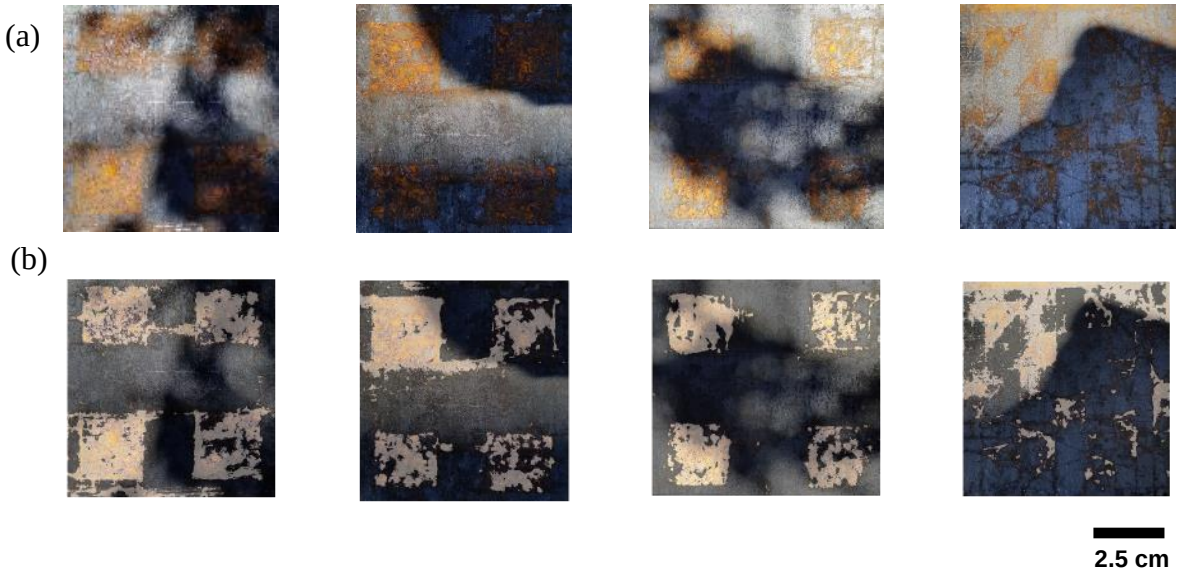


Figure 4.8. Test images of partially corroded steel plates with shadows cast in natural daylight. (a) ground truth images, (b) MLP-based corrosion prediction.

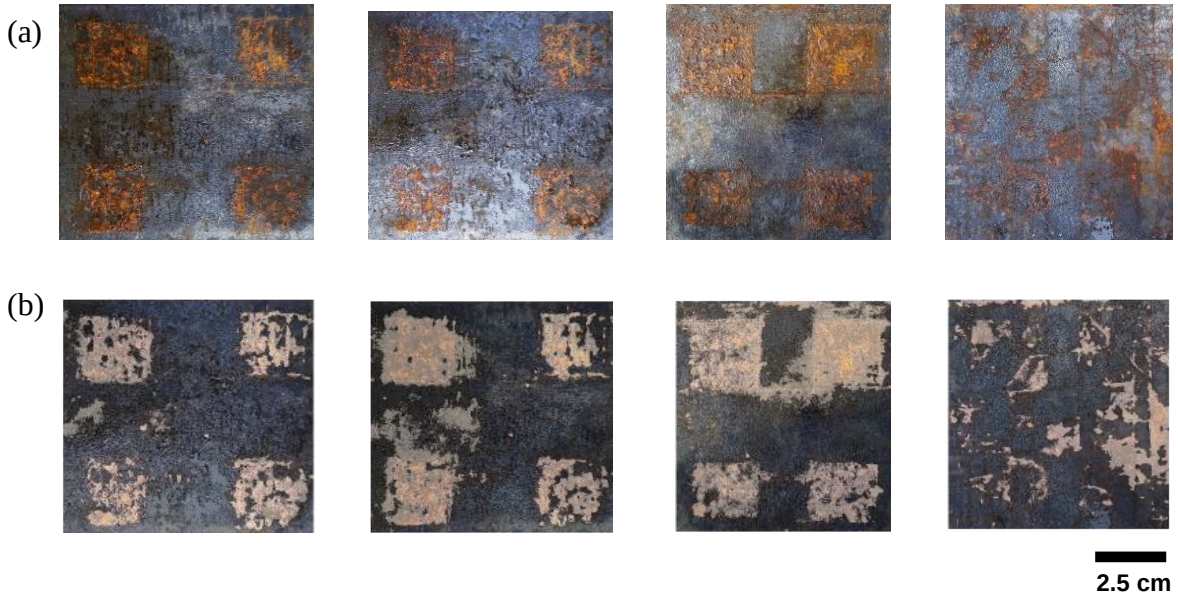


Figure 4.9. Test images of partially corroded steel plates wetted in water and acquired in natural daylight. (a) ground truth images, (b) MLP-based corrosion prediction.

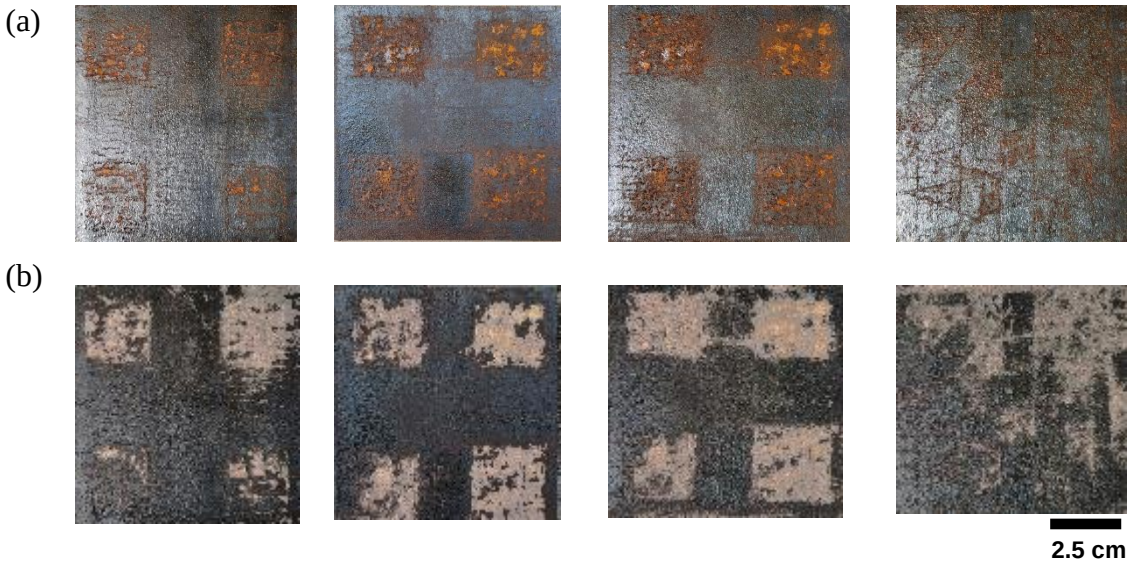


Figure 4.10. Test images of partially corroded steel plates wetted in oil and acquired in natural daylight. (a) ground truth images, (b) MLP-based corrosion prediction.

4.5.3. Detection of Corrosion in Steel Bridge

An MLP configuration consisting of single hidden layer with 4 neurons (1st HL (4N)) trained on 'rgb' color features is finally deployed to detect corroded portions in the steel bridge images and the results obtained are shown in Figure 4.11. Similar to Figure 4.7-Figure 4.10, the corroded portions of the bridge are highlighted as a bright mask in Figure 4.11 and the ground truth images are provided for a qualitative comparison. From Figure 4.11, it can be observed that the trained MLP detects corrosion accurately in the steel girder bridges under naturally varying illuminations and self-shadows. While 'Marker 1' and 'Marker 2' shown in Figure 4.11 (a) reveals the ability of the trained MLP to detect corrosion under brighter illumination, 'Marker 3' indicates the ability of the trained MLP to detect corrosion under comparatively darker illuminations and self-shadows. Further from Figure 4.11(b) it is also evident that the trained MLP configuration of single hidden layer with 4 neurons (1st HL (4N)) is able to predict corrosion in the bottom side of the deck of the bridge that has a dark shadow. At this juncture, it is important to emphasize the fact that it is highly unlikely for a human vision to detect corrosion

under dark shadows. Keeping in view that a greater portion of corrosion is found in the bottom side of the deck of the bridge, the proposed method will be very useful.

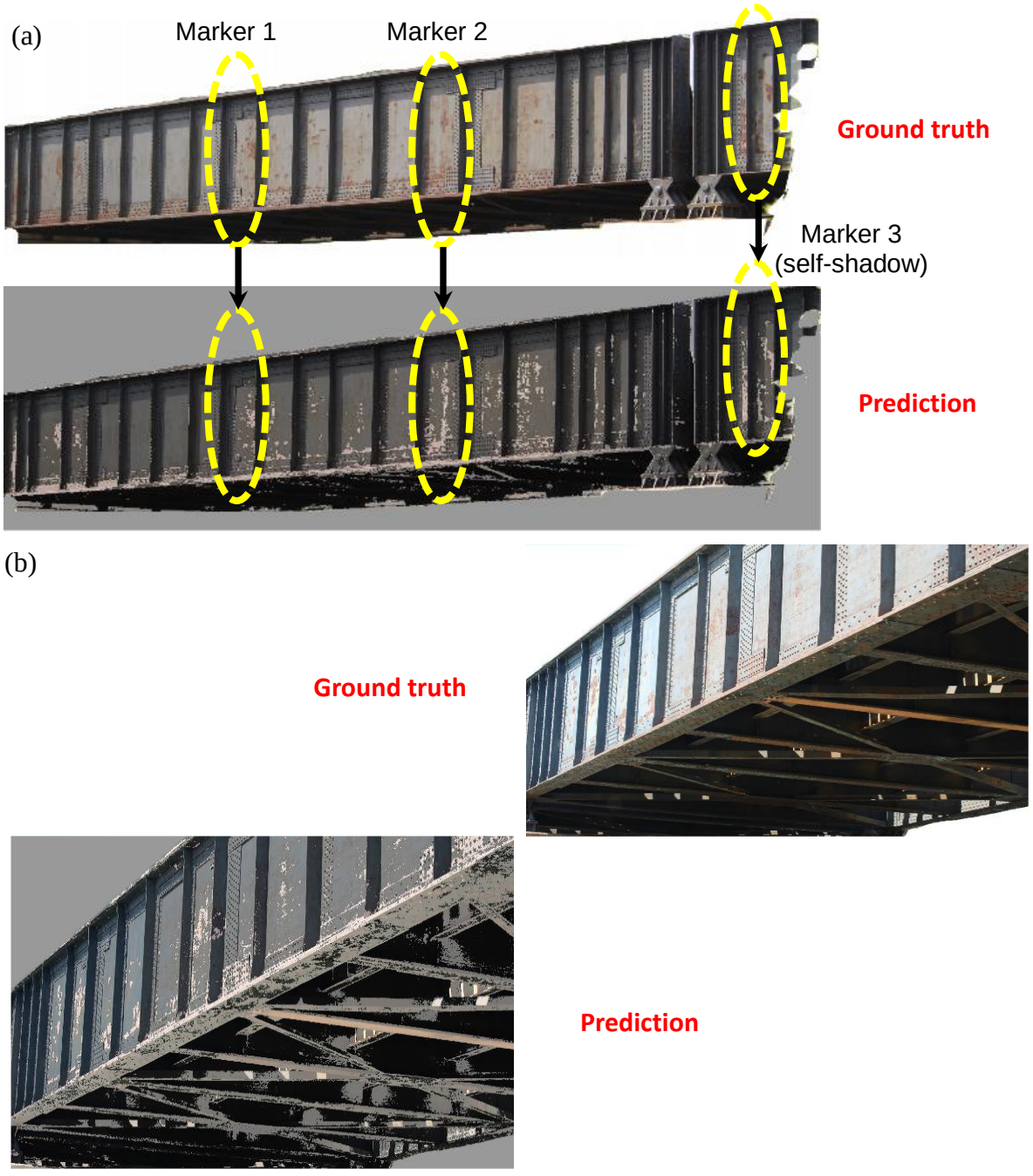


Figure 4.11. Identification of corrosion in the steel bridges using the single hidden layer with 4 neurons (1st HL(4N)) MLP Configuration. (a) steel plate girders with naturally varying illumination and self-shadows, (b) bottom side of the deck of the bridge with dark self-shadows.

4.6. Conclusions and Limitations

Color spaces in conjunction with different MLP configurations are explored to detect corrosion initiation in steel structures under ambient lighting conditions. To this end, sixteen different combinations of color spaces and MLP configurations are explored. The performance of each combination is then assessed through the validation dataset obtained from lab generated images and the best combination is determined. Subsequently, the obtained combination is deployed on the test image database and the efficacy of trained MLP to detect corrosion in real-world scenarios is demonstrated.

From the current study following conclusions can be drawn.

1. Among all sixteen combinations of color space and an MLP configuration, the combination of ‘rgb’ color space and an MLP configuration of a single hidden layer with 4 neurons (1^{st} HL (4N)) yielded the highest ‘Recall’ of 81% and hence chosen as the best combination.
2. While the accuracy (up to 91%) of ‘rgb’ color space is found to be more or less similar for all the MLP configurations, the accuracy of ‘RGB’ color space is observed to increase from 68% to 81% with the addition of third hidden layer. Improved accuracy in the case of ‘RGB’ color space can be attributed to the increased non-linearity of the decision boundary generated by the MLP which will ultimately lead to overfitting issues.
3. Under shadows and wetting conditions, the trained MLP is still found to yield correct predictions when ‘rgb’ color features are used. Especially, the detection of corrosion in the bottom side of the deck of a bridge under dark shadows is noteworthy.

4. The proposed method is insensitive to the camera sensor employed for the image acquisition i.e., irrespective of images being acquired from a mobile camera or a DSLR camera the efficacy of trained MLP to detect corrosion was not affected.
5. MLP trained on varying illumination dataset alone is sufficient for detecting the corrosion under shadows and wetting conditions.

Although the efficacy of color spaces for corrosion detection is demonstrated in this study, it is important to note that the employment of color features alone may have some limitations. One case where this technique is not applicable is when the objects in the acquired images possess hue values similar to that of a corroded surface. For instance, coatings, dirt, or some vegetation in the background may be misclassified as corrosion. This limitation will be addressed in Chapter 5.

4.7. References

1. Troitsky MS. Planning and design of bridges: John Wiley & Sons; 1994.
2. Haas T. Are Reinforced Concrete Girder Bridges More Economical Than Structural Steel Girder Bridges?: A South African Perspective. Jordan Journal of Civil Engineering. 2014;159:1-15.
3. Sastri VS. Challenges in corrosion: costs, causes, consequences, and control: John Wiley & Sons; 2015.
4. Chen W-F, Duan L. Bridge engineering handbook: construction and maintenance: CRC press; 2014.
5. García-Martín J, Gómez-Gil J, Vázquez-Sánchez E. Non-destructive techniques based on eddy current testing. Sensors. 2011;11:2525-65.

6. Pavlopoulou S, Staszewski W, Soutis C. Evaluation of instantaneous characteristics of guided ultrasonic waves for structural quality and health monitoring. Structural Control and Health Monitoring. 2013;20:937-55.

7. Sharma S, Mukherjee A. Ultrasonic guided waves for monitoring corrosion in submerged plates. Structural Control and Health Monitoring. 2015;22:19-35.

8. Nowak M, Lyasota I, Baran I. The Test of Railway Steel Bridge with Defects Using Acoustic Emission Method. Journal of Acoustic Emission. 2016;33.

9. Cole P, Watson J. Acoustic emission for corrosion detection. Advanced Materials Research: Trans Tech Publ; 2006. 231-6.

10. Deraemaeker A, Reynders E, De Roeck G, Kullaa J. Vibration-based structural health monitoring using output-only measurements under changing environment. Mechanical systems and signal processing. 2008;22:34-56.

11. McCrea A, Chamberlain D, Navon R. Automated inspection and restoration of steel bridges—a critical review of methods and enabling technologies. Automation in Construction. 2002;11:351-73.

12. Doshvarpassand S, Wu C, Wang X. An overview of corrosion defect characterization using active infrared thermography. Infrared Physics & Technology. 2018.

13. Jahanshahi MR, Kelly JS, Masri SF, Sukhatme GS. A survey and evaluation of promising approaches for automatic image-based defect detection of bridge structures. Structure and Infrastructure Engineering. 2009;5:455-86.

14. Chen P-H, Chang L-M. Artificial intelligence application to bridge painting assessment. Automation in construction. 2003;12:431-45.

15. Chen P-H, Yang Y-C, Chang L-M. Automated bridge coating defect recognition using adaptive ellipse approach. Automation in Construction. 2009;18:632-43.
16. Shen H-K, Chen P-H, Chang L-M. Automated steel bridge coating rust defect recognition method based on color and texture feature. Automation in Construction. 2013;31:338-56.
17. Gevers T, Gijsenij A, Van de Weijer J, Geusebroek J-M. Color in computer vision: fundamentals and applications: John Wiley & Sons; 2012.
18. Koschan A, Abidi M. Digital color image processing: John Wiley & Sons; 2008.
19. Zhang Z, Flores P, Igathinathane C, Naik DL, Kiran R, Ransom JK. Wheat Lodging Detection from UAS Imagery Using Machine Learning Algorithms. Remote Sensing. 2020;12:1838.
20. Naik DL, Kiran R. Identification and characterization of fracture in metals using machine learning based texture recognition algorithms. Engineering Fracture Mechanics. 2019;219:106618.
21. Medeiros FN, Ramalho GL, Bento MP, Medeiros LC. On the evaluation of texture and color features for nondestructive corrosion detection. EURASIP Journal on Advances in Signal Processing. 2010;2010:817473.
22. Ranjan R, Gulati T. Condition assessment of metallic objects using edge detection. Int J Adv Res Comput Sci Softw Eng. 2014;4:253-8.
23. Lee S, Chang L-M, Skibniewski M. Automated recognition of surface defects using digital color image processing. Automation in Construction. 2006;15:540-9.
24. Chen P-H, Shen H-K, Lei C-Y, Chang L-M. Support-vector-machine-based method for automated steel bridge rust assessment. Automation in Construction. 2012;23:9-19.

25. Ghanta S, Karp T, Lee S. Wavelet domain detection of rust in steel bridge images. 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP): IEEE; 2011;1033-6.

26. Nelson BN, Slebodnick P, Lemieux EJ, Singleton W, Krupa M, Lucas K, et al. Wavelet processing for image denoising and edge detection in automatic corrosion detection algorithms used in shipboard ballast tank video inspection systems. Wavelet Applications VIII: International Society for Optics and Photonics; 2001. 134-45.

27. Son H, Hwang N, Kim C, Kim C. Rapid and automated determination of rusted surface areas of a steel bridge for robotic maintenance systems. Automation in Construction. 2014;42:13-24.

28. Liao K-W, Lee Y-T. Detection of rust defects on steel bridge coatings via digital image recognition. Automation in Construction. 2016;71:294-306.

29. Sajid HU, Naik DL, Kiran R. Microstructure–Mechanical Property Relationships for Post-Fire Structural Steels. Journal of Materials in Civil Engineering. 2020;32.

30. Sajid HU, Kiran R. Influence of corrosion and surface roughness on wettability of ASTM A36 steels. Journal of Constructional Steel Research. 2018;144:310-26.

31. Delgado-González M, Carmona-Jiménez Y, Rodríguez-Dodero M, García-Moreno M. Color Space Mathematical Modeling Using Microsoft Excel. ACS Publications; 2018.

32. Garcia-Lamont F, Cervantes J, López A, Rodriguez L. Segmentation of images by color features: A survey. Neurocomputing. 2018;292:1-27.

33. Smith AR. Color gamut transform pairs. ACM Siggraph Computer Graphics. 1978;12:12-9.

34. Liu G-H, Yang J-Y. Exploiting color volume and color difference for salient region detection. IEEE Transactions on Image Processing. 2018;28:6-16.
35. Shanmuganathan S. Artificial neural network modelling: An introduction. Artificial Neural Network Modelling: Springer; 2016. 1-14.
36. Priddy KL, Keller PE. Artificial neural networks: an introduction: SPIE press; 2005.
37. Aggarwal CC. Neural networks and deep learning. Cham: Springer International Publishing. 2018.
38. Daniel G. Principles of artificial neural networks: World Scientific; 2013.
39. Ogliari E, Leva S. Computational Intelligence in Photovoltaic Systems: MDPI; 2019.
40. Engel T, Gasteiger J. Chemoinformatics: basic concepts and methods: John Wiley & Sons; 2018.
41. Naik DL, Sajid HU, Kiran R. Texture-Based Metallurgical Phase Identification in Structural Steels: A Supervised Machine Learning Approach. Metals. 2019;9:546.
42. Naik DL, Kiran R. Naïve Bayes classifier, multivariate linear regression and experimental testing for classification and characterization of wheat straw based on mechanical properties. Industrial Crops and Products. 2018;112:434-48.
43. Sethi IK, Jain AK. Artificial neural networks and statistical pattern recognition: old and new connections: Elsevier; 2014.
44. Li C, Wang B. Fisher linear discriminant analysis. August, 31. 2014.
45. Gu Q, Li Z, Han J. Linear discriminant dimensionality reduction. Joint European conference on machine learning and knowledge discovery in databases: Springer; 2011. 549-64.

46. Tan K, Cheng X. Specular reflection effects elimination in terrestrial laser scanning intensity data using Phong model. Remote Sensing. 2017;9:853.

47. Lee H-C. Introduction to color imaging science: Cambridge University Press; 2005.

48. Mestha LK, Dianat SA. Control of color imaging systems: analysis and design: CRC Press; 2009.

5. HYPERSPECTRAL IMAGING FOR THE ELIMINATION OF VISUAL AMBIGUITY IN CORROSION DETECTION AND IDENTIFICATION OF CORROSION SOURCES[4]

5.1. Introduction

In recent years, various image processing techniques have been developed for the identification of corrosion in different sectors of infrastructure such as bridges, oil, and gas refinery, power plants, underground pipelines, etc. (see Figure 1.8) [1–3]. In this technique, the digital images of the steel surface are first acquired on-site and are then analyzed using image processing methods off-site. While some of the available approaches in the literature are limited to grayscale images [4–8], the other approaches employ color images [9–11]. Image features such as edges, texture [12], pixel intensity, hue, etc., were considered in these approaches in conjunction with machine learning (ML) algorithms [13–17] for the detection of corrosion. In a recent study, the authors have explored various color spaces in conjunction with a multi-layer perceptron to address the misclassification associated with varying illumination of sunlight, dark shadows, water wetting, and oil wetting [18]. Nevertheless, all the available approaches still lack the ability to distinguish a corroded surface from a similar hue possessing object in the image, i.e., the object which has the hue similar to that of the corroded surface may always be misidentified as a corroded surface in a given image. For instance, coatings, brick walls, dirt, or some vegetation is often noticed in the background of a structure when the images are acquired (see Figure 1.7) and can be misclassified as corrosion. This optical confusion between the

[4] This chapter is based on the paper "Hyperspectral Imaging for the Elimination of Visual Ambiguity in Corrosion Detection and Identification of Corrosion Sources". *Structural Health Monitoring*. The material in this chapter was co-authored by Dayakar Naik Lavadiya (DNL), Ravi Kiran Yellavajjala (RK) and Hizb Ullah Sajid (HUS). Contributions of authors are as follows: Conceptualization, D.L.N. and R.K.; Formal analysis, D.L.N.; Funding acquisition, R.K.; Investigation, R.K.; Methodology, D.L.N., and R.K.; Project administration, R.K.; Resources, H.U.S. and R.K.; Software, D.L.N.; Supervision, R.K.; Validation, D.L.N.; Writing—original draft, D.L.N.; Writing—review & editing, D.L.N. and R.K.

corroded surfaces and other objects with similar hue is referred to as visual ambiguity in the rest of this paper.

In addition to addressing the visual ambiguity, it is also important to distinguish the corroded surfaces that are chemically distinct and then use this information to identify the source of corrosion, i.e., the corrosive media. Identifying the source of corrosion will aid in deploying an appropriate corrosion mitigation strategy [19–21]. Note that the chemical distinctiveness of corroded surface is often expressed qualitatively in terms of corrosion products, which is also known to govern the corrosion rate [22,23]. Corrosion products are the minerals of iron oxide and oxyhydroxides resulting from the reaction between the steel metal substrate and the corrosive media [24]. While an acidic media such as hydrochloric acid (HCl) is found to produce Magnetite (Fe_3O_4), Ferrous Chloride ($FeCl_2$), and FeOOH as corrosion products in steel, the salts such as NaCl and Na_2SO_4 are found to produce Akageneite (β-FeOOH) [25] and Goethite (α-FeOOH) [25], respectively as corrosion products in steel. The redox reactions of steel and all three different corrosive media (HCl, NaCl and Na_2SO_4) are mentioned elsewhere [26–31]. As reported in the literature, HCl exhibits a corrosion rate of 49.96 mm/year [32] which is more aggressive when compared to the corrosion rate exhibited by the salts NaCl and Na_2SO_4 i.e., 2.45 mm/year [33] and 1.27 mm/year [34], respectively. Note that Goethite was found to decelerate the corrosion rate [35], and Akageneite was found to increase the corrosion rate [36].

The shortcomings of the existing image processing techniques in the literature can be attributed to the limited spectral information acquired in the images, i.e., broadband spectra of Red, Green, and Blue color (RGB). The current study is based on the premise that the acquisition of multiple narrow-band spectra of the corroded surface in the visible and near infra-red (VNIR) region of Electromagnetic (EM) radiation will aid in providing more information that can

distinguish corrosion from visually ambiguous objects and identify the chemically distinct corroded surface. Hyperspectral imaging (see Figure 5.1) is an emerging technology that integrates conventional imaging and spectroscopy to simultaneously collect spatial and spectral information from the surface of an object [37–39]. The images obtained from hyperspectral imaging sensor are referred to as data cubes since they are composed of spatially arranged pixels in a two-dimensional space and a set of spectral reflectance intensities (for each pixel) in the third dimension. In comparison with conventional photos, which have a few spectral broadbands, for example, RGB and multispectral image (MSI), the hyperspectral images have several hundreds of narrow spectral bands of the same scene [40]. While the use of hyperspectral imaging technique for object recognition and characterization in the fields of pharmaceuticals [41,42], agriculture [43–45], food quality control [46–49], material identification, and mapping of the artworks [50–52] is evident from the literature, its importance in the field of structural health monitoring, especially corrosion damage detection in civil infrastructure is found to be very limited to the best of our knowledge [53].

This chapter aims to identify the chemically distinctive and visually ambiguous corroded surface (i.e., corrosive media) using the support vector machine (SVM) classifier. For training the SVM classifier the reflectance magnitude of the VNIR spectral bands of corroded/coated surface are employed as the descriptive features. Furthermore, the key wavelengths of the VNIR spectra that capture the distinguishability of visually ambiguous corroded surfaces are also determined, which may aid in building a customized multispectral camera for on-field applications [54]. The rest of the paper is organized as follows: protocol for the acquisition of hyperspectral data is described in Section 5.2, generating training, test, and validation datasets are explained in Section 5.3, the methodology adopted for this study is detailed in Section 5.4,

results are discussed in Section 5.5, and the conclusions and applications are provided in Section 5.6.

5.2. Acquisition of Hyperspectral Data

Hyperspectral data of the corroded (HCl, NaCl, Na_2SO_4), non-corroded, and paint coated ASTM A572 structural steel plates are acquired in this study to train an SVM classifier. In this section, the process of inducing corrosion onto the ASTM A572 structural steel plates is described, and the procedure followed for acquiring the spectral data using a hyperspectral imaging system (HIS) is provided.

5.2.1. Lab Generated Coated and Corroded Steel Plates

ASTM A572 plates (3"×3"×5/16") are acquired from a commercial supplier and are coated with the acrylic/vinyl paint that has R, G, and B values 124, 0, and 32, respectively. The paint material chosen here has the visual appearance of the protective coatings used in real-world scenarios. Furthermore, the choice of RGB color ensures that the visual ambiguity with the corroded surface is achieved (see Figure 1.7(c)) for fulfilling the objective of the study. The chemical composition of the ASTM A572 plates as supplied by the manufacturer is provided in Table 5.1. A total of five steel plates are coated in this study. While two plates are completely surface coated, the rest of the steel plates are only partially coated (see Figure 5.2 and Figure 5.3). Completely coated plates are used to generate the training and test datasets (see Section 5.3.1) consisting of spectral information extracted from numerous pixels randomly chosen from the hyperspectral image and, partially coated plates are used to create the validation dataset (see Section 5.3.2) that consist of spectral information extracted from each coated pixel of the hyperspectral image.

Table 5.1. Chemical composition of ASTM A572 structural steel.

Chemical composition (%)	ASTM A572 Gr. 50
Carbon (C)	0.05
Manganese (Mn)	1.34
Phosphorous (P)	0.011
Sulphur (S)	0.004
Silicon (Si)	0.15
Copper (Cu)	0.28
Chromium (Cr)	0.19
Nickle (Ni)	0.13
Molybdenum (Mo)	0.04
Vanadium (V)	0.083
Titanium (Ti)	0.001
Niobium (Nb)	0.003
Iron (Fe)	97.718

For corroding ASTM A572 plates, three different corrosive media, namely 'Acid' – 1M hydrochloric acid (HCl), 'Salt' – 3.5 wt.% sodium chloride (NaCl), and 'Sulfate' – 3 wt.% sodium sulfate (Na_2SO_4) are employed. The purpose of using three different corrosive media is to achieve chemically distinctive corroded ASTM A572 plate surfaces, i.e., different corrosion products (see Section 5.1). For achieving the NaCl corrosion and Na_2SO_4 corrosion the steel plates are placed inside an environmental chamber that is operated at a constant temperature of 30°C and a relative humidity of 60%. In the case of NaCl corrosion, 3.5 wt.% NaCl solution is periodically sprayed (every ~7 hours) on the steel plate surfaces for three days. Similarly, in the case of Na_2SO_4 corrosion, 3 wt.% Na_2SO_4 solution is periodically sprayed (every 7 hours) on the surfaces of the steel plates for three days. For HCl corrosion, steel plates are kept immersed in a 1M HCl solution for 1 hr. and are then removed and placed in a humid environment for three days to achieve complete corrosion in the immersed areas of the steel plates. Note that the above-mentioned procedure is not a standard protocol and is only followed for the sake of achieving visually identifiable corrosion on the steel surfaces. One set of four plates each are corroded using each of the three corrosive media. While the surfaces of three of the plates in each set are exposed completely to a corrosive media, one of the four plates in each set, specifically, the

partially paint coated plate, is exposed partially to the corrosive media. In the process of corroding the partially paint coated plate, the painted region and some portion of the non-corroded region are masked with the acid-resistant tape such that only two small square slots of dimensions 0.5×0.5 inch are exposed for corrosion (see Figure 5.3).

5.2.2. Data Acquisition and Calibration

Specim FX10® (Spectral Imaging Ltd., Oulu, Finland) benchtop hyperspectral imaging system (HIS) is employed to acquire the hyperspectral data of corroded (#4 HCl, #4 NaCl, #4 Na_2SO_4), non-corroded (#2) and coated plates (#2). The schematic of the HIS used in this study is shown in Figure 5.1, which consists of a two-column frame that holds the hyperspectral camera and the illumination source on the top and a movable scanning platform perpendicular to the camera lens on the bottom. The hyperspectral camera employed herein is capable of capturing 448 narrow spectral bands of the reflected light with wavelengths ranging from 397 nm - 1004 nm (a.k.a Visible and Near Infra-Red (VNIR) spectra). Two 150-Watt halogen lamps (provided by the manufacturer) are employed as an illumination source. Halogen lamps are a source of incandescent light that emits energy in the VNIR range of the electromagnetic spectrum [55]. For obtaining the hyperspectral data, the plates are placed on the scanning platform that is allowed to move in a horizontal direction at a speed of 13.7 mm/s. The vertical distance between the camera lens and the scanning platform is maintained at 30 cm, and the exposure time of the camera is set to 29.22 ms. Note that the image acquisition of the surface of the object is carried out in a strip-by-strip fashion as the scanning platform moves (scanning principle) i.e., the spectral information along the line strip of the object is acquired one-at-a-time and not the entire surface (see Figure 5.1). This type of image acquisition is referred to as a push-broom or line scan technique. The spatial sampling comprises 1024 pixels in a one-line strip. The

data acquired from scanning is then stored in a raw data format by the Lumo® Recorder software interface connected to the HIS.

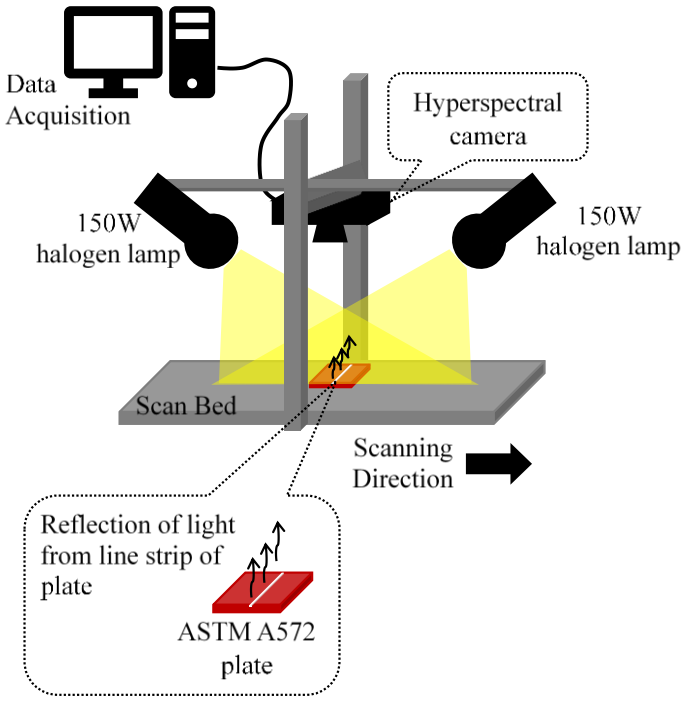


Figure 5.1. Schematic of the line-scan/push-broom Hyperspectral Imaging System (HIS).

The raw spectral data in the form of a data cube (the third dimension of the cube has the spectral reflectances) obtained from the HIS consists of noise from (1) the light reflected by the instrument and the illumination in the background and (2) the 'dark current' inherently generated by the sensor itself when no light/photon is incident on the sensor (i.e., when the camera shutters are closed). Dark current is a thermal phenomenon in which electrons are generated spontaneously within the sensor when they are thermally excited, and this phenomenon does not depend on the intensity of light reflected from the object [55]. The raw spectral data is calibrated beforehand using the white and dark reference spectra that are acquired along with the spectra of the plate specimens to eliminate the above-mentioned noises. The white reference spectra are obtained by scanning a white reference tile made from Spectralon whose spectral reflectance

ranges from 250 nm to 2500 nm [55], and the dark reference spectra were captured by fully obscuring the camera objective lens using an opaque black cap. The calibrated data are obtained using

$$I = \frac{I_o - I_d}{I_w - I_d} \tag{5.1}$$

where, I is the magnitude of calibrated spectral reflectance, I_o is the magnitude of raw spectral reflectance, I_d is the magnitude of dark spectral reference and I_w is the magnitude of white spectral reference. In the current study, Prediktera® Evince software is employed to calibrate the data cube.

5.3. Datasets

The calibrated spectral reflectance data of the corroded (HCl, NaCl, Na_2SO_4), non-corroded, and coated pixels are extracted from the spatial dimensions of the hyperspectral cube of corroded, non-corroded, and paint coated steel plates, respectively, and a master dataset is generated. A master dataset D herein is referred to as a matrix of N rows, and $r = q + 1$ columns such that each row represents the spectral information of the corroded, non-corroded, or paint coated pixel; each of the q columns (descriptive features) represents the reflectance magnitude of VNIR spectral wavelength or spectral dimension $\lambda_{i=1,2,\dots q}$ associated with each pixel and; the $(q+1)^{th}$ column has the class labels that belong to the set $\{C_1, C_2, \dots, C_m\}$ associated with each pixel (spectral information). In the context of the current study, the total number of data points, $N = 50{,}000$, number of spectral bands, $q = 448$, and the number of classes, $m = 5$; wherein the class labels C_1, C_2, C_3, C_4 and C_5 are associated with 'Non-corrosion', 'Coating', 'Acid' (HCl corrosion), 'Salt' (NaCl corrosion), and 'Sulfate' (Na_2SO_4 corrosion), respectively.

5.3.1. Training and Test Dataset

The training and test dataset required to train and test the machine learning classifier is generated by partitioning the master dataset D. Note that the master dataset generated in this study is a balanced dataset, i.e., each of the five classes has the same number of observations (10,000 each). A partitioning ratio of 70:30 is adopted to obtain training and test datasets, respectively, and the observations in these datasets are randomly chosen from the master dataset. After partitioning D, the balanced training dataset consisted of 35,000 observations with 448 spectral reflectance values such that 7,000 observations belong to each of the class labels C_1, C_2, C_3, C_4 and C_5, and balanced test dataset consisted of 15,000 observations with 448 spectral reflectance values such that 3,000 observations belonged to each of the class labels.

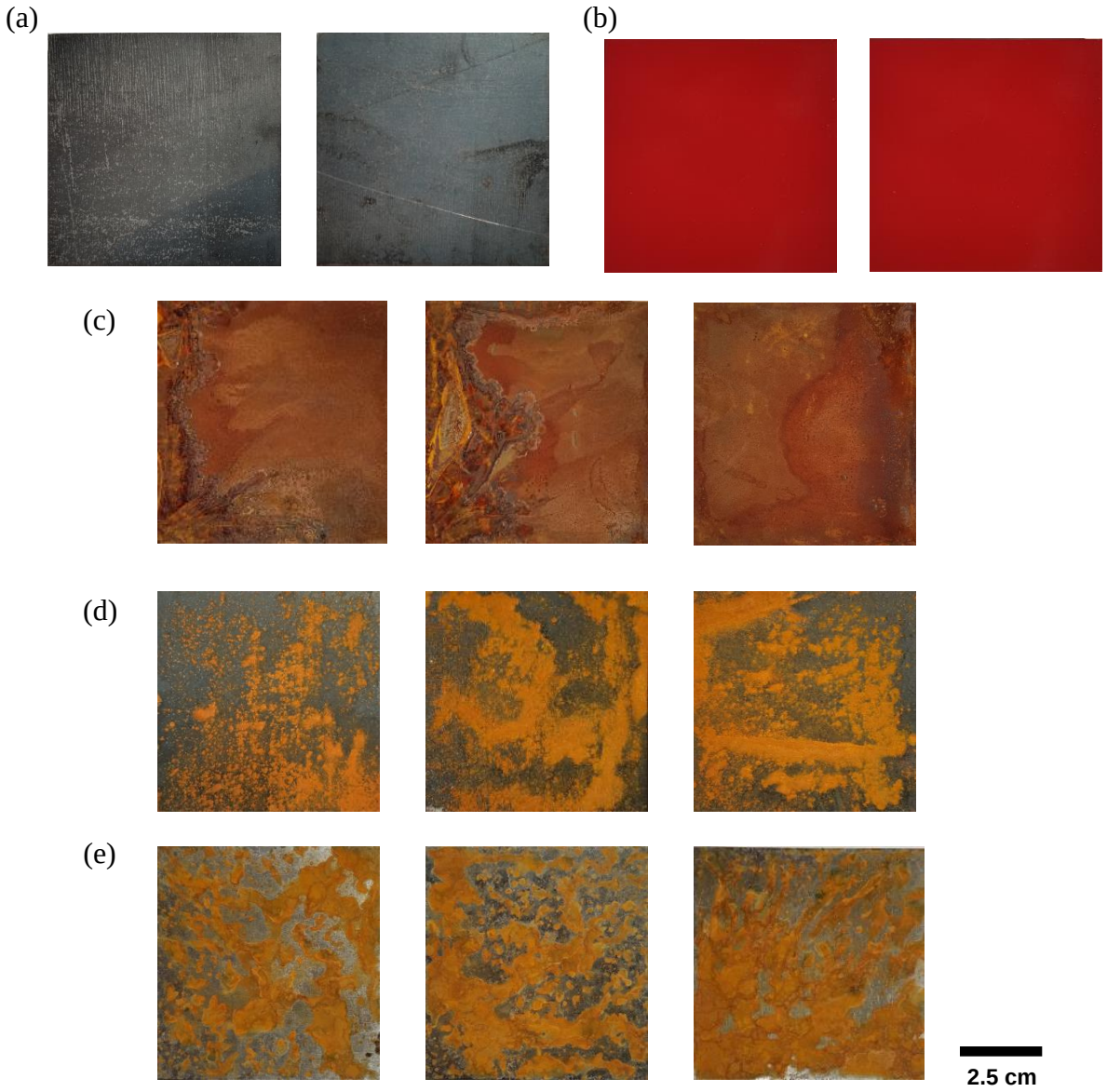


Figure 5.2. ASTM A572 structural steel plates used for the acquisition of hyperspectral images to generate the training dataset. (a) 'Non-corrosion', (b) 'Coating', (c) 'Acid' (1M HCl corroded), (d) 'Salt' (3.5 wt.% NaCl corroded) and, (e) 'Sulfate' (3 wt.% Na_2SO_4 corroded).

5.3.2. Validation Dataset

The calibrated data cubes of partially coated and corroded steel plates (see Figure 5.3) are used as the validation datasets that are not used for the purpose of training. Validation datasets are employed to verify the generalized prediction ability of the trained machine learning classifier. A total of three validation datasets are generated, i.e., the first validation dataset is

associated with a partially coated plate exposed to HCl, the second validation dataset is obtained from a partially coated plate exposed to NaCl, and the third validation dataset is generated by exposing the partially coated plate to Na_2SO_4. Note that each pixel in the validation datasets consists of 448 spectral reflectance values similar to the master dataset (see Figure 5.4).

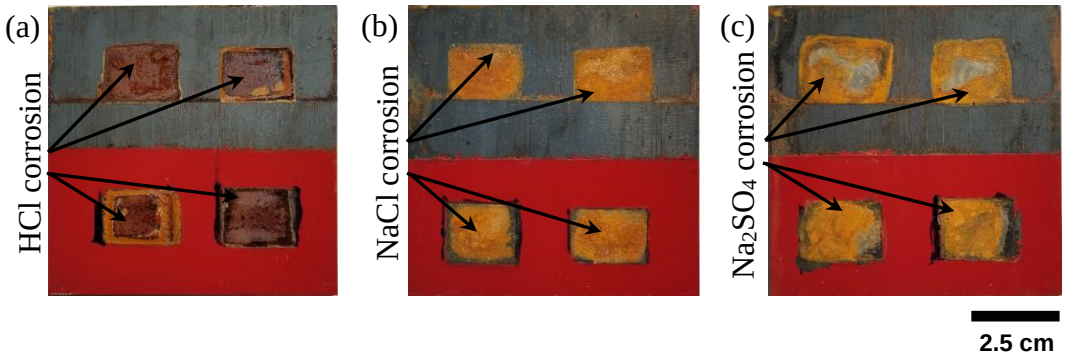


Figure 5.3. Partially coated ASTM A572 structural steel plates used for the acquisition of hyperspectral images to generate the validation dataset. (a) 'Acid' (1M HCl corroded), (b) 'Salt' (3.5 wt.% NaCl corroded) and (c) 'Sulfate' (3 wt.% Na_2SO_4 corroded).

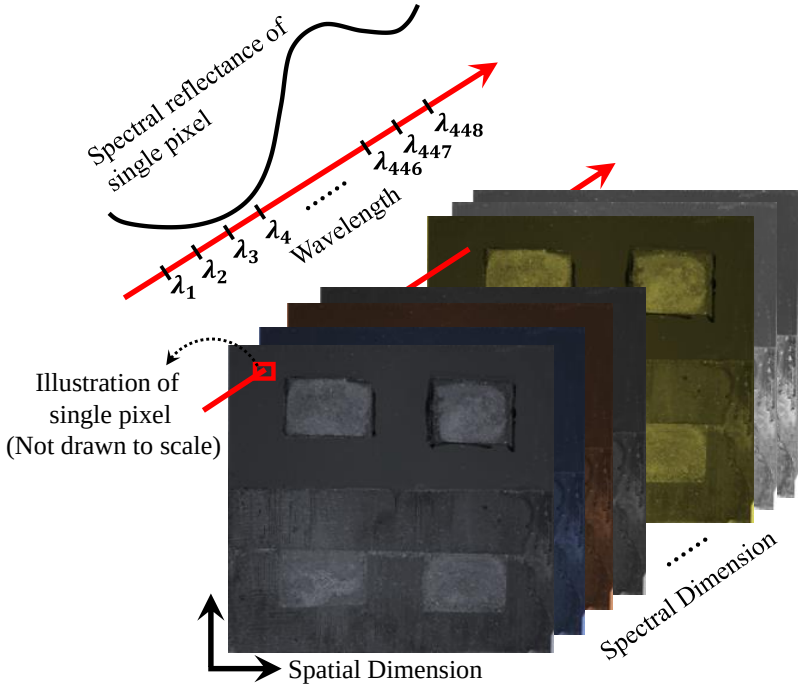


Figure 5.4. Pseudo-schematic of a hyperspectral data cube.

5.4. Methodology

The methodology adopted to identify the 'Non-corrosion', 'Coating', 'Acid', 'Salt', and 'Sulfate' using hyperspectral data involves two tasks, (1) dimensional reduction of the hyperspectral dataset and, (2) training, testing, and validation of the ML classifier. Hyperspectral images generally consist of information from correlated neighboring spectral bands that often convey the same information about the object. In practice, a correlation coefficient matrix R is evaluated from the training dataset and its off-diagonal elements ($\rho_{ij}, i \neq j$) are examined to determine the extent of correlation between the information obtained from the spectral bands.

$$\rho_{ij} = \frac{\mathrm{cov}(x_i, x_j)}{\sqrt{\mathrm{var}(x_i)\mathrm{var}(x_j)}} \tag{5.2}$$

where x_i and x_j are any two column vectors of the spectral bands $X \in R^{N\times q} = (x_1, x_2, \ldots, x_{448})$ in the training dataset such that subscript i and j indicate the column number ranging from 1 to 448, $\mathrm{cov}(.)$ represents the covariance, $\mathrm{var}(.)$ represents the variance and $-1 \leq \rho_{ij} \leq 1$ represents a measure of linear correlation between the reflectance magnitudes of spectral bands x_i and x_j. While the magnitude of ρ_{ij} close to +1 indicates a perfectly linear positive correlation, the magnitude of ρ_{ij} close to -1 indicates a perfectly linear negative correlation. In the current study, the spectral wavelengths in the range of 600 nm-1004 nm are found to be highly positively linearly correlated, and the ones in the range of 397 nm-510 nm are found to be moderately negatively linearly correlated (see Figure 5.5).

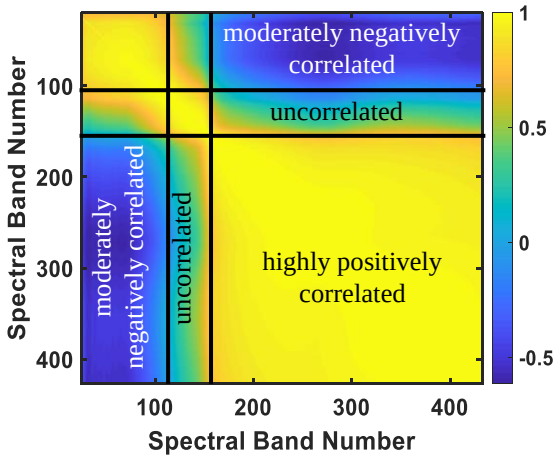


Figure 5.5. Correlation coefficient matrix of wavelengths in VNIR spectra represented as pseudo colors. While the larger positive magnitude (+1) on the color bar indicates higher positive linear correlation, the larger negative magnitude (-1) indicates higher negative linear correlation. The narrow strip in horizontal and vertical directions represents the range of wavelengths that are least correlated and are useful in distinguishing coating from corroded surface (eliminating visual ambiguity).

Including highly correlated features may result in the poor performance of the classifier and will also lead to higher computational effort [56,57]. Dimensional reduction is performed using Principal Component Analysis (PCA) to eliminate the effects of redundant information on the performance of classifiers [58]. PCA finds a new set of latent variables called principal components (PCs) which are expressed as a linear combination of original features. PC's are uncorrelated and are determined with an aim to capture the maximum variance in the given data. A detailed mathematical derivation of PCA can be found elsewhere [59,60]. For further analysis, only the top PC's that account for more than 90% variance within the training data are chosen (see Section 5.5.2). Herein, the MATLAB® code developed in house is used for performing PCA and determining the principal components.

The dimensionally reduced training, test, and validation datasets are obtained by performing the linear transformation of the respective datasets with the principal components and are subsequently used to train, test, and validate a Support Vector Machine (SVM). Support

Vector Machine is a discriminant technique that aims at finding an optimal decision surface that separates the distinct classes such that the margin between the support vectors is maximized [61]. Support vectors are the instances that are closest to the separating decision surface. Mathematically, a linear binary SVM optimization problem is written as [62]

$$\min_{w,b} \frac{1}{2}\|w\|^2 \text{ subjected to } y_i(w^T x + b) - 1 \geq 0 \quad (5.3)$$

where $w \in R^d$ are the weights associated with dimensionally reduced features (reflectance of spectral bands), $b \in R$ is an intercept, $y_{i=1,2} \in \{-1, +1\}$ are the binary class labels and, $x \in R^d$ is a vector of dimensionally reduced features. Note that $w^T x + b = 0$ in Eq. (3) represents a hyperplane. In the context of the current study, since the number of class labels are more than two, one-against all strategy is implemented to obtain m binary decision functions (hyperplanes) to determine the class labels [62].

5.5. Results

In this section, the VNIR spectral profile of the 'Non-corrosion', 'Coating', 'Acid' (HCl corrosion), 'Salt' (NaCl corrosion), and 'Sulfate' (Na_2SO_4 corrosion) is provided and, the dimensionally reduced spectral band reflectances (principal components) of the datasets are determined. Further, the performance of the trained SVM classifier is assessed, and its efficacy on the validation dataset is verified.

5.5.1. Spectral Profiles

A Spectral profile represents the variation in the magnitude of surface reflectance as a function of the wavelength of the light. It differs from one object to the other based on the physical, chemical, and morphological characteristics of the surface and the illumination source [63]. The reflectance of VNIR spectra averaged for 10000 pixels each from the 'Non-corrosion', 'Coating', 'Acid', 'Salt', and 'Sulfate' hyperspectral images (VNIR range: 397nm to 1004 nm) is

shown in Figure 5.6. From Figure 5.6, it can be deduced that the reflectance intensity of 'Non-corrosion', 'Coating', 'Acid', 'Salt', and 'Sulfate' are distinguishable from each other in few regions of the spectra. While the difference in the reflectance intensity for 'Non-corrosion', 'Coating', 'Acid' is clearly noticeable for the wavelength ranging from 523 nm to 580 nm (visible region), the difference in the reflectance intensity for 'Salt' and 'Sulfate' is noticeable for the wavelength ranging from 866 nm to 1004 nm (near-infrared region). The difference in the reflectance intensity observed between the 'Coating' and the rest of the corroded surfaces further reveals that the visual ambiguity can be eliminated if the images are acquired with illuminations in the range of 523 nm to 580 nm wavelengths. The peak observed at the 750nm for 'Acid', 'Salt', 'Sulfate' and 'Coating' indicates the reflectance associated with the visible 'red' color whose range lies between 640nm to 780nm. Although the differences in the reflectance intensities are noticed for a certain range of wavelengths, this alone may not be sufficient to achieve satisfactory classification [64]. The latent variables need to be extracted for improving the performance of the SVM classifier, which is described next.

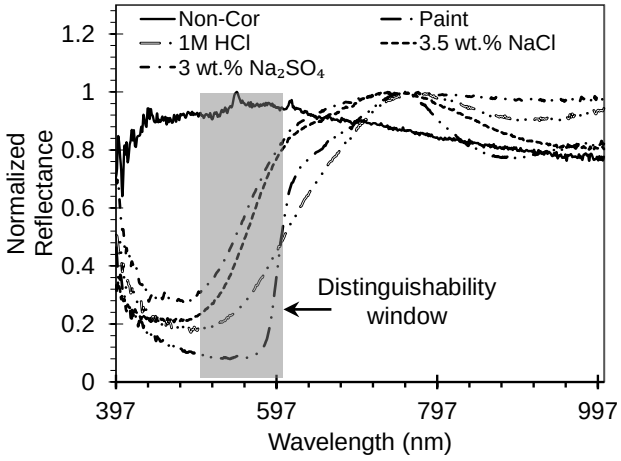


Figure 5.6. Averaged reflectance from VNIR spectra of 'Non-corrosion', 'Coating', 'Acid' (1M HCl corroded), 'Salt' (3.5 wt.% NaCl corroded), and 'Sulfate' (3 wt.% Na_2SO_4 corroded) pixels.

5.5.2. Choosing the Number of Principal Components

For a given dataset, the maximum number of principal components obtained from PCA is equal to the number of features of the dataset. In the context of the current study, the maximum number of principal components is 448. Note that only a few PC's can account for the maximum variance within the data and hence are sufficient for classification. The coefficients used in obtaining the PC's are the eigenvectors or loading vectors associated with the higher magnitude eigenvalues obtained during PCA. After performing PCA on the training dataset, it was found that the first, second, and third PC's captured 79%, 14%, and 3% of the variance within the training data, respectively. That is, the sum of the variance of the first three PC's accounted for 96% of the total variance within the data. Since the first two PC's alone accounted for 93% variance, the first two PC's are considered for the task of classification using SVM. To visualize the distinguishability of 'Non-corrosion', 'Coating', 'Acid', 'Salt', and 'Sulfate' in dimensionally reduced space, the observations of the training dataset with the first two PC's are plotted in Figure 5.7. From Figure 5.7, it is evident that 'Coating' is distinguishable from corroded surface 'Acid', 'Salt' and 'Sulfate', indicating the possible elimination of the visual ambiguity. Further, from Figure 5.7, it can also be found that 'Acid' is distinguishable from 'Salt' and 'Sulfate' classes revealing the identification of chemically distinct corroded surfaces. However, in the case of 'Salt' and 'Sulfate', a significant overlap is observed.

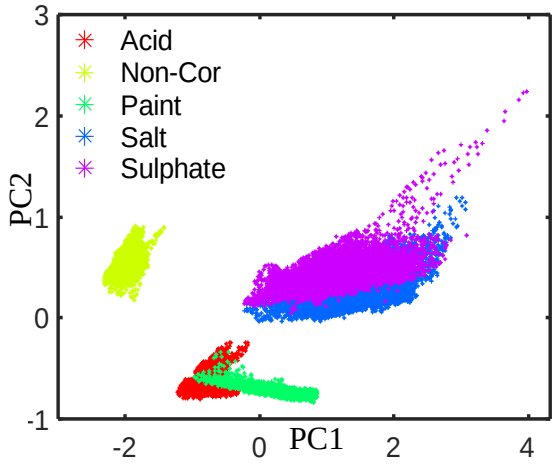


Figure 5.7. Biplot of top two principal components revealing clusters of 'Non-Corrosion', 'Coating', 'Acid', 'Salt' and 'Sulfate' observations. PC1 and PC2 accounts for 79% and 14% of total variance, respectively.

Furthermore, the coefficients of the first two PC's along with their first and second derivatives are shown in Figure 5.8. The coefficients of PC's and their derivatives provide additional information about the spectral profile such as change in the slope and the curvature. This additional information would aid in identifying the key wavelengths that would distinguish the coating and all three chemically distinct corroded surfaces [46]. From Figure 5.8 (a), it is found that the highest and least magnitude of coefficients for the first PC is associated with 760nm and 515nm wavelength, respectively, while for the second PC the highest and least magnitude of the coefficients is associated with 515nm and 760nm wavelength, respectively. Similarly, from Figure 5.8 (b), the wavelengths of 580 nm, 665 nm and 841 nm and, from Figure 5.8 (c), the wavelengths of 507 nm, 680 nm, and 886 nm are identified as the key wavelengths. Comparing all the wavelengths obtained from the coefficients and their derivatives, the following four sets of wavelength ranges are identified as key wavelength ranges for identifying the 'Non-corrosion', 'Coating', 'Acid', 'Salt', and 'Sulfate' surface: 500-520 nm, 660-680 nm, 760-770 nm and 830-850 nm.

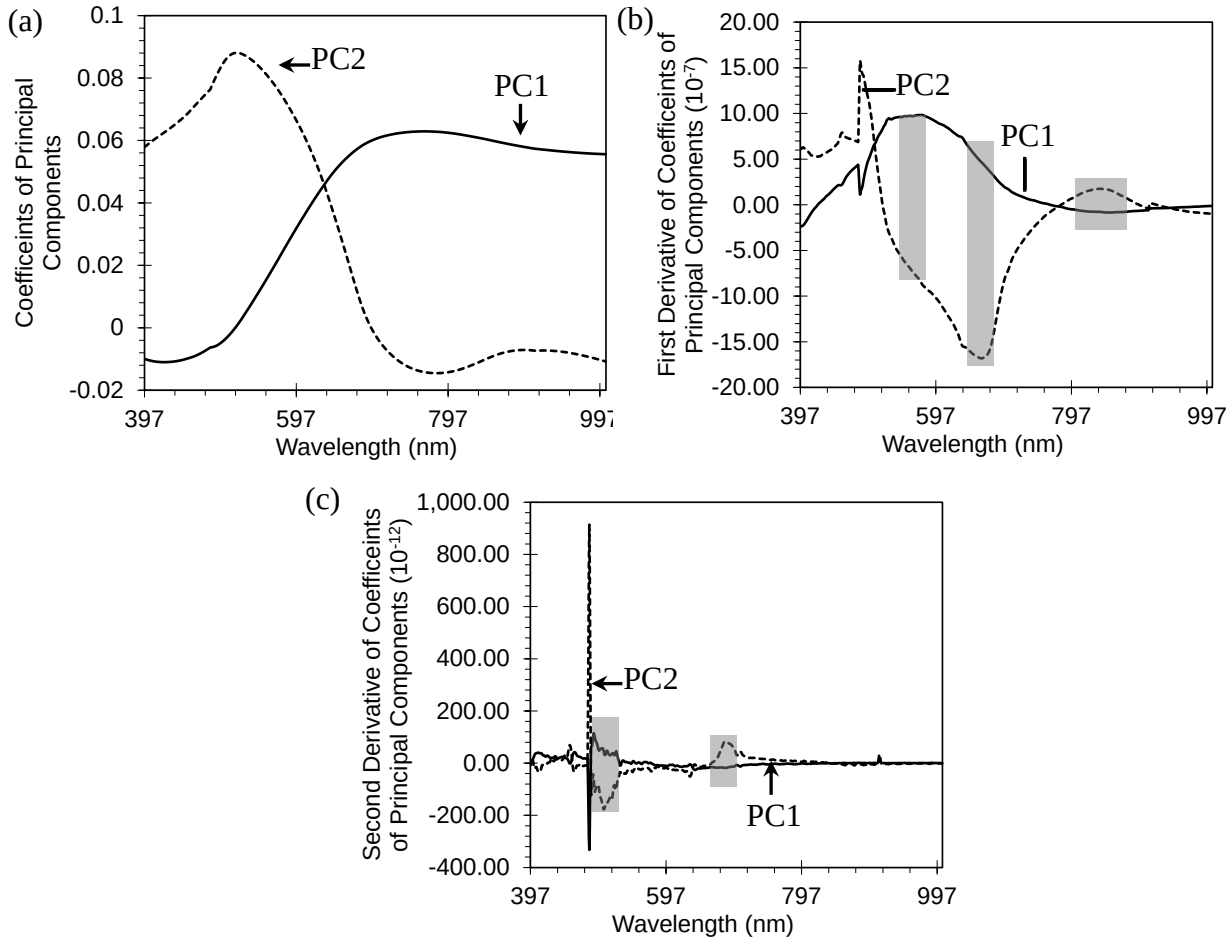


Figure 5.8. (a) Coefficients of top two principal components; and their derivatives, (b) first derivative, and (c) second derivative.

5.5.3. Performance Assessment of Trained SVM

The performance of the trained SVM classifier to predict the class labels accurately is assessed using the test dataset (see Section 5.3.1), i.e., the known class labels of the test dataset and the predicted class labels from the trained SVM classifier are compared. A summary of the correct and incorrect classifications is provided as a confusion matrix (C) in **Error! Reference s ource not found.**. Confusion matrix is a m × m square matrix, where m represents the number of class labels and each element C_{ij} of C represents the frequency of instances from validation dataset that are assigned class label j by the classifier which in reality belongs to class label i [65]. For assessing the performance of the classifier, a metric referred to as prediction accuracy

is used, which is defined as the ratio of the total number of observations whose class labels are correctly identified to the total number of observations present in the test dataset. Mathematically it is expressed as

$$A_c = \frac{\sum_{i=1}^{m} C_{ii}}{\sum_{i=1}^{m} \sum_{j=1}^{m} C_{ij}} \times 100\% \tag{5.4}$$

In this study, the SVM classifier resulted in an accuracy of 93.2%, i.e., 93.2% ('Non-Cororsion'-100%, 'Coating'-99%, 'Acid'-99%, 'Salt'-82% and 'Sulfate'-87%) observations of the test data are correctly classified. While 'Non-corrosion', 'Coating', and 'Acid' are predicted accurately i.e., misclassifications were less than 2%, 18% of 'Salt' class labels are misclassified as 'Sulfate' class labels and 13% of 'Sulfate' class labels are misclassified as 'Salt' class labels. Accurate prediction of 'Coating' indicates that its visual ambiguity between corroded and coated surfaces is eliminated, and accurate prediction of 'Acid' corrosion reveals that chemically distinct corrosion can be identified using the proposed approach. As mentioned earlier, the misclassification in 'Salt' and 'Sulfate' class labels may be attributed to the overlap of their respective spectra in a dimensionally reduced space due to chemical and morphological similarities between the corrosion products.

Table 5.2. Confusion matrix of correctly and incorrectly classified class labels (in percentage fraction).

Class		**Actual Class Label**				
		Non-Corrosion	Coating	Acid	Salt	Sulfate
Predicted Label	Non-Corrosion	1	0	0	0	0
	Coating	0	0.99	0.03	0	0
	Acid	0	0.01	0.97	0	0
	Salt	0	0	0	0.82	0.13
	Sulfate	0	0	0	0.18	0.87

5.5.4. Validation

Further validation is performed to verify the generalization capability of trained SVM classifiers on the validation dataset that was not used for training purposes and is different from the test dataset. The validation images used for this purpose are shown in the Figure 5.3. A pixel-wise approach is employed wherein the spectral data (reflectance) of each pixel of the validation hyperspectral cube is chosen one after another for classification. The results obtained after classification is shown in Figure 5.9(a)-(c). Note that the ground truth images are also provided in Figure 5.9 for qualitative comparison. From Figure 5.9(a), it can be observed that the SVM classifier was able to predict the 'Acid', 'Non-corrosion' and 'Coating' class labels correctly with slight misclassifications at the top left corner of the plate where 'Coating' is misclassified as 'Acid'. However, in the case of 'Salt' and 'Sulfate' validation images, some portions of 'Salt' class label are misclassified as 'Sulfate' class label, and some portion of 'Sulfate' corrosion is misclassified as 'Salt' corrosion as mentioned in the previous section. The misclassifications in the case of 'Salt' and 'Sulfate' class labels could be attributed to the presence of similar iron oxide corrosion products in 'Salt' and 'Sulfate' corroded specimens. X-ray diffraction (XRD) was performed using the Bruker Xray diffractometer equipped with Cu-K_{alpha} radiation at 40 kV and 30 mA to prove the presence of similar corrosion products. A 2θ range of 10° to 85° was maintained during data acquisition. The corrosion product identifications were performed by comparison to the International Center for Diffraction Database (ICDD) available on Match3® software.

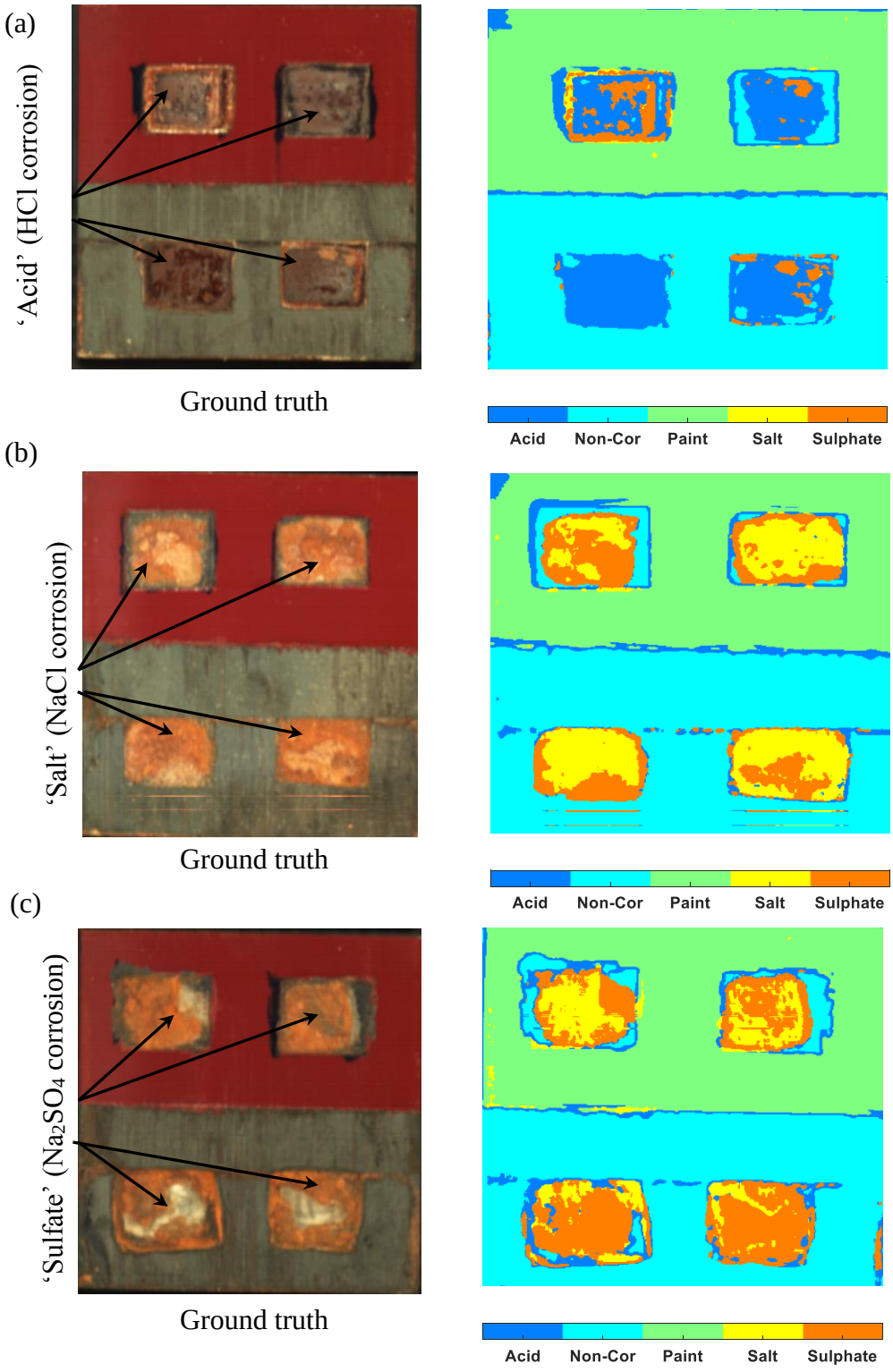


Figure 5.9. Corrosion source identification: validation for (a) 'Acid' (1M HCl), (b) 'Salt' (3.5 wt.% NaCl) and (c) 'Sulfate' (3 wt.% Na_2SO_4).

The XRD of ‘Salt’ and ‘Sulfate’ corroded samples are provided in Figure 5.10. While the XRD spectra of the ‘Salt’ sample (see Figure 5.10(a)) revealed the presence of Akageneite and Goethite iron oxide minerals, the XRD spectra of the ‘Sulfate’ sample (see Figure 5.10(b)) revealed the presence of Goethite and Lepidocrocite iron oxide minerals. The corrosion products identified in this study through XRD are in good agreement with the results published in the literature [23,25,30,31]. The presence of similar iron oxide mineral Goethite in both ‘Salt’ and ‘Sulfate’ corroded specimen is a possible explanation for the overlap between the reflectance of VNIR spectral bands. Interestingly, both ‘Salt’ and ‘Sulfate’ corroded specimens look similar also from the hue point of view (i.e., visual ambiguity) despite the difference in the corrosion products, namely, Akageneite and Lepidocrocite. A more detailed description of color discrimination of various iron oxide minerals is provided elsewhere [63, 66–68]. The misclassification between the ‘Salt’ and ‘Sulfate’ corroded samples can hence be attributed to the presence of Goethite and similar hue in the visible spectrum. However, the misclassification rate is below 20%, which is within the acceptable range in the structural health monitoring community [69]. In the future work, the authors would carry out FTIR/Raman spectroscopy studies to show the relationship between the corrosion products and the absorbed bands of light.

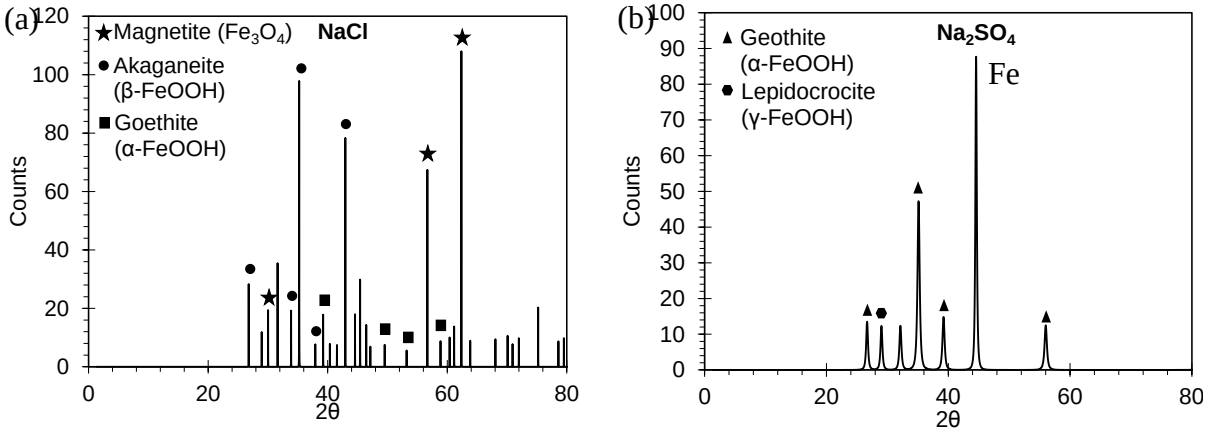


Figure 5.10. XRD spectra for (a) ‘Salt’ (3.5 wt.% NaCl), and (b) ‘Sulfate’ (3 wt.% Na_2SO_4) surfaces.

5.6. Conclusions

Hyperspectral data of ‘Non-corrosion’, ‘Coating’, ‘Acid’, ‘Salt’ and ‘Sulfate’ labeled specimens are acquired in the VNIR range of the EM spectrum and are used to train, test and validate the SVM classifier. The conclusions drawn from this study are as follows

1. The visual ambiguity between the coatings and corroded surfaces with a similar hue can be eliminated using the VNIR spectra. In this study, the top two principal components of the reflectance of VNIR spectra, along with an SVM, are used to eliminate visual ambiguity between the coating and corroded surfaces.
2. The source of corrosion, i.e., ‘Acid’, ‘Salt’ and ‘Sulfate’ is also identified using the top two principal components of the reflectance of VNIR spectra. The trained SVM classifier was able to identify the source of corrosion with an overall accuracy of 94%.
3. The misclassifications in the case of ‘Coating’ and ‘Acid’ data is less than 2%. However, the misclassifications observed in the case of ‘Salt’ and ‘Sulfate’ data is 18% and 13%, respectively.
4. Misclassifications of ‘Salt’ and ‘Sulfate’ class labels may be attributed to the presence of similar iron oxide corrosion products. For confirmation, XRD characterization tests were carried out, and the corrosion product Goethite was found on both ‘Salt’ and ‘Sulfate’ corroded surfaces.
5. Among the 448 spectral bands that were acquired from the hyperspectral images, only a few spectral bands were found to play an important role in the identification of corrosion sources and elimination of visual ambiguity. The important ranges of the wavelengths of the spectral bands identified for the classification of coating and corroded surfaces are 500-520 nm, 660-680 nm, 760-770 nm, and 830-850 nm.

Push broom hyperspectral imaging systems are expensive and are primarily developed for benchtop applications limiting their use in field applications. Building a customized multispectral imagining sensor with the ability to capture the spectral information in the desired range of wavelengths may be a feasible and an economical option. The multispectral sensors are portable and hence could be easily mounted on UAVs for easy navigation and maneuvers in the field. The key wavelengths identified in this study can be used to build a multispectral imagining sensor that can eliminate the visual ambiguity and detect the chemically distinctive corroded surfaces in civil, structural, aerospace, and offshore structures.

5.7. References

1. Ahuja, SK, Shukla, MK. A survey of computer vision based corrosion detection approaches. Smart Innov Syst and Technol 2018; 84: 55–63. Springer Science and Business Media Deutschland GmbH. DOI: 10.1007/978-3-319-63645-0_6
2. Medeiros, FNS, Ramalho, GLB, Bento, MP, et al. On the evaluation of texture and color features for nondestructive corrosion detection. EURASIP J Adv Signal Process 2010; 1–7. DOI: 10.1155/2010/817473
3. Jahanshahi, MR, Kelly, JS, Masri, SF, et al. A survey and evaluation of promising approaches for automatic image-based defect detection of bridge structures. Struct Infrastruct Eng 2009; 5: 455–486. DOI: 10.1080/15732470801945930
4. Chen, PH, Chang, LM. Artificial intelligence application to bridge painting assessment. Autom Constr 2003; 12: 431–445. DOI: 10.1016/S0926-5805(03)00016-5
5. Ranjan, RK, Gulati, T. Condition assessment of metallic objects using edge detection. Int J Adv Res Comput Sci Softw Eng 2014; 4: 253–258.

6. Ghanta, S, Karp, T, Lee, S. Wavelet domain detection of rust in steel bridge images. In: 2011 IEEE international conference on acoustics, speech and signal processing (ICASSP), Prague, Czech Republic, 22–27 May 2011, 1033–1036. DOI: 10.1109/ICASSP.2011.5946583
7. Feliciano, F, Mainier, F. Possible use of texture parameters to corrosion evolution analysis. In: IWSSIP 2014 proceedings, Dubrovnik, Croatia, 12–15 May 2014.
8. Lee, S, Chang, L-M. Digital image processing methods for assessing bridge painting rust defects and their limitations. In: International conference on computing in civil engineering 2005, Cancun, Mexico, 12-15 July 2005, American Society of Civil Engineers, 1–12. DOI: 10.1061/40794(179)80
9. Chen, P-H, Yang, Y-C, Chang, L-M. Automated bridge coating defect recognition using adaptive ellipse approach. Autom Constr 2009; 18: 632–643. DOI: 10.1016/j.autcon.2008.12.007
10. Shen, H-K, Chen, P-H, Chang, L-M. Automated steel bridge coating rust defect recognition method based on color and texture feature. Autom Constr 2013; 31: 338–356. DOI: 10.1016/j.autcon.2012.11.003
11. Liao, K-W, Lee, Y-T. Detection of rust defects on steel bridge coatings via digital image recognition. Autom Constr 2016; 71: 294–306. DOI: 10.1016/j.autcon.2016.08.008
12. Naik, DL, Sajid, HU, Kiran, R. Texture-based metallurgical phase identification in structural steels: a supervised machine learning approach. Metals 2019; 9: 546. DOI: 10.3390/met9050546
13. Nash, WT, Powell, CJ, Drummond, T, et al. Automated corrosion detection using crowd sourced training for deep learning. Corrosion 2020; 76(2): 135–141.

14. Khayatazad, M, De Pue, L, De Waele, W. Detection of corrosion on steel structures using automated image processing. Dev Built Environ 2020; 3: 100022. DOI: 10.1016/j.dibe.2020.100022https://doi.org/10.1016/j.dibe.2020.100022

15. Petricca, L, Moss, T, Figueroa, G, et al. Corrosion detection using A.I.: a comparison of standard computer vision techniques and deep learning. Model 2016: 91: 9. DOI: 10.5121/csit.2016.60608

16. Bonnin-Pascual, F, Ortiz, A. Corrosion detection for automated visual inspection. In: Development in Corrosion Protection. USA: InTech, 2014, pp. 619–632 DOI: 10.5772/57209

17. Hoang, ND . Image processing-based pitting corrosion detection using metaheuristic optimized multilevel image thresholding and machine-learning approaches. Math Probl Eng 2020; 2020: 6765274. DOI: 10.1155/2020/6765274.

18. Naik, DL, Sajid, HU, Kiran, R, et al. Detection of corrosion-indicating oxidation product colors in steel bridges under varying illuminations, shadows, and wetting conditions. Metals 2020; 10: 1439. DOI: 10.3390/met10111439

19. Fouda, AS, Abousalem, AS, El-Ewady, GY. Mitigation of corrosion of carbon steel in acidic solutions using an aqueous extract of Tilia cordata as green corrosion inhibitor. Int J Ind Chem 2017; 8: 61–73. DOI: 10.1007/s40090-016-0102-z

20. Al-Sodani, KAA, Maslehuddin, M, Al-Amoudi, OSB, et al. Efficiency of generic and proprietary inhibitors in mitigating corrosion of carbon steel in chloride-sulfate environments. Sci Rep 2018; 8: 11443. DOI: 10.1038/s41598-018-29413-7

21. Mahdavian, M, Naderi, R. Corrosion inhibition of mild steel in sodium chloride solution by some zinc complexes. Corros Sci 2011;53:1194–1200. DOI: 10.1016/j.corsci.2010.12.013

22. Suzuki, S . Surface analysis of oxides and corrosion products formed on surfaces of iron-based alloys. In: Characterization of Corrosion Products on Steel Surfaces, Berlin Heidelberg, Springer; 2006, Vol. 7. 131–158. DOI: 10.1007/978-3-540-35178-8_7

23. Takahashi, Y, Matsubara, E, Suzuki, S, et al. In-situ X-ray diffraction of corrosion products formed on iron surfaces. Mater Res 2005;46(3):637–642.

24. Antunes, RA, Costa, I, Faria, DLAd. Characterization of corrosion products formed on steels in the first months of atmospheric exposure. Mater Res 2003; 6: 403–408. DOI: 10.1590/s1516-14392003000300015

25. Yamashita, M, Konishi, H, Kozakura, T, et al. In Situ Observation of Initial Rust Formation Process on Carbon Steel under Na2SO4 and NaCl Solution Films with Wet/dry Cycles Using Synchrotron Radiation X. Amsterdam, Wiley, NY: Elsevier, 2005.

26. Alcántara, J, de la Fuente, D, Chico, B, et al. Marine atmospheric corrosion of carbon steel: a review. Materials 2017; 10: 406. DOI: 10.3390/ma10040406

27. Daukšys, M, Bitautaitė, E, Mockienė, J, et al. NaCl and Na2SO4 solution effect on weathering steel visual appearance when the ambient temperature changes cyclically. Cogent Eng 2019; 6: 1659124. https://doi.org/10.1080/23311916.2019.1659124

28. Arzola-Peralta, S, Genescá Llongueras, J, Palomar-Pardavé, M, et al. Study of the electrochemical behaviour of a carbon steel electrode in sodium sulfate aqueous solutions using electrochemical impedance spectroscopy. J Solid State Electrochem 2003; 7: 283–288. DOI: 10.1007/s10008-002-0344-x

29. Askey, A, Lyon, SB, Thompson, GE, et al. The corrosion of iron and zinc by atmospheric hydrogen chloride. Corros Sci 1993; 34: 233–247. DOI: 10.1016/0010-938X(93)90004-Z

30. Sajid, HU, Kiran, R, Qi, X, et al. Employing corn derived products to reduce the corrosivity of pavement deicing materials. Constr Build Mater 2020; 263: 120662. DOI: 10.1016/j.conbuildmat.2020.120662https://doi.org/10.1016/j.conbuildmat.2020.120662

31. Sajid, HU, Kiran, R. Influence of corrosion and surface roughness on wettability of ASTM A36 steels. J Constr Steel Res 2018; 144: 310–326. DOI: 10.1016/j.jcsr.2018.01.023

32. Mohan, R, Joseph, A. Corrosion protection of mild steel in hydrochloric acid up to 313 K using propyl benzimidazole: electroanalytical, adsorption and quantum chemical studies. Egypt J Pet 2018; 27: 11–20. DOI: 10.1016/j.ejpe.2016.12.003https://doi.org/10.1016/j.ejpe.2016.12.003

33. Pradityana, A, Sulistijono, Shahab, A, et al. Inhibition of corrosion of carbon steel in 3.5% NaCl solution by myrmecodia pendans extract. Int J Corros 2016; 2016: 6058286. DOI: 10.1155/2016/6058286https://doi.org/10.1155/2016/6058286

34. Hasan, BO, Sadek, SA. Corrosion Behavior of Carbon Steel in Oxygenated Sodium Sulphate Solution under Different Operating Conditions. UK: SEP-Publisher, 2013.

35. Oh, SJ, Cook, DC, Townsend, HE. Atmospheric corrosion of different steels in marine, rural and industrial environments. Corros Sci 1999; 41: 1687–1702. DOI: 10.1016/S0010-938X(99)00005-0

36. Morcillo, M, Alcántara, J, Díaz, I, et al. Marine atmospheric corrosion of carbon steels. Rev Metal 2015; 51(2): e045. https://doi.org/10.3989/revmetalm.045.

37. Li, X, Li, R, Wang, M, et al. Hyperspectral imaging and their applications in the nondestructive quality assessment of fruits and vegetables. In: Hyperspectral Imaging Agriculture, Food and Environment. InTechOpen, 2018. DOI: 10.5772/intechopen.72250

38. Liu, Z, Jing, W. Hyperspectral endmember detection method based on Bayesian decision theory. Adv Intell Soft Comput 2012; 114: 727–732. DOI: 10.1007/978-3-642-03718-4_89https://doi.org/10.1007/978-3-642-03718-4_89

39. Luna Maldonado, AI, Rodriguez-Fuentes, H, Vidales Contreras, JA. Hyperspectral Imaging in Agriculture, Food and Environment. IntechOpen, London, 2018.

40. Gowen, A, O'Donnell, C, Cullen, P, et al. Food Science &Hyperspectral Imaging–An Emerging Process Analytical Tool for Food Quality and Safety Control. Wiley, NY: Elsevier, 2018.

41. Cruz, J, Bautista, M, Amigo, JM, et al. Nir-chemical imaging study of acetylsalicylic acid in commercial tablets. Talanta 2009; 80: 473–478. DOI: 10.1016/j.talanta.2009.07.008

42. Gendrin, C, Roggo, Y, Collet, C. Content uniformity of pharmaceutical solid dosage forms by near infrared hyperspectral imaging: a feasibility study. Talanta 2007; 73: 733–741. DOI: 10.1016/j.talanta.2007.04.054https://doi.org/10.1016/j.talanta.2007.04.054

43. Chen, YR, Chao, K, Kim, MS. Machine vision technology for agricultural applications. Comput. Electron. Agric 2002; 36: 173–191. DOI: 10.1016/S0168-1699(02)00100-X

44. Teke, M, Deveci, HS, Haliloglu, O, et al. A short survey of hyperspectral remote sensing applications in agriculture. In: 2013 6th international conference on recent advances in space technologies (RAST), Istanbul, Turkey, 12–14 June 2013, 171–176. DOI: 10.1109/RAST.2013.6581194

45. Caballero, D, Calvini, R, Amigo, JM. Hyperspectral imaging in crop fields: precision agriculture. Data Handl Sci Technol 2020; 32: 453–473. DOI: 10.1016/B978-0-444-63977-6.00018-3

46. Huang, M, Wan, X, Zhang, M, et al. Detection of insect-damaged vegetable soybeans using hyperspectral transmittance image. J Food Eng 2013; 116: 45–49. DOI: 10.1016/j.jfoodeng.2012.11.014

47. Lara, MA, Lleó, L, Diezma-Iglesias, B, et al. Monitoring spinach shelf-life with hyperspectral image through packaging films. J Food Eng 2013; 119: 353–361. DOI: 10.1016/j.jfoodeng.2013.06.005

48. Lu, R, Chen, Y-R. Hyperspectral imaging for safety inspection of food and agricultural products. In: Chen, Y-R (ed) Pathogen Detection and Remediation for Safe Eating, Vol. 3544, SPIE, 1999, 121–133. DOI: 10.1117/12.335771

49. Xu, Y, Chen, Q, Liu, Y, et al. A novel hyperspectral microscopic imaging system for evaluating fresh degree of pork. Korean J Food Sci Anim Resour 2018; 38: 362–375. DOI: 10.5851/kosfa.2018.38.2.362

50. Fischer, C, Kakoulli, I. Multispectral and hyperspectral imaging technologies in conservation: current research and potential applications. Stud Conserv 2006; 51: 3–16. DOI: 10.1179/sic.2006.51.supplement-1.3

51. Rosi, F, Miliani, C, Braun, R, et al. Noninvasive analysis of paintings by mid-infrared hyperspectral imaging. Angew Chem - Int Ed 2013; 52: 5258–5261. DOI: 10.1002/anie.201209929

52. Legrand, S, Vanmeert, F, Van der Snickt, G, et al. Examination of historical paintings by state-of-the-art hyperspectral imaging methods: from scanning infra-red spectroscopy to computed X-ray laminography. Herit Sci 2014; 2. DOI: 10.1186/2050-7445-2-13

53. Merin Antony, M, Sandeep, CSS, Vadakke Matham, M. Monitoring system for corrosion in metal structures using a probe based hyperspectral imager. In: Fujigaki, M, Xie, H, Zhang, Q, et al. (eds) Seventh Int. Conf. Opt. Photonic Eng. (icOPEN 2019). Bellingham,WA: SPIE; 2019, 82. DOI: 10.1117/12.2542907

54. Li, Y, Kontsos, A, Bartoli, I. Automated rust-defect detection of a steel bridge using aerial multispectral imagery. J Infrastruct Syst 2019; 25: 04019014. DOI: 10.1061/(asce)is.1943-555x.0000488

55. Polak, A, Kelman, T, Murray, P, et al. Hyperspectral imaging combined with data classification techniques as an aid for artwork authentication. J Cult Herit 2017; 26: 1–11. DOI: 10.1016/j.culher.2017.01.013

56. Datta, A, Ghosh, S, Ghosh, A. Unsupervised band extraction for hyperspectral images using clustering and kernel principal component analysis. Int J Remote Sens 2017; 38: 850–873. DOI: 10.1080/01431161.2016.1271470https://doi.org/10.1080/01431161.2016.1271470

57. Su, J, Yi, D, Liu, C, et al. Dimension reduction aided hyperspectral image classification with a small-sized training dataset: experimental comparisons. Sensors 2017; 17(12): 2726. DOI: 10.3390/s17122726

58. Dong, P, Liu, J. Hyperspectral image classification using support vector machines with an efficient principal component analysis scheme. In: Advances in Intelligent and Soft

Computing. Berlin, Heidelberg: Springer, 2011, 131–140. DOI: 10.1007/978-3-642-25664-6_17

59. Jolliffe, IT, Cadima, J. Principal component analysis: a review and recent developments. Philos Trans R Soc A Math Phys Eng Sci 2016; 374: 20150202. DOI: 10.1098/rsta.2015.0202

60. Rencher, AC, Christensen, WF. Methods of Multivariate Analysis. 3rd edition. New Jersey, USA: Wiley, 2002

61. Zhang, Z, Flores, P, Igathinathane, C, et al. Wheat lodging detection from UAS imagery using machine learning algorithms. Remote Sens 2020; 12: 1838. DOI: 10.3390/rs12111838

62. Awad, M, Khanna, R. Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers. New York, NY: Apress Media LLC, 2015. DOI: 10.1007/978-1-4302-5990-9

63. Pedrosa, J, Costa, BFO, Portugal, A, et al. Controlled phase formation of nanocrystalline iron oxides/hydroxides in solution - An insight on the phase transformation mechanisms. Mater Chem Phys 2015; 163: 88–98. DOI: 10.1016/j.matchemphys.2015.07.018

64. Zhao, Y, Zhu, S, Zhang, C, et al. Application of hyperspectral imaging and chemometrics for variety classification of maize seeds. RSC Adv 2018; 8: 1337–1345. DOI: 10.1039/c7ra05954j

65. Naik, DL, Kiran, R. Identification and characterization of fracture in metals using machine learning based texture recognition algorithms. Eng Fract Mech 2019; 219: 106618. DOI: 10.1016/j.engfracmech.2019.106618

66. Scheinost, AC, Schwertmann, U. Color identification of iron oxides and hydroxysulfates. Soil Sci Soc Am J 1999; 63: 1463–1471. DOI: 10.2136/sssaj1999.6351463x

67. Schwertmann, U . Relations between Iron Oxides, Soil Color, and Soil Formation, Hoboken, NJ: John Wiley & Sons, Ltd, 2015, 51–69. DOI: 10.2136/sssaspecpub31.c4

68. Schwertmann, U . Some Properties of Soil and Synthetic Iron Oxides. Iron Soils Clay Miner. Dordrecht Netherlands: Springer, 1988, 203–250. DOI: 10.1007/978-94-009-4007-9_9

69. Azimi, M, Eslamlou, A, Pekcan, G. Data-driven structural health monitoring and damage detection through deep learning: state-of-the-art review. Sensors 2020; 20: 2778. DOI: 10.3390/s20102778

70. Weiss, K, Khoshgoftaar, TM, Wang, D. A survey of transfer learning. J Big Data 2016; 3: 1–40. DOI: 10.1186/S40537-016-0043-6.

6. LITERATURE REVIEW OF EXPLAINABLE ARTIFICIAL INTELLIGENCE (XAI)

In this chapter a brief background on history of XAI is provided. Further the review of currently available XAI algorithms are summarized along with their advantages and disadvantages.

6.1. History of XAI

The history of explanation in intelligent systems can be traced back to 1970's when the expert systems were built to retain the knowledge of a bounded domain problem in the form of IF-THEN rules for reasoning the prediction. Since then various explanation systems emerged in the field of artificial intelligence which could be categorized into three distinct generations (see Figure 6.1) [1]: (1) first generation-expert systems (1970's), (2) second generation knowledge-based tutors (mid 1980's) and (3) third generation systems (after 2010). A brief overview of all three generation systems is as follows.

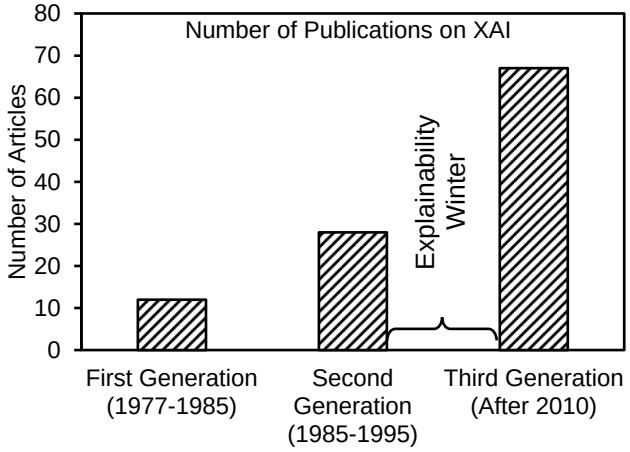


Figure 6.1. Number of publications identified in the field of explainable artificial intelligence (XAI) (Redrawn [1]).

6.1.1. First Generation-Expert System

According to Feigenbaum [2], an expert system is an intelligent computer program that draws solutions for problems in the bounded domain through the knowledge and inference

procedures entailing facts and heuristics which would otherwise require human attention. In other words, it provides the solutions to the problems in the narrow domains through heuristic knowledge fed by an expert into the system and cannot be a general solver. For instance, MYCIN [3] was developed for diagnosing only the infectious blood diseases which could be considered as a sub-domain of medical field and cannot be generalized to other sub-domains of the same field. Expert systems are assistants to decision makers and not the substitutes for them. Some of the first generation explanation systems include MYCIN, the Digitalis Therapy Advisor [4], XPLAIN [5], BLAH [6] which typically operated in human-computer interaction mode where the queries by the users were answered [7].

The schematic of the architecture used to build an expert system is shown in Figure 6.2 which consists of four main components [7], (1) knowledge acquisition module (from domain experts), (2) knowledge base, (3) inference engine and (4) input/output interface. Knowledge base consists of domain specific knowledge elicited from the domain experts, knowledge engineers and other sources such as books, manuals etc. The knowledge stored in knowledge base is in a representable form such as simple if-then rules, semantic networks, conceptual graphs, frame and object oriented schemes and petri nets. The inference engine constitutes algorithms that extract the knowledge stored in knowledge base to solve a problem. The input/output interface facilitates the interaction of the system with the user through GUI and provide a reason why those test results were required for diagnosing. Expert systems developed in other areas included the tasks of classification.

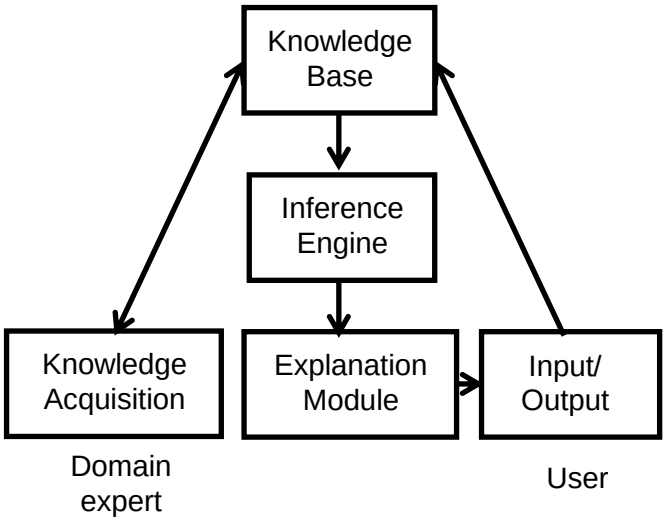


Figure 6.2. Architecture of first generation-expert system [7].

Although first generation expert systems have exhibited good performance in terms of deriving correct recommendations, they were prone to some limitations. Keravnou et.al. [8] classified these limitations into three categories, namely human-computer interaction, problem-solving flexibility and extensibility/maintainability. In human-computer interaction, redundant and incoherent sequence of questions were noticed, and the users were allowed to enter information only in specific formats else the information was ignored. Due to limited data storage capabilities, the historical information of the user was not maintained and the information from user is sought every time during new consultation. In problem-solving flexibility, either the inability of the system to recognize difficult or rare problem posed to the system was noticed or degradation in performance was observed. In extensibility (maintainability), the system was not facilitated with the ability to evolve on experience in problem-solving i.e. learning.

6.1.2. Second Generation-Knowledge Based Tutors

Second generation systems addressed some of the limitations of first generation-expert systems i.e. it enhanced the explanation facilities: the explanations they offer are richer and more coherent, they are better adapted to the user's needs and knowledge, and the explanation

facilities can offer clarifying explanations to correct misunderstandings [7]. Major developments that have differentiated explanation in second generation systems from explanation in first generation systems: 1) new architectures have been developed to capture more knowledge that is needed for explanation, and 2) more powerful explanation generators have been developed from problem-solving point of view. List of various architectures developed in second generation systems are as follows [9]: first principles architecture [10], Neomycin and heuristic classification architecture [11], multi-levels of abstraction architecture [8], Generic tasks architecture [12], explainable expert systems framework [13], Steel's model-based architecture [14] and reconstructive explanation [7].

First principles architecture is built based on the premise that the reasoning should be derived from first principles. Such systems should possess the ability to impose and relax assumptions as necessary. Neomycin architecture uses metarules to provide higher-level representations for problem-solving strategy that were implicit in MYCIN framework. This was achieved by two sub-modules EXPLORE-AND-REFINE and FINDOUT wherein the former one was associated with the ruleset and the later was associated with the question-asking [7]. In Neomycin, rules in the ruleset were accessed through a control process which involved its activation through metarules. Multi-levels of abstraction architecture included more detailed pathophysiological knowledge (accurate attribution of findings) and less detailed phenomenological knowledge (global view of search space) which were modelled through causal networks embodying a link that indicated the relation between attributes of cause and effect. Generic tasks architecture is based on compilation of deep knowledge from the task perspective i.e. the experiential shallow knowledge representing the patterns of particular task and the deep knowledge obtained from the first principles. While experiential knowledge is considered as task

dependent and can deal with only routinely occurring instances, the deep knowledge is considered as task independent and can deal with reasoning of rare instances.

Unlike first generation expert systems which lacks the ability to provide good justifications, the explainable expert systems captures the design information and enriches the explanations by building a system containing the facts about the domain, its terminology and general problem-solving strategies. In Steel's model-based architecture the reasoning is represented as a structure of tasks whose execution is carried out using the control structure. The process of carrying out the tasks depends on domain specific knowledge, referred to as problem-solving methods. The connection between domain model and the problem-solving method is initiated by heuristic role annotations. In reconstructive explanations the system constructs the post hoc explanations that justifies the prediction by using a separate knowledge base.

6.1.3. Third Generation Systems

Between 1990's and 2010's, which is referred to as an explainability winter, not much progress took place in the field of XAI [1] (see Figure 6.1). More focus was put on developing state-of-art AI algorithms which aimed at higher accuracies of prediction and enhanced model performance. Examples include, neural networks, deep learning, ensemble learning etc. Detailed review of various explainable strategies that was developed during the third generation systems is discussed in the next question.

6.2. Review of Interpretable AI methods

Methods developed to interpret the black box or opaque machine learning models can be broadly classified into two categories namely, model-agnostic and model-specific [15]. While model-agnostic methods draws the interpretations of the trained ML model by accessing the input features of the model and their respective outcomes (as predicted by the trained BB model)

(see Figure 6.3(a)), the model-specific methods draws interpretations based on the analysis of the parameters of internal components and their interaction of a specific machine learning model (e.g. weights or model coefficients in linear regression, activation of the hidden neurons and how they interact with inputs in neural networks etc. Figure 6.3(b)). In other words, model-agnostic methods provide interpretations without accessing the structure of the ML model and model-specific methods provides interpretations through the access of the structure of ML model. List of methods available under these two categories are provided in Table 6.1 [15-17].

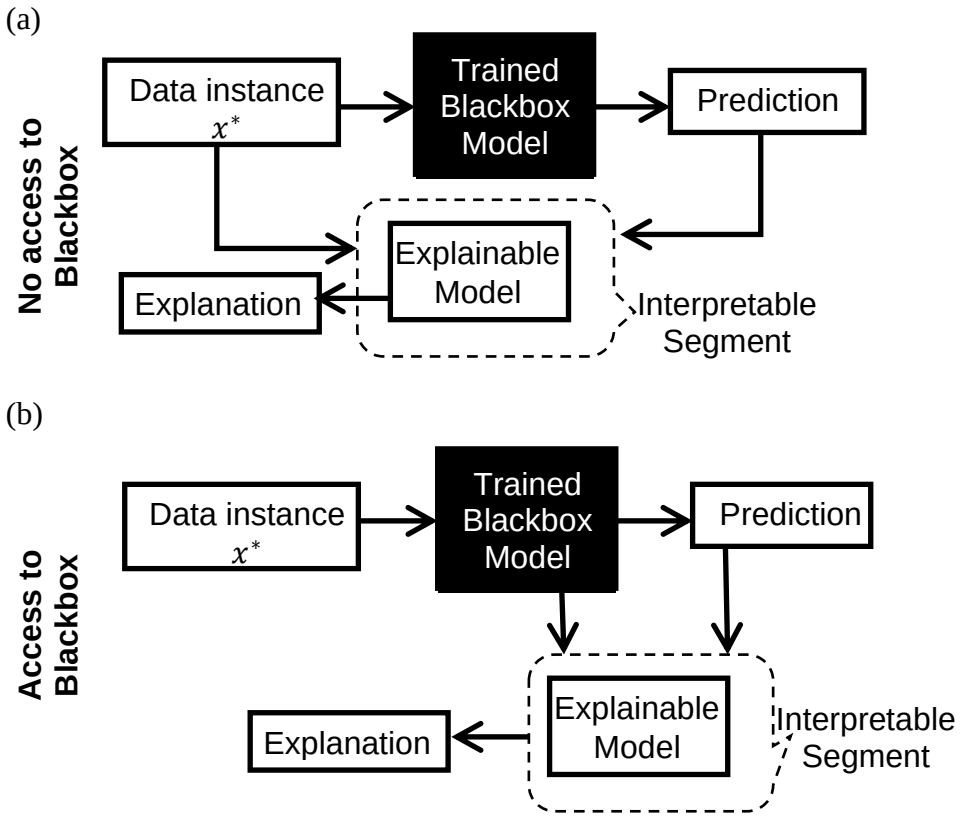


Figure 6.3. Illustration of (a) model-agnostic method and (b) model-specific method.

Table 6.1. List of interpretable methods for ML models.

Model-agnostic methods	Model-specific methods	
1. Partial dependence plots. 2. Independent component plots. 3. Accumulated local effects plot. 4. Local Interpretable Model-Agnostic Explanations (LIME). 5. Anchor plots. 6. Kernel-Shapley Additive Prediction (SHAP). 7. Sensitivity analysis. 8. Global surrogate model. 9. Local Rule-based Explanations (LORE).	Multi-layer perceptron	1. Decompositional (DAATAMINER, KT, CRED, DeepRED). 2. Pedagogical (RxREN). 3. Ecletic.
	Convolutional Neural network	1. Activation Maximization. 2. Class activation maximization (CAM). 3. Gradient Class activation Maximization (GradCAM). 4. Deconvolutional neural network (Deconvnet). 5. Guided backpropagation. 6. Deep LIFT. 7. Integrated gradients. 8. Deep Taylor decomposition. 9. Saliency. 10. Layerwise relevance propagation (LRP). 11. DeepSHAP.
	Ensemble tree	1. TreeSHAP.

6.2.1. Model-Agnostic Methods

6.2.1.1. Local Interpretable Model-Agnostic Explanation (LIME) [18]

LIME generates an intrinsically interpretable model $g(x^*)$ which is locally faithful to the trained black box model $f(x)$ i.e. it captures the behavior of the trained black box model $f(x)$ in the vicinity of an instance x^* that is being investigated. LIME model is developed based on the

premise that the features which are important in the global context may not be important in the local context. For generating a LIME interpretable model, the local behavior of the black box model f(x) is learned though a sample of artificially generated instances obtained by perturbing the instance that is being investigated (see Figure 6.4) i.e. x^*. In other words, a new sample dataset (marked as '+') is generated in the vicinity of an instance x^*, whose labels are determined from a trained black box model f(x). Note that the instances of the locally generated sample dataset are assigned weights based on their proximity to the instance x^* i.e. the instances that are far from x^* are assigned less weightage and the instances that are close to x^* are assigned more weightage. An intrinsically interpretable model such as decision tree or additive model is then obtained for the weighted sample dataset around x^*.

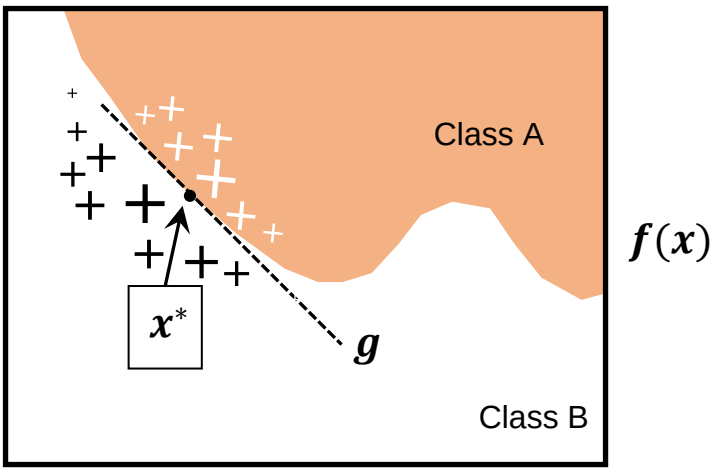


Figure 6.4. Illustration of LIME approach. The size of the markers ('+') representing the artificially generated instances vary depending on their distance from x^*.

Let $g \in G$ be considered as an interpretable model where G is a class/set of interpretable models such as linear models, decision tress, rule fit etc. If $\pi_{x^*}(z)$ denotes the proximity measure between x^* and any instance z in its vicinity (e.g. $\pi_{x^*}(z) = \exp(-d(x^*, z)/\sigma$ where d is a distance function such as cosine or Euclidean and σ is width) and $\Omega(g)$ indicates the complexity

of the g (e.g. depth of decision tree), then the LIME model is obtained by minimizing the loss function expressed as

$$\arg\min_{g \in G} L(f, g, \pi_{x^*}) + \Omega(g) \tag{6.1}$$

where $L(f, g, \pi_{x^*})$ is a measure of how faithful g is in approximating f in the locality defined by π_{x^*}.

Advantages

1. Can be used for any kind of ML model.
2. Explanations are short, selective and possibly contrastive.
3. Facilitates features mapping into more representable or understandable form. For instance, the presence or absence of pixels in the image can be represented as '1' and '0' respectively.

Limitations

1. The definition of the local neighborhood or of vicinity of the observation is not precise.
2. LIME suffers from the inclusion of unrealistic data instances when features are correlated.
3. The complexity of the explanation model has to be defined in advance.
4. Explanations can be instable between two very close points.
5. The main drawback may be related to its basic assumption. LIME, assumes linearity at locality level which may not hold true.

6.2.1.2. Anchor LIME [19]

Anchor LIME addresses one of the important limitations of LIME i.e. it determines the local neighborhood in which the explanations are valid. The explanations are then expressed as easy-to-understand IF-THEN rules, called anchors (see Figure 6.5). Anchors include two

notions that needs to be satisfied, (1) precision and (2) coverage. While precision refers to the ratio of number of times the label hasn't changed after perturbations to the number of samples satisfying the anchor, coverage refers to the space limited by the anchor (see Figure 6.5). The precision directly reflects the quality of the anchor i.e. it shows how stable the anchor is to perturbations. Finding anchors involves an exploration of input feature space which can be seen as a multi-armed bandit problem in the discipline of reinforcement learning. To this end, neighbors, or perturbations, are created and evaluated for every instance that is being explained. If multiple anchors satisfy the precision and coverage criteria, then the most global one is selected.

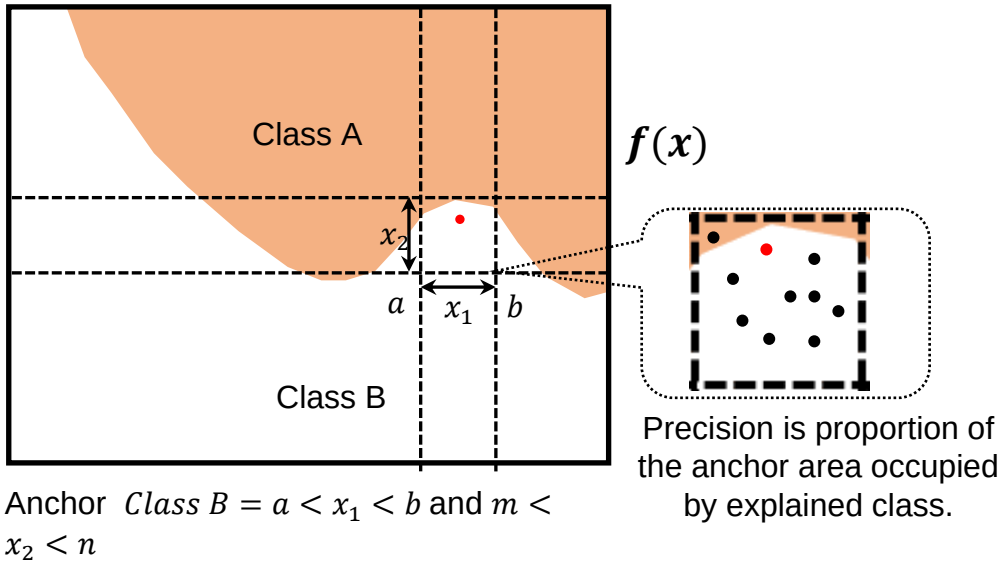


Figure 6.5. Illustration of Anchor LIME approach.

Advantages

1. Similar to LIME, anchor LIME is applicable to any kind of ML model.
2. The algorithm's output is easier to understand, as the rules are easy to interpret.
3. The anchors approach works when model predictions are non-linear or complex in an instance's neighborhood.

Limitations

1. Threshold for precision needs to be stated.
2. The process of finding anchors requires multiple calls to ML model.
3. The notion of coverage is undefined in some domains. For example, there is no obvious or universal definition of how superpixels in one image compare to such in other images [15].

6.2.1.3. Local Rule-Based Explanations (LORE) [20]

Similar to LIME and anchor LIME, LORE also learns a local interpretable predictor on an artificially generated neighborhood. However, these artificial instances are generated using a genetic algorithm. From the local interpretable predictor, the meaningful explanations are then derived in the form of decision rules. Additionally, LORE also provides a set of counterfactual rules, suggesting the changes in the instance's features that lead to a different outcome.

Definition of Local explanation: Let x^* be an instance for which explanation is sought, and $f(x^*) = y^*$ be the decision of the black box model. A local explanation $e = \langle r, \phi \rangle$ is then formulated as a pair of decision rule $r = (p \rightarrow y)$ consistent with $f(x^*)$; and a set $\Phi = \{p[\delta_1] \rightarrow \hat{y}, \ldots, p[\delta_v] \rightarrow \hat{y}\}$ counterfactual rules for p consistent with $f(x^*)$.

LORE adopts genetic algorithm to generate a balanced dataset $z \in Z_{=} \cup Z_{\neq}$, by maximizing the following two fitness functions:

$$\text{fitness}_{=}(z) = I_{b(x)=b(z)} + (1 - d(x, z)) - I_{x=z}$$

$$\text{fitness}_{\neq}(z) = I_{b(x)\neq b(z)} + (1 - d(x, z)) - I_{x=z}$$

where $d : X^m \rightarrow [0, 1]$ is a distance function, $I_{\text{true}} = 1$, and $I_{\text{false}} = 0$. The first fitness function looks for instances z similar to x (term $1 - d(x, z)$), but not equal to x (term $I_{x=z}$) for

which the black box f(x) produces the same outcome as x$\left(\text{term } I_{b(x)=b(z)}\right)$. The second one leads to the generation of instances z similar to x, but not equal to it, for which f(x) returns a different decision.

Distance Function. Distance function $d(x, z)$ in above equation is expressed as follows.

$$d(x, z) = \frac{h}{m}.\text{Match}(x, z) + \frac{m - h}{m}\text{NormEuclid}(x, z)$$

where h are categorical features.

Advantages

1. Produces a high-quality training data in the local neighborhood to learn the local decision tree.
2. A high expressiveness of the local explanations along with counterfactuals suggesting what should be different in the vicinity of the data point to reverse the predicted outcome.

Limitations

1. LORE is demonstrated only for tabular data.
2. Extension of LORE to global description of the black box is not provided.

6.2.2. Shapley Values and SHapely Additive exPlanations (SHAP) [21, 22]

Shapley values is a concept from cooperative game theory which assigns payout to each individual player depending on their contribution towards the total payout. In the context of ML explanation, features represent the players and the prediction represents total payout. Shapely values indicate the perfect decomposition of the prediction among all the features. Note that the term 'prediction' here refers to the difference between the prediction of an instance f(x) and the average model prediction E(f(X)) i.e. total payout or the prediction is equal to $f(x) - E(f(X))$. If ϕ_j denotes the Shapley value or contribution of each feature x_j, then summation of all Shapley

values is expressed as $\sum_{j=1}^{p} \phi_j = f(x) - E(f(X))$ where p is the number of features. Shapley value is the average contribution of a feature value to the prediction in different coalitions. The shapely value for a feature x_j is evaluated as

$$\phi_j = \sum_{S \subseteq \{x_1, x_2, \ldots, x_p\} \setminus x_j} \frac{|S|!\,(p - |S| - 1)!}{p!} \left(f(S \cup \{x_j\}) - f(S)\right) \tag{6.2}$$

where S is the subset of the features used in the model except x_j and $|S|$ is the number of features in the subset that is chosen except x_j. The magnitude of ϕ_j can be interpreted as the weighted mean over all subsets of features excluding x_j. The first term on the right hand side of the equation indicates the weightage from each combination of subset of features and the second term indicates the difference in the predicted value when x_j is included with subset of features S. For the sake of illustration, let $p = 3$ and the features of an instance be given by $x = (x_1, x_2, x_3)$. If the contribution of x_3 is to be evaluated, then the total combinations of features excluding x_3 are 2^{3-1} i.e. $\emptyset, \{x_1\}, \{x_2\}, \{x_1, x_2\}$. Note that when only one feature is chosen in subset S then $|S| = 1$ and when two features are chosen then $|S| = 2$ and so on. The Shapley value for x_3 is then computed as

$$\phi_3 = \frac{1}{3}\left(f(x_1, x_2, x_3) - f(x_1, x_2)\right) + \frac{1}{6}\left(f(x_1, x_3) - f(x_1)\right) + \frac{1}{6}\left(f(x_2, x_3) - f(x_2)\right) + \frac{1}{3}\left(f(x_3) - f(\emptyset)\right)$$

Shapley values can be both positive and negative. Higher positive Shapley value indicates higher positive contribution and higher negative Shapley value indicates higher negative contribution to the prediction of an instance when compared to the average prediction for the dataset.

Limitations

1. For more than a few features, number of possible coalitions exponentially increases as more and requires a lot of computing time.
2. Explanations created with the Shapley value method always use all the features.
3. Access to the data is required to calculate the Shapley value for a new data instance.
4. Similar to other methods, the Shapley value method suffers from inclusion of unrealistic data instances when features are correlated.

6.2.2.1. Kernel SHAP

Kernel SHAP is an additive feature attribution method which explains the prediction of the BB model for an instance x^* by decomposing the prediction into the respective feature contributions. Mathematically this is expressed as

$$f(x^*) = \phi_0 + \sum_{i=1}^{p} \phi_i^* \tag{6.3}$$

Where $\phi_0 = E[f(x)]$ is the global average prediction and ϕ_i^* is the Shapley value for a feature x_i in x^*. Higher Shapley value of a feature indicates its significance towards the prediction.

To compute ϕ_i^* Kernel SHAP carries out minimization of weighted least squares formulation

$$\sum_{S \subseteq \{x_1, x_2, \ldots, x_p\}} \left(f(S) - \left(\phi_0 + \sum_{i=1}^{p} \phi_i \right) \right)^2 K(p, S) \tag{6.4}$$

with respect to $\phi_0, \phi_1, \ldots, \phi_p$, where $K(p, S) = \frac{p-1}{\binom{p}{|S|} |S|(p-|S|)}$ are the Shapley kernel weights.

Given f(S), the computation of Shapley value vectors $\phi = (\phi_0, \phi_1, \ldots, \phi_p)$ becomes expensive and intractable when the number of features increases resulting in more number of combinations of features. Therefore, assuming that we have a proper approximations for f(S) when few of the

Shapley kernel weights are ignored due to their little contribution to Shapley values, only a subset of features can be chosen instead of all 2^p combinations and the computation effort can be reduced without compromising the accuracy. The subset of features is chosen such that they follow the probability distribution of Shapley weight kernel. Note that $|S| = \emptyset$ and $|S| = p$ are avoided as they result in infinite kernel weights (i.e. $K(p,0) = K(p,p) = \infty$). The magnitude of f(S) for all the possible feature subsets is evaluated as

$$f(S) = E[f(x)|x_S = x_S^*]$$

i.e. the selected feature subset values x_S^* is substituted into the trained model f(x) and the expected value is evaluated. However, the model f(x) needs input of all features for valid computation i.e. the complement of subset S, denoted by $\bar{S}$ is missing in the above equation. To resolve the missing features issue, the marginalized features are used in the place of $\bar{S}$ by assuming feature independence and is obtained from training data sample as

$$f(S) = \frac{1}{N}\sum_{i=1}^{N} f(x_{\bar{S}}^i, x_S^*)$$

Limitations

1. Kernel SHAP is computationally slow.
2. Kernel SHAP ignores the feature dependence.

6.2.3. Partial Dependence Plots (PDP) [23, 24]

The partial dependence plot (short PDP or PD plot) reveals the marginal effect of one or two features have on the predicted outcome of a machine learning model. A partial dependence plot can show whether the relationship between the target and a feature is linear, monotonic, or more complex. Mathematically it can be expressed as

$$f(x_S) = E[f(x_{\bar{s}}, x_S)] = \int p(x_{\bar{s}}) f(x_{\bar{s}}, x_S) dx_{\bar{s}} \quad (6.5)$$

where x_S are the set of features for which dependence of prediction is examined, $x_{\bar{s}}$ denotes the set of features not included in x_S, $p(x_{\bar{s}})$ marginal distribution of $x_{\bar{s}}$. For a given training data, the partial dependence of prediction on subset of features x_S is practically evaluated as

$$f(x_S) = \frac{1}{N}\sum_{i=1}^{N} f\left(x_{\bar{s}}^{i}, x_S\right)$$

Where $x_{\bar{s}}^{i}$ denotes the actual feature values of set $\bar{S}$ from the training dataset containing $i = 1,2, \ldots, N$ samples.

Advantages and Limitations

1. Procedure is intuitive and interpretations are clear.
2. It is easy to implement

Limitations

1. Assumes that the features are independent.
2. Any heterogeneous effects cannot be detected.
3. Providing visualization for more than two features is difficult.

6.2.4. Accumulated Local Effects (ALE) [15, 25]

Similar to partial dependence plots, ALE also evaluates the influence of the feature x_j on the prediction $f(x^*)$. However, ALE additionally considers the lower-order interaction effects of the pair of features i.e. x_j vs every other feature in x^*. Lower-order interactions reduces the bias of the estimated feature effect and avoids the average predictions of the artificial data instances that are unlikely in reality which arises when the pair of features are strongly correlated. For instance, consider the plot shown in Figure 6.6(a). Two features x_1 and x_2 are positively correlated. The marginal distribution plot for x_2 reveals the inclusion of unlikely combinations of x_1 and x_2. To avoid the inclusion of unlikely data instances a conditional distribution is

considered instead of marginal distribution (see Figure 6.6(b)). However, the mixed effect of correlated features would still prevail i.e. the dependence of prediction on feature x_j alone is not completely obtained unless it is decoupled from the other feature it is correlated with.

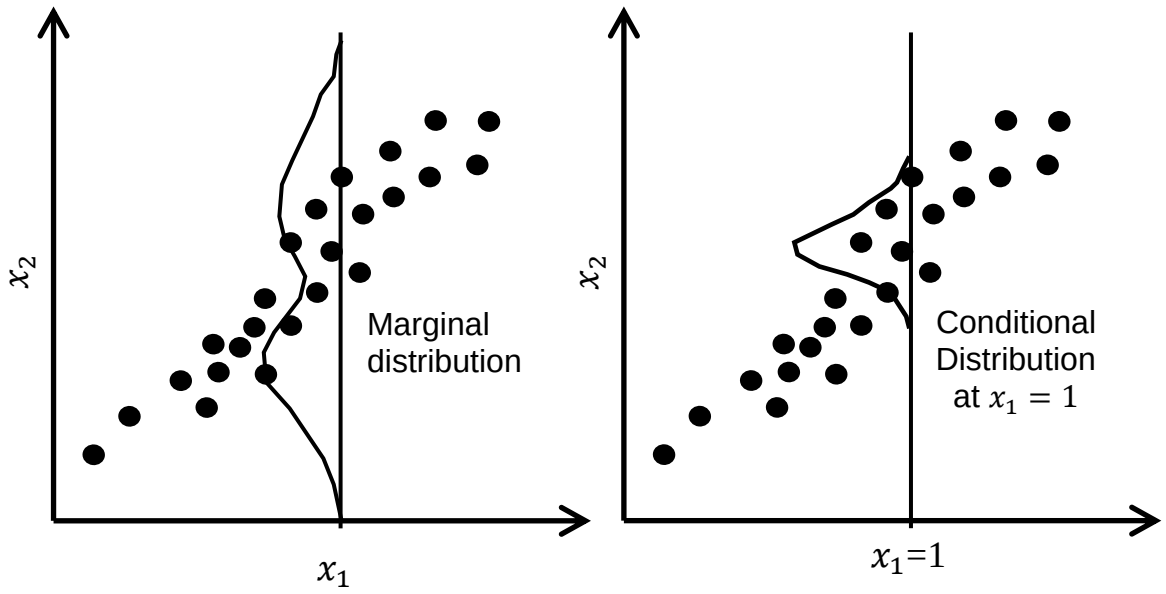


Figure 6.6. (a) marginal distribution and (b) conditional distribution of x_2 for positively correlated features x_1 and x_2.

ALE solves the problem of mixed effects by averaging the differences in predictions of $f(x^*)$ obtained in small interval of the feature x_j while holding the other feature value constant i.e. consider if $x_1$and x_2 are the two features and the effect of x_1 on $f(x_1, x_2)$ is to be investigated, then the difference of predictions $f(x_1 + \delta, x_2) - f(x_1 - \delta, x_2)$ is evaluated for all x_2 values falling in the interval $[x_1 - \delta, x_1 + \delta]$ and then averaged. Mathematically this is expressed as

$$\mathrm{ALE}(x_1) = \int_{x_1-\delta}^{x_1+\delta} \mathrm{E}\left[\frac{\partial f(x_1, x_2)}{\partial x_1} \,\middle|\, x_1 = z_1\right] dz_1 - C_1 \tag{6.6}$$

where $\frac{\partial f(x_1,x_2)}{\partial x_1}$ indicated the local effect of x_1 on f, integration signifies the accumulation and constant C_1 is chosen such that the ALE values are independent of the bounds and have zero

mean over the distribution of x_1 i.e. first term on the right hand side of equation is uncentered ALE. The above formulation can also be easily extended to interaction of two features with other features. However, the visualization becomes intractable as the number of dimensions increase.

Advantages

1. ALE plots are unbiased and works when features are correlated.
2. ALE plots are faster to compute than PDPs.

Limitations [33]

1. ALE plots may appear have noise and non-smooth with a high number of intervals. However, reducing the number of intervals smoothens out the complexity of the prediction model.
2. There is no perfect solution for setting the number of intervals. If the number is too small, the ALE plots might not be very accurate. If the number is too high, the curve can become shaky.
3. The implementation of ALE plots is much more complex and less intuitive compared to partial dependence plots.
4. Even though ALE plots are not biased in case of correlated features, interpretation remains difficult when features are strongly correlated. Because if they have a very strong correlation, it only makes sense to analyze the effect of changing both features together and not in isolation.

6.2.5. Individual Conditional Expectational Plot (ICE) [26]

ICE plots provide the visualization of influence of the features on the prediction i.e. it addresses the question of how prediction changes with change in the features. However, the basic difference between PDP and ICE plots is that the ICE plots include the display for each and

every individual instance instead of averaging them over all the data instances i.e. the average of all the lines in ICE plots is equivalent to a PDP plot. To obtain an ICE plot, each instance of the dataset is chosen separately, and the input of a particular feature is varied to examine its influence on the prediction while holding other features constant. For instance, if there are N data instances then an ICE plot for a particular feature x_j includes N ICE lines exhibiting the behavior of each instance in the dataset. ICE plots help to explore individual differences and interactions between model inputs.

$$f^i(x_S) = f\left(x_{\bar{S}}^i, x_S^i\right)$$

where $i = 1,2, \ldots, N$ and $f^i(x_S)$ is plotted against x_S^i while $x_{\bar{S}}^i$ is fixed.

Advantages and Limitations

1. Individual conditional expectation curves are even more intuitive to understand than partial dependence plots. One line represents the predictions for one instance if we vary the feature of interest.
2. Unlike partial dependence plots, ICE curves can uncover heterogeneous relationships.

Limitations

1. Similar to PDP, ICE curves also suffer from the problem of including unrealistic data in the case of strongly correlated features.
2. In ICE plot the curves become overcrowded when the data instances increase and may be difficult to grasp.
3. In ICE plots it might not be easy to see the average. However, it can be combined with PDP.

6.2.6. Sensitivity Analysis [27, 28]

Sensitivity analysis is a method that measures the change in the output of a given model when the input is varied through the range of value. Based on the amount of change in the output, the input features are assigned ranks. Commonly used sensitivity measures for continuous outputs are range (S_r), gradient (S_g), average absolute deviation (S_d) and variance (S_v).

$S_r = \max\left(\hat{y}_{a_j}\right) - \min\left(\hat{y}_{a_j}\right)$ where $j \in \{1,2,..,l\}$ and l are the levels of an input feature x_a.

$S_g = \sum_{j=2}^{l} \frac{\left(|\hat{y}_{a_j} - \hat{y}_{a_{j-1}}|\right)}{l-1}$

$S_v = \sum_{j=1}^{l} \frac{\left(\hat{y}_{a_j} - \bar{y}_a\right)^2}{l-1}$ where $\bar{y}_a$ is the mean of the response.

$S_d = \sum_{j=1}^{l} \frac{\left(|\hat{y}_{a_j} - \tilde{y}_a|\right)}{l}$ where $\tilde{y}_a$ is the median of the response.

Higher the measure, higher is the importance of the of the input feature. For the sake of convenience relative importance of feature is evaluated and are plotted as bar plots for visualization.

$$r_a = \frac{\varsigma_a}{\sum_{i=1}^{N} \varsigma_i} \tag{6.7}$$

where ς_a is the sensitivity measure of x_a.

6.2.7. Model-Specific Interpretable Methods (Neural Networks and CNN)

6.2.7.1. Deconvolutional Neural Network (deconvnet) [29]

Deconvnet is a visualization technique that provides insight into the function of the intermediate feature layers. In other words, they reveal the properties of the input image learned by each layer. For this, Deconvnet maps the activations of intermediate feature layers back to the input pixel space. It is similar to the convolutional neural network operation which takes place in

reverse order. For instance, in convolution neural networks first the convolution is carried out with learned set of filters and then the responses are recorded by passing the convoluted values through rectified linear units and then max pooling operation is performed. In deconvent, following the reverse order, first unpooling is performed, then the rectification is carried out and then finally deconvolution operation is implemented to obtain the reconstructed image. Note that unpooling is non-invertible. An approximate inverse of unpooling is obtained by substituting the values in the locations from which the maximum values were extracted during pooling operation. In other words, unpooling results in a sparse matrix.

Due to zeroing out of negative gradients during backpropagation, deconvnet fails to highlight the inputs that negatively contributes the output. Additionally, the saturation problem and zero gradient problem prevails.

6.2.7.2. Layer-Wise Relevance Propagation (LRP) [30-33]

LRP explains the prediction of an instance obtained from the trained neural network by back tracking its associated relevance score (R) to the input neurons layer-by-layer i.e. it identifies the most relevant input features that attributed to the final outcome $y^* = f(x^*)$. Generally, the activation value of the output neuron is chosen as the relevance score. Let R_j denote the relevance score of the neuron j in the succeeding layer k and R_i denote the relevance score of a neuron i in the preceding layer $k-1$ that is connected to neuron j through weight parameter w_{ij}. If there are n neurons in the $k-1$ layer and m neurons in k layer, then the relevance score $R_{i=1..n}$ of each neuron in $k-1$ layer is evaluated as

$$R_i = \sum_j \frac{a_i w_{ij}}{\sum_i a_i w_{ij}} R_j \tag{6.8}$$

where a_i is the magnitude of the activation function of neuron i, w_{ij} is the weight parameter connecting neuron i and j. Here the summation on j indicates the number of neurons in the layer k that are connected with neuron i (see Figure 6.7) and the coefficient of R_j determines the proportion of weight contribution between neuron i and j. The above equation is also referred to as LRP-0 rule. Taking into account the contribution from positive and negative parts of the weight parameters, a generic LRP rule called as LRP αβ – rule is also provided in [34]

$$R_i = \sum_j \left(\alpha \frac{a_i w_{ij}^+}{\sum_i a_i w_{ij}^+} - \beta \frac{a_i w_{ij}^-}{\sum_i a_i w_{ij}^-} \right) R_j \tag{6.9}$$

Where α and β are subjected to constraints $\alpha - \beta = 1$ and $\beta \geq 0$.

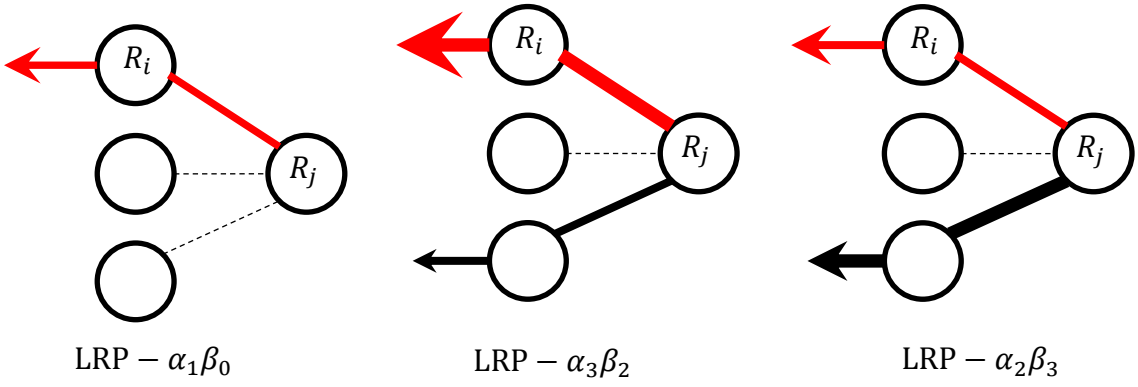


Figure 6.7. Variations of Layer wise Relevance Propagation rules (Adapted from [34]).

6.2.7.3. Deep Taylor Decomposition (DTD) [34, 35]

Similar to LRP, deep Taylor decomposition also redistributes the relevance score of the prediction to the input features. However, in DTD the redistribution rules for the relevance score is determined based on the Taylor series decomposition. If f(x) is a nonlinear function, then the Taylor decomposition at well-chosen root point $\tilde{x}$ i.e. $f(\tilde{x}) = 0$ is expressed as

$$f(x) = f(\tilde{x}) + \left(\frac{\partial f}{\partial x}\Big|_{x=\tilde{x}}\right)^{T} . (x - \tilde{x}) + \varepsilon$$
$$= 0 + \sum_{i} \frac{\partial f}{\partial x_i}\Big|_{x_i=\tilde{x_i}} . (x_i - \tilde{x_i}) + \varepsilon \tag{6.10}$$

where i denotes the input features and the summation term indicates the relevance score of f(x). Based on the above equation, the relevance score R_i of a neuron i in the preceding layer $k-1$ obtained from the relevance scores R_j of the neuron j the succeeding layer k during back propagation is as follows

$$R_i = \sum_{j} \frac{\partial R_j}{\partial x_i}\Big|_{(\tilde{x_i})^j} . \left(x_i - (\tilde{x_i})^j\right)$$

A good root point is the one that does not consist of the information of an object in the input features that belong to the prediction class. To determine a root point $\tilde{x_i}$ at the neuron j, a w^2-rule is proposed in [35], which is expressed as

$$(\tilde{x_i})^j = x_i - \frac{w_{ij}}{\sum_i w_{ij}^2}\left(\sum_i w_{ij} + b_j\right) \tag{6.11}$$

The relevance redistributed on to neuron i is then evaluated as

$$R_i = \sum_{j} \frac{w_{ij}^2}{\sum_i w_{ij}^2} R_j \tag{6.12}$$

6.2.7.4. Saliency Maps [36]

Saliency map is a visualization technique that aids in interpreting the prediction of the neural network model by providing an insight into the salient features that are responsible for the prediction. For this, the prediction of the model is mapped back to input feature space where the salient features are highlighted i.e. pixels in the case of an image can be displayed as a heatmap. Given an image I_o and the prediction made by the trained convolutional neural network through

a class score function $S_c(I)$ for an image I, let the score function $S_c(I)$ which is differentiable can be approximated in the neighborhood of I_o as

$$S_c(I) \approx w^T I + b \tag{6.13}$$

where w is the derivative of S_c with respect to an image I at I_o i.e. $w = \frac{\partial S_c}{\partial I}\Big|_{I_o}$ and is interpreted as the weightage of the pixel in the class prediction. For evaluating weights w backpropagation algorithm is implemented. The evaluated weights are then converted into heat map by choosing only the absolute value of the weights. Shrikumar et. al. [37] suggested to use gradient × pixel value instead of highlighting only the weights alone. Saliency maps are often noisy due to saturation and discontinuous gradients on ReLU.

6.2.7.5. Guided Backpropagation [38]

While the negative gradient values are taken into consideration in the backward pass of the relevant score in the backpropagation technique, they are set to zero in the guided back propagation technique. In other words, Guided back propagation technique combines both deconvnet and the saliency map when handling ReLU nonlinearity.

$$R_i^l = \left(f_i^l > 0\right).\left(R_i^{l+1} > 0\right).R_i^{l+1} \tag{6.14}$$

where R_i^l and R_i^{l+1} are the relevance scores in layer l and $(l+1)$, and f_i^l is the ReLU function of neuron i in layer l.

6.2.7.6. Deep Learning Important FeaTures (Deep LIFT) [37, 39]

Similar to saliency, deconvnet and guided back propagation techniques, Deep LIFT also relies on back propagation approach. However, it employs the difference in activations from the chosen reference value instead of using direct gradients. If Δt denotes the difference in the target value due to the difference in the input value Δx (can be the neurons from preceding layer), then the contributions of the input neurons are assigned to the difference in target value such that

$\sum_{i=1}^{n} C_{\Delta x \Delta t} = \Delta t$. Authors of Deep LIFT have also defined a term referred to as 'multipliers' which indicates change in t due to infinitesimal change in x and is denoted as $m_{\Delta x \Delta t} = \frac{C_{\Delta x \Delta t}}{\Delta x}$. Applying chain rule to the multipliers that connect neurons in one layer to the other layer, backpropagation can be easily carried out. For assigning contribution scores, three different rules are proposed namely, linear rule, rescale rule and reveal cancel rule. While linear rule is employed for the linear function maps on input to output, the rest of the two rules are employed for nonlinear functions. In the case of linear rule the multipliers are directly determined as the weights connecting the neurons and in the case of rescale rule the multipliers are the ratios of change in the output neuron to the change in the input neuron. To evaluate the contribution of each input feature, the multipliers along the network path are multiplied using chain rule and then combined with change in the value of the input feature.

6.2.7.7. Integrated Gradients [40]

Unlike Deep LIFT technique which considers the direct difference of the target value and input value from their respective reference or baseline value for evaluating the contributions of input features, integrated gradients technique constructs the difference in step by step manner by scaling the input linearly from the reference value. For instance, consider an image classification problem and let the reference image be a black image. The difference between the original image and reference image is interpolated into series of images such that the pixel intensities vary linearly from reference image to original image. The gradients are calculated for all the series of the images and then averaged to obtain the integrated gradient image.

$$IG(x) = (x - x') \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha(x - x'))}{\partial x_i} d\alpha \qquad (6.15)$$

6.2.7.8. Class Activation Mapping (CAM) [41]

CAM and Grad-CAM techniques identifies the important features or dominant regions of the input image which activates the neuron in the output layer associated with a class label y_c. For this purpose, first the weights in the output layer are projected back on to each feature map of the last convolutional layer. The weighted feature maps are then combined to obtain the heat maps referred to as class activation maps. However, CAM and Grad-CAM differ in their approach. In CAM, the CNN architecture is tweaked to replace the fully connected layers with the Global Averaging Pooling (GAP) layer (see Figure 6.8). GAP layer consists of neurons representing the feature maps from the last convolutional layer wherein the magnitude of each neuron is obtained by calculating the average of each feature map. In other words, if there are k feature maps of size $u \times v$ in the last convolution layer, then the GAP layer will also consists of k neurons where the magnitude of each neuron is evaluated as $\frac{1}{uv}\sum_{i=1}^{u}\sum_{j=1}^{v} A_{ij}^{k}$ where A^k denotes the kth feature map and the subscript ij indicates the (i,j)th pixel in the feature map. The tweaked CNN architecture is then retrained to obtain the weights of the synapses connecting the GAP layer and the output layer. If $w_1^c, w_2^c, \ldots, w_k^c$ are the weights of the synapses associated with the neurons in the output layer, then the score y_c of each neuron in output layer is expressed as

$$y_c = \sum_{l=1}^{k} w_i^c \frac{1}{uv} \sum_{i=1}^{u}\sum_{j=1}^{v} A_{ij}^{l} \tag{6.16}$$

Each feature map is then multiplied by the weights associated with the predicted class label y_c and are combined. CAM is expressed as

$$\text{CAM} = \sum_{i=1}^{k} w_i^c A^i \tag{6.17}$$

However, the heatmaps obtained through CAM have limitations. Since the fully connected layers are removed from the CNN architecture, the model performance is compromised. Besides this the model has to be retrained for the determination of weights in the GAP layer.

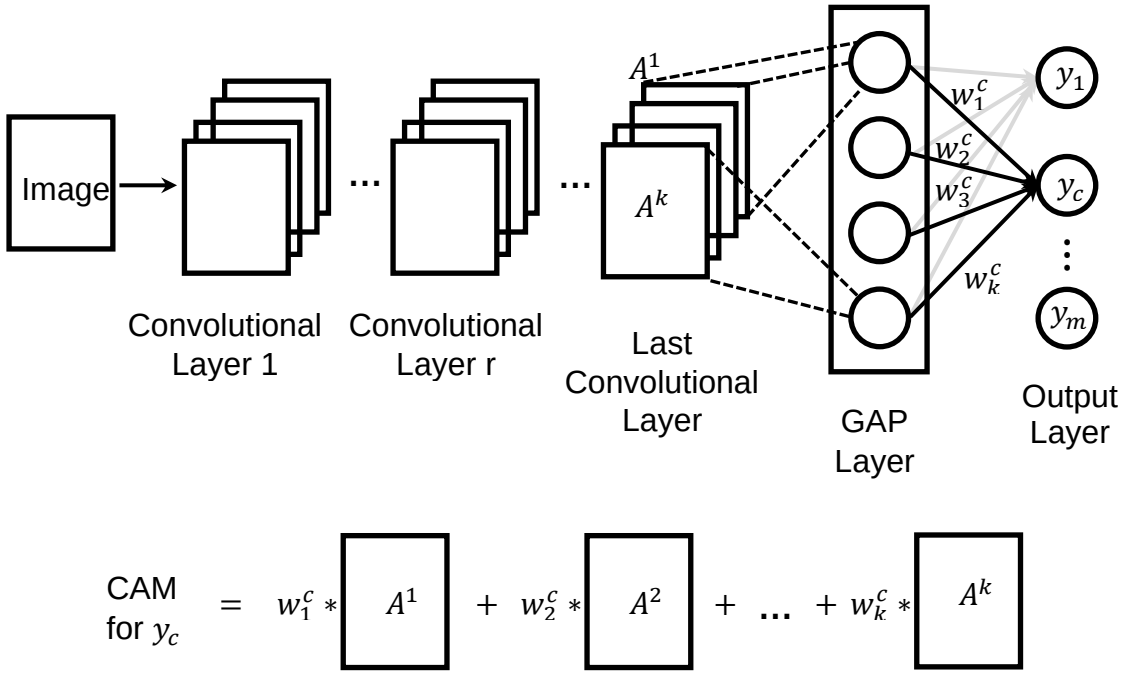


Figure 6.8. The schematic of class activation mapping (CAM).

6.2.7.9. Gradient-Class Activation Mapping (Grad-CAM) [42, 43]

Grad-CAM addresses the limitations of CAM by (1) retaining the fully connected layers in the CNN architecture and (2) employing the gradients of class score y_c with respect to the feature map A^k i.e. it incorporates $\frac{\partial y_c}{\partial A^k}$. Note $\frac{\partial y_c}{\partial A^k}$ that can be evaluated through backpropagation. The schematic of Grad-CAM is shown in Figure 6.9. Unlike CAM, no GAP layers are explicitly used in Grad-CAM. However, the global average pooling technique is employed on gradient feature maps to obtain the importance weights α_c^k i.e.

$$\alpha_c^k = \frac{1}{uv}\sum_{i=1}^{u}\sum_{j=1}^{v}\frac{\partial y_c}{\partial A_{ij}^k} \tag{6.18}$$

To obtain the final heat map, Grad-CAM evaluates the weighted combination of feature maps and passes it through ReLU.

$$\text{GradCAM} = \text{ReLU}\left(\sum_{i=1}^{k}\alpha_c^i A^i\right) \tag{6.19}$$

Drawbacks of Grad-CAM include inability to localize multiple occurrences of an object in an image and inaccurate localization of heatmap with reference to coverage of class region due to the partial derivatives premise. The continual upsampling and downsampling processes may also result in loss of signal.

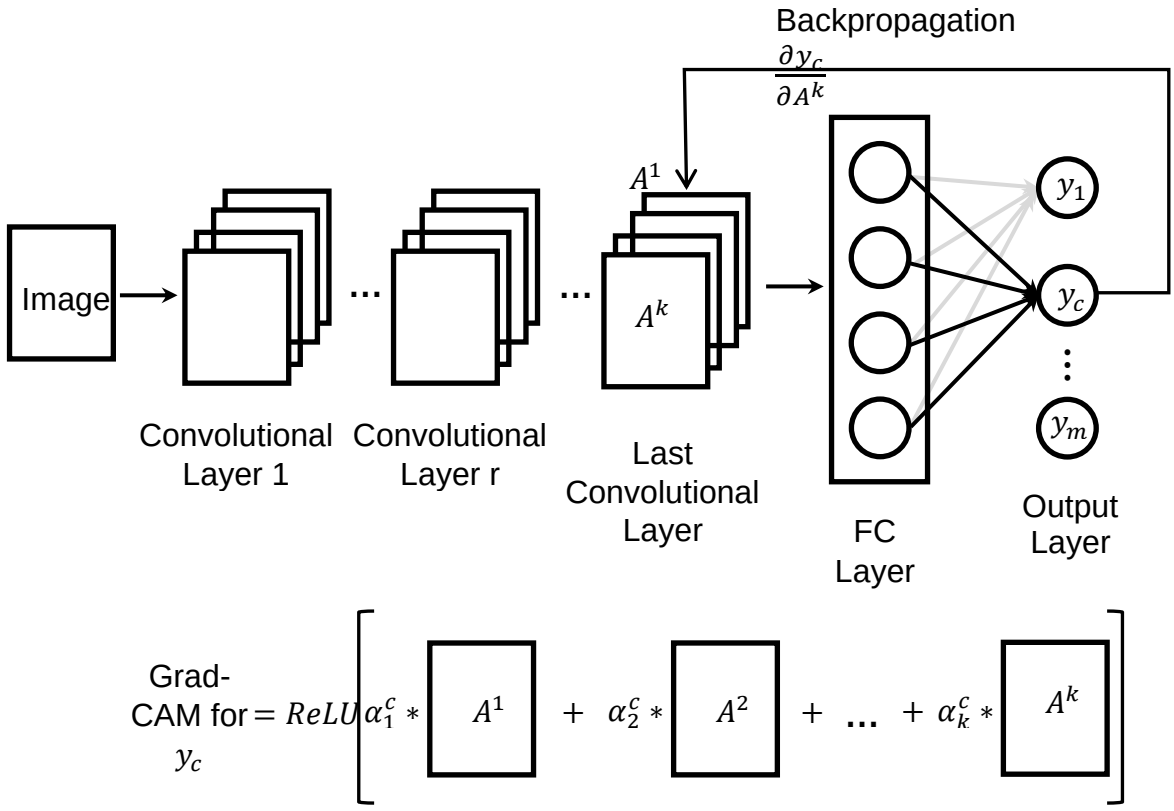


Figure 6.9. The schematic of Gradient-class activation mapping (Grad-CAM).

6.2.8. Shallow and Deep Neural Networks

Various rule extraction techniques for shallow neural networks with at most one or two hidden layers have been proposed in late 90's. These rules were either in "IF-THEN" form, or "M-OF-N rules" form or a decision tree. Collectively, these techniques could be put into three categories namely decompositional algorithms, pedagogical algorithms, and eclectic algorithms [44]. Decomposition algorithms work at neuron/layer level of the network and then later aggregates the rules from all the neurons/layers to represent the whole network. Pedagogical approach extracts the rules by mapping input to its corresponding output rather than working on the internals of the network. Eclectic approaches combine both decompositional and pedagogical approach. A list of techniques falling under these categories is provided in Table 6.2. Among the listed techniques, only Deep RED (extension of CRED) and RxREN are briefed herein as they extend to deep neural networks and rest of techniques are restricted to shallow networks.

Table 6.2. List of rule extraction algorithms for neural networks [44].

Algorithm Type	Algorithms	Type of rule extraction
Decompositional	KT [45]	IF-THEN
	CRED [46]	Decision Tress
	DIFACON-Miner [47]	IF-THEN
	Tsukimoto's Algorithm [48]	IF-THEN
	FRENN [49]	M-of-N rules
	Deep RED [50]	Decision Tree
Pedagogical	ANN-DT [51]	Binary decision tree
	HYPINV [52]	Hyperplane rule
	TREPAN [53]	M-of-N split, Decision tree
	BIO-RE [54]	Binary rule
	RxREN [55]	
Eclectic	Rx [56]	IF-THEN
	Kahramanli and Allahverdi's algorithm [57]	IF-THEN

6.2.8.1. Deep RED

Deep RED is an extension of CRED algorithm [46] that extracts both continuous and discrete rules from the trained neural network using decision tree (C4.5). While CRED included rule extraction for a shallow network consisting of one hidden layer, Deep RED extracts rules for the deep network consisting of multiple hidden layers. Deep RED involves two phases, (1) extracting the rules between each layer in the network and (2) substituting and merging all rules. In the first phase, two successively connected layers are chosen iteratively (in the direction of output layer to input layer) and the decision tree is built in each iteration such that the hidden neurons in preceding layer acts as input features and the neurons in the succeeding layer acts as an output unit. Let the rules obtained in each iteration be denoted by $R_{a\to b}$ where a represents the preceding layer and b represents the succeeding layer. For instance, if a neural network consists of $h_1, h_2, .., h_k$ hidden layers and let i be the input layer and o be the output layer, then the rules obtained from decision tress between layers h_k and o is denoted as $R_{h_k\to o}$. The next set of rules are extracted for $R_{h_{k-1}\to h_k}$ and the process is repeated until $R_{i\to h_1}$. All the rules from the hidden layers are then merged to obtain $R_{i\to o}$ such that redundant rules are omitted.

Although Deep RED provides comprehensible rules from deep neural networks for binary classification, further research is required to gauge its performance for multi class classification. Also, the influence of decision tree parameters needs to be thoroughly investigated.

6.2.8.2. Rule Extraction by Reverse Engineering the Neural Networks (RxREN)

RxREN is a pedagogical algorithm i.e. it maps the input to the output without accessing the internals of the neural network architecture. RxREN algorithm constitutes two phases: (1) removal of insignificant input neurons from the trained network and obtaining a data range for

significant input neurons and, (2) constructing the rules for each class using the data ranges of significant input neurons. To identify the insignificant input neurons, the number of misclassified instances resulting from the deletion of one input neuron at a time from the trained neural network is evaluated. A threshold criterion is then set which indicates the number of misclassified instances and the input neurons satisfying this criterion are removed from the network. The pruned network with significant neurons is then re-executed to verify the accuracy. Upon satisfying the accuracy, the instances of the training dataset are grouped based on their class label for each significant input neuron. The minimum and maximum values of each significant input neuron are then determined for each class label. For instance, let i_2 be a significant neuron and let the number of class labels be 3. Then each class will consist of minimum i_2 value and maximum i_2 value which will be obtained (see Figure 6.10). Based on the extracted minimum and maximum values for significant input neuron and its associated class label, the rules are extracted.

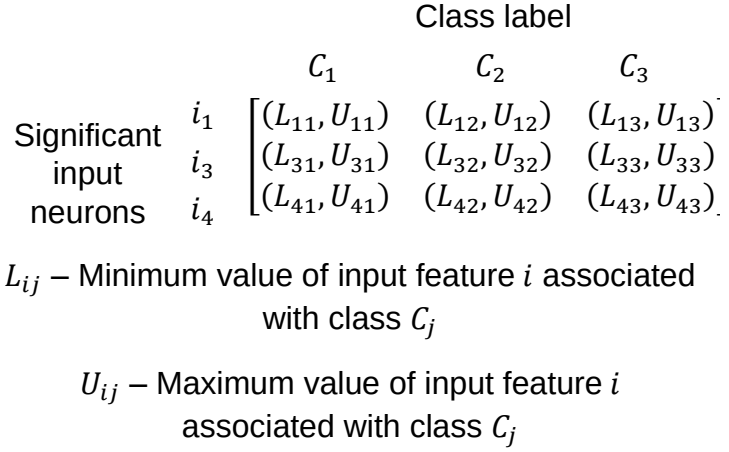


Figure 6.10. Significant neurons in the network and range of values associated with each class.

6.2.9. Potential Research Gaps in XAI

Following research gaps are identified in the field of XAI.

1. Most of the existing approaches in deep learning rely on visualization techniques to identify or highlight the important features in the image. However, visualization maps alone might be insufficient for the purpose of interpretation or explanation since human bias might hinder the proper use of XAI in mission-critical applications. In other words, the interpretations obtained from visualization maps needs to be presented in more elaborated form rather than qualitatively.
2. Explanations of models using techniques like Deep LIFT, Integrated gradients and Deep Taylor Decomposition rely on the reference point. Choosing a wrong reference point might result in misleading explanations. Therefore, a general approach for identifying the representative reference point is needed for generating reliable and consistent explanations.
3. Metrics for quantitatively measuring the explanations is unavailable. For instance, the performance of ML model can be quantified using ‘accuracy’, ‘precision’, ‘F-measure’ etc.
4. Often the bias terms are ignored during the interpretation of deep learning models e.g. saliency. According to Wang et. al.[58] the bias terms may have strong attribution towards the outcome.
5. Debugging of deep neural networks to remove insignificant neurons and improve the architecture is not extensively studied. Although RxREN removes the insignificant neurons from the neural networks, the approach is relied on training dataset.

Incorporating the new instances through perturbation might guide in developing more optimal neural network architecture.

6. Most of the interpretable models does not consider feature interaction. Incorporating techniques like Accumulated Local Effects along with the kernel SHAP, LIME may provide better explanations.
7. Finite difference schemes are employed currently for sensitivity analysis. However, finite difference schemes prone to subtractive cancellation errors [19,20] (see Chapter 7). Subtractive cancellation errors are caused by subtracting two close numbers whose difference could be in the order of the precision of the calculations.

6.3. Scope of the Current Research in XAI

In this dissertation, the limitation of sensitivity analysis is addressed. A novel algorithm is proposed that eliminates the subtractive cancellation errors. Specifically, the proposed method is implemented in the framework of deep neural networks.

6.4. References

1. Mueller ST, Hoffman RR, Clancey W, Emrey A, Klein G. Explanation in human-AI systems: A literature meta-review, synopsis of key ideas and publications, and bibliography for explainable AI. arXiv preprint arXiv:190201876. 2019.
2. Feigenbaum EA. Themes and case studies of knowledge engineering. Expert systems in the micro-electronic age. 1979:3-25.
3. Shortliffe EH. MYCIN: a rule-based computer program for advising physicians regarding antimicrobial therapy selection. Stanford Univ Calif Dept of Computer Science; 1974.
4. Swartout WR. A digitalis therapy advisor with explanations. Proceedings of the 5th international joint conference on Artificial intelligence- 21977. 819-25.

5. Swartout WR. XPLAIN: A system for creating and explaining expert consulting programs. Artificial intelligence. 1983;21:285-325.
6. Weiner J. BLAH, a system which explains its reasoning. Artificial intelligence. 1980;15:19-48.
7. Nikolopoulos C. Expert systems: introduction to first and second generation and hybrid knowledge based systems: Marcel Dekker, Inc.; 1997.
8. Keravnou E. What is a deep expert system? An analysis of first-generation limitations and a review of second-generation architectures. Computer Physics Communications. 1990;61:3-12.
9. Keravnou ET, Washbrook J. What is a deep expert system? An analysis of the architectural requirements of second-generation expert systems. The Knowledge Engineering Review. 1989;4:205-33.
10. Davis R. Reasoning from first principles in electronic troubleshooting. International Journal of Man-Machine Studies. 1983;19:403-23.
11. Clancey WJ, Letsinger R. NEOMYCIN: Reconfiguring a rule-based expert system for application to teaching: Department of Computer Science, Stanford University Stanford; 1982.
12. Bylander T, Chandrasekaran B. Generic tasks for knowledge-based reasoning: the "right" level of abstraction for knowledge acquisition. International Journal of Man-Machine Studies. 1987;26:231-43.
13. Swartout W, Paris C, Moore J. Explanations in knowledge systems: Design for explainable expert systems. IEEE Expert. 1991;6:58-64.
14. Steels L. The deepening of expert systems. AI Communications. 1987:9-16.

15. Molnar C. Interpretable Machine Learning; 2020.
16. Arrieta AB, Díaz-Rodríguez N, Del Ser J, Bennetot A, Tabik S, Barbado A, et al. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion. 2020;58:82-115.
17. Carvalho DV, Pereira EM, Cardoso JS. Machine learning interpretability: A survey on methods and metrics. Electronics. 2019;8:832.
18. Ribeiro MT, Singh S, Guestrin C. " Why should I trust you?" Explaining the predictions of any classifier. Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining 2016. 1135-44.
19. Ribeiro MT, Singh S, Guestrin C. Nothing else matters: model-agnostic explanations by identifying prediction invariance. arXiv preprint arXiv:161105817. 2016.
20. Guidotti R, Monreale A, Ruggieri S, Pedreschi D, Turini F, Giannotti F. Local rule-based explanations of black box decision systems. arXiv preprint arXiv:180510820. 2018.
21. Lundberg S, Lee S-I. An unexpected unity among methods for interpreting model predictions. arXiv preprint arXiv:161107478. 2016.
22. Aas K, Jullum M, Løland A. Explaining individual predictions when features are dependent: More accurate approximations to Shapley values. arXiv preprint arXiv:190310464. 2019.
23. Friedman JH. Greedy function approximation: a gradient boosting machine. Annals of statistics. 2001:1189-232.
24. Zhao Q, Hastie T. Causal interpretations of black-box models. Journal of Business & Economic Statistics. 2019:1-10.

25. Apley DW, Zhu J. Visualizing the effects of predictor variables in black box supervised learning models. arXiv preprint arXiv:161208468. 2016.

26. Goldstein A, Kapelner A, Bleich J, Pitkin E. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. Journal of Computational and Graphical Statistics. 2015;24:44-65.

27. Cortez P, Embrechts MJ. Opening black box data mining models using sensitivity analysis. 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM): IEEE; 2011. 341-8.

28. Cortez P, Embrechts MJ. Using sensitivity analysis and visualization techniques to open black box data mining models. Information Sciences. 2013;225:1-17.

29. Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. European conference on computer vision: Springer; 2014. 818-33.

30. Samek W, Montavon G, Binder A, Lapuschkin S, Müller K-R. Interpreting the predictions of complex ml models by layer-wise relevance propagation. arXiv preprint arXiv:161108191. 2016.

31. Binder A, Bach S, Montavon G, Müller K-R, Samek W. Layer-wise relevance propagation for deep neural network architectures. Information Science and Applications (ICISA) 2016: Springer; 2016. 913-22.

32. Montavon G, Binder A, Lapuschkin S, Samek W, Müller K-R. Layer-wise relevance propagation: an overview. Explainable AI: interpreting, explaining and visualizing deep learning: Springer; 2019. 193-209.

33. Bach S, Binder A, Montavon G, Klauschen F, Müller K-R, Samek W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one. 2015;10:e0130140.

34. Montavon G, Lapuschkin S, Binder A, Samek W, Müller K-R. Explaining nonlinear classification decisions with deep taylor decomposition. Pattern Recognition. 2017;65:211-22.

35. Montavon G, Samek W, Müller K-R. Methods for interpreting and understanding deep neural networks. Digital Signal Processing. 2018;73:1-15.

36. Simonyan K, Vedaldi A, Zisserman A. Deep inside convolutional networks: Visualising image classification models and saliency maps. 2014.

37. Shrikumar A, Greenside P, Shcherbina A, Kundaje A. Not just a black box: Learning important features through propagating activation differences. arXiv preprint arXiv:160501713. 2016.

38. Springenberg JT, Dosovitskiy A, Brox T, Riedmiller M. Striving for simplicity: The all convolutional net. arXiv preprint arXiv:14126806. 2014.

39. Shrikumar A, Greenside P, Kundaje A. Learning important features through propagating activation differences. arXiv preprint arXiv:170402685. 2017.

40. Sundararajan M, Taly A, Yan Q. Axiomatic attribution for deep networks. arXiv preprint arXiv:170301365. 2017.

41. Zhou B, Khosla A, Lapedriza A, Oliva A, Torralba A. Learning deep features for discriminative localization. Proceedings of the IEEE conference on computer vision and pattern recognition 2016. 2921-9.

42. Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D. Grad-cam: Visual explanations from deep networks via gradient-based localization. Proceedings of the IEEE international conference on computer vision 2017. 618-26.

43. Selvaraju RR, Das A, Vedantam R, Cogswell M, Parikh D, Batra D. Grad-CAM: Why did you say that? arXiv preprint arXiv:161107450. 2016.

44. Hailesilassie T. Rule extraction algorithm for deep neural networks: A review. arXiv preprint arXiv:161005267. 2016.

45. Fu L. Rule generation from neural networks. IEEE Transactions on Systems, Man, and Cybernetics. 1994;24:1114-24.

46. Sato M, Tsukimoto H. Rule extraction from neural networks via decision tree induction. IJCNN'01 International Joint Conference on Neural Networks Proceedings (Cat No 01CH37222): IEEE; 2001. 1870-5.

47. Özbakır L, Baykasoğlu A, Kulluk S. A soft computing-based approach for integrated training and rule extraction from artificial neural networks: DIFACONN-miner. Applied Soft Computing. 2010;10:304-17.

48. Tsukimoto H. Extracting rules from trained neural networks. IEEE Transactions on Neural networks. 2000;11:377-89.

49. Setiono R, Leow WK. FERNN: An algorithm for fast extraction of rules from neural networks. Applied Intelligence. 2000;12:15-25.

50. Zilke JR, Mencía EL, Janssen F. Deepred–rule extraction from deep neural networks. International Conference on Discovery Science: Springer; 2016. 457-73.

51. Schmitz GP, Aldrich C, Gouws FS. ANN-DT: an algorithm for extraction of decision trees from artificial neural networks. IEEE Transactions on Neural Networks. 1999;10:1392-401.

52. Saad EW, Wunsch II DC. Neural network explanation using inversion. Neural networks. 2007;20:78-93.

53. Craven MW, Shavlik JW. Using sampling and queries to extract rules from trained neural networks. Machine learning proceedings 1994: Elsevier; 1994. 37-45.

54. Taha I, Ghosh J. Three techniques for extracting rules from feedforward networks. Intelligent Engineering Systems Through Artificial Neural Networks. 1996;6:23-8.

55. Augasta MG, Kathirvalavakumar T. Reverse engineering the neural networks for rule extraction in classification problems. Neural processing letters. 2012;35:131-50.

56. Lu H, Setiono R, Liu H. Effective data mining using neural networks. IEEE transactions on knowledge and data engineering. 1996;8:957-61.

57. Kahramanli H, Allahverdi N. Rule extraction from trained adaptive neural networks using artificial immune systems. Expert Systems with Applications. 2009;36:1513-22.

58. Wang S, Zhou T, Bilmes J. Bias also matters: Bias attribution for deep neural network explanation. International Conference on Machine Learning2019. 6659-67.

7. NOVEL SENSITIVITY METHOD FOR EVALUATING THE FIRST DERIVATIVE OF THE FEED-FORWARD NEURAL NETWORK OUTPUTS AND PERFORMING FEATURE SELECTION[5]

7.1. Introduction

Multilayer feedforward neural networks (FFDNN) are parameterized nonlinear models that approximate a mathematical mapping between the input features and the output target variables [1]. Although FFDNNs are known to possess the potential for approximating various functions [2,3], they are often treated as black-box models because of the complexity involved in generating the closed-form expression of the learned function. Sensitivity analysis can be performed to understand the relationship and influence of each input on the output of a problem [4–7]. Sensitivity analysis is performed by examining the change in the target output when one of the input features is perturbed. In other words, performing sensitivity analysis involves the computation of partial derivatives of the outputs with respect to the inputs. While a larger magnitude of partial derivative suggests a drastic change in output with a small variation in the input, a smaller magnitude of partial derivative suggests smaller sensitivity of the output to the input [4].

In FFDNNs the first derivative (i.e., $\frac{\partial y}{\partial x_k}$) of the output y with respect to the k^{th} input x_k is evaluated employing the backpropagation algorithm, which involves the application of derivative chain rule [8–11]. Application of chain rule in this context is similar to the one employed during

[5] This chapter is based on the paper "Novel Sensitivity Method for Evaluating the First Derivative of the Feed-Forward Neural Network Outputs". *J Big Data 8, 88 (2021).* https://doi.org/10.1186/s40537-021-00480-4. The material in this chapter was co-authored by Dayakar Naik Lavadiya (DNL), and Ravi Kiran Yellavajjala (RK). Contributions of authors are as follows: RK: Conception, design of work, interpretation of results, revising the manuscript, and acquiring funding. DLN: execution, data generation, coding, first draft preparation, interpretation of results, and revision of manuscript.

the training of an FFDNN where $\frac{\partial E}{\partial w_{ij}}$ is evaluated for backpropagating the error with respect to the parametric weights w_{ij} of the network [12–15]. The goal of this chapter is to evaluate the derivatives of the FFDNN outputs with respect to the inputs without the need for backpropagation employing numerical differentiation techniques.

Finite difference schemes are employed for evaluating numerical derivatives [16–18]. In finite difference schemes, the input features are perturbed one at a time (e.g. x_k) with a finite step size (h) and the change in the output of a trained FFDNN is obtained. Popularly employed finite difference schemes include finite difference approximation (FDA) (see Eq. (7.1)) and central finite difference approximation (CFDA) (see Eq. (7.2)) methods, which are given as follows

Finite difference approximation (FDA)

$$f'(x_1, x_2, \dots x_k, \dots x_q) \approx \frac{\left(f(x_1, x_2, \dots x_k + h, \dots x_q) - f(x_1, x_2, \dots x_k, \dots x_q)\right)}{h} \tag{7.1}$$

Central finite difference (CFDA)

$$f'(x_1, x_2, \dots x_k, \dots x_q) \approx \frac{\left(f(x_1, x_2, \dots x_k + h, \dots x_q) - f(x_1, x_2, \dots x_k - h, \dots x_q)\right)}{2h} \tag{7.2}$$

where $x = (x_1, x_2, \dots x_k, \dots x_q)' \in R^{q \times 1}$ are the inputs, q is the number of inputs, $f(.)$ is the function mapping the inputs to the output variable and, $f'(.)$ is the first partial derivative approximation of $f(.)$ with respect to the input x_k. However, finite difference schemes are prone to subtractive cancellation errors [19,20]. Subtractive cancellation errors are caused by subtracting two close numbers whose difference could be in the order of the precision of the calculations. This scenario is inevitable in the case of finite difference schemes due to the subtractive operation as seen in the numerators of Eq. (7.1) and Eq. (7.2) and the use of very low h values to lower the truncation errors [19]. With this, an additional computational step to

evaluate the ideal h value to minimize the truncation error without increasing the subtractive cancellation error is necessary when finite difference schemes are evaluated. A novel differentiation scheme is necessary to avoid this additional step and to achieve analytical quality derivatives by minimizing both truncation and subtractive cancellation errors.

In this chapter, firstly, a novel method for determining the analytical quality first derivative of feedforward deep neural network outputs is proposed and then its extension to generate explanations of the feed-forward neural networks predictions in terms of feature attribution is described. i.e., steps involved in performing sensitivity analysis of the FFDNN are provided. To this end, a brief overview on the concept of complex-step derivative approximation (CSDA) is provided, and its ability to circumventing the subtractive cancellation errors associated with other numerical differentiation techniques is illustrated in Section 7.2. Implementing CSDA in the framework of FFDNN for regression and classification tasks is demonstrated in Section 7.3, and steps involved in performing complex-step sensitivity analysis to generate explanations are mentioned in Section 7.4 and 7.5.

7.2. Complex-Step Derivative Approximation (CSDA)

CSDA is a numerical differentiation technique proposed by Lyness and Moler [21]. CSDA was successfully implemented in various fields of engineering, including aerospace [22–25], computational mechanics [26–28], estimation theory (e.g., second-order Kalman filter) [29], etc., for performing sensitivity analysis and evaluating the first-order derivatives. In this section, the mathematical description of CSDA to estimate analytical quality first-order derivative of a single scalar variable scalar function is provided [30].

Let f be an analytic function of a complex variable z. Also, assume that f is real on the real axis. Then f has a complex Taylor series expansion which is expressed as

$$f(x + ih) = f(x) + ihf'(x) - \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f'''(x) + \cdots \tag{7.3}$$

where, h is the step size and $i^2 = -1$. By taking the imaginary component of $f(x + ih)$, dividing it by the step size and truncating the higher-order terms in the Taylor series, the CSDA for the first derivative can be expressed as

$$f'(x) = \frac{\text{Imag}\big(f(x + ih)\big)}{h} + O(h^2) \tag{7.4}$$

where Imag (*) denotes the imaginary component and $O(h^2)$ is the second-order truncation error. It is interesting to note that there are no subtractive operations in Eq. (7.4), which are inevitable in the finite difference approximations (see Eq. (7.1) and Eq. (7.2)). The absence of subtractive operations in the numerator ensures that the CSDA is not prone to subtractive cancellation errors. Hence, a very small value of h can be chosen in order to eliminate the truncation errors without the fear of subtractive cancellation errors. A simple example is provided next, which illustrates the accuracy of CSDA over finite difference schemes.

7.2.1. Illustrative Example

Consider a smooth function f(x) provided in Eq. (7.5). The exact first-order derivative of the function computed at $x = \frac{\pi}{4}$ is given as 2.65580797029498.

$$f(x) = \frac{e^x + x^3}{\pi + \cos(\pi x)} \tag{7.5}$$

The numerical first-order derivative of the above function is evaluated using all three approximation methods, namely, finite difference approximation (Eq. (7.1)), central finite difference approximation (Eq. (7.2)), and CSDA (Eq. (7.4)). The step size h employed for the purpose of computation ranged from 10^{-1} to 10^{-16}. The absolute error (ε) for each step size is then evaluated using Eq. (7.6), and the results are shown in Figure 7.1.

$$\varepsilon = \left|\widehat{f'(x)} - f'(x)\right| \tag{7.6}$$

where $\widehat{f'(x)}$ is the approximate first derivative at $x = \frac{\pi}{4}$ for a chosen step size h and, $f'(x)$ is the exact first derivative of function f(x) at $x = \frac{\pi}{4}$.

From Figure 7.1, it can be noticed that the absolute error decreased initially for both FDA and CFDA with the reduction in the step size. However, for step sizes less than $h = 10^{-8}$ for FDA and $h = 10^{-5}$ for CFDA, the absolute error was found to increase. The increase in the absolute error after a certain step size can be attributed to the subtractive operation in the numerator of finite difference schemes. On the contrary, in the case of CSDA, the absolute error was not only found to decline with a reduction in step size but approached a double float precision (~10^{-16}) with a further decrease in the step size beyond $h = 10^{-7}$. In other words, no subtractive cancellation errors were observed, and hence analytical quality derivatives with errors reaching the precision employed were obtained.

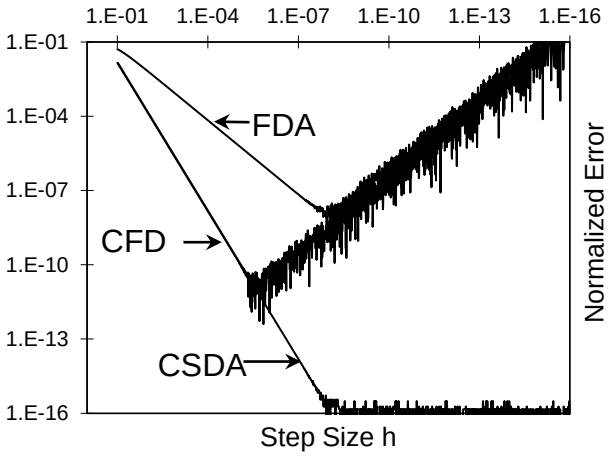


Figure 7.1. Illustration of the subtractive cancellation errors in finite difference methods and the CSDA. Both FDA and CFDA suffer from subtractive cancellation errors unlike CSDA. The truncation errors in CSDA can be minimized by choosing a very low h value. (CSDA – Complex-Step Derivative Approximation; FDA – Finite Difference, and CFDA – Central Finite Difference Approximation).

7.3. Implementation of CSDA in Feed-Forward Deep Neural Networks

Obtaining a closed-form expression in feedforward deep neural networks (FFDNN) is not only challenging but also a tedious task. Nevertheless, CSDA can be implemented in the framework of the feedforward deep neural network (FFDNN) for evaluating the variation of the output variable $y \in R$ with respect to the change in an input $x_k \in R$, where the subscript k represents the k^{th} input. The extended form of CSDA (see Eq. (7.4)) applied to a multivariate function can be expressed as

$$f'(x_1, x_2, \ldots x_k, \ldots x_q) = \frac{\text{Imag}\left(f(x_1, x_2, \ldots x_k + ih, \ldots x_q)\right)}{h} + O(h^2) \tag{7.7}$$

where $x = (x_1, x_2, \ldots x_k, \ldots x_q)' \in R^{q\times 1}$ are the input features, q is the number of input features, $f(.)$ is the function mapping the input features to the output target variable and, $f'(.)$ is the first-order derivative approximation of $f(.)$ with respect to the input feature x_k.

Implementation of CSDA in FFDNN involves three steps (see Figure 7.2): (1) configure and train the FFDNN for a given dataset, (2) perturb the input feature x_k one at a time (see Eq. (7.7)) with an imaginary step size of ih (where $h \ll 10^{-8}$) and perform the feedforward operation on the trained FFDNN and (3) obtain the output neuron's imaginary component with respect to the perturbed input and divide this component with the step size (h). Configuring the FFDNN is a trial-and-error process that involves finding the appropriate number of neurons and hidden layers in a network. A network is said to be configured when it is capable of learning an approximate mathematical mapping between the input features and the associated target variable such that it could be generalized to the unseen data instances. Guidelines for choosing trial configurations of FFDNN can be found elsewhere [31]. For training the feedforward neural network, the backpropagation algorithm, in conjunction with the Levenberg-Marquardt

optimization technique, is employed in this study [32]. Note that the code for implementing the CSDA in FFDNN was written and executed in the MATLAB® environment.

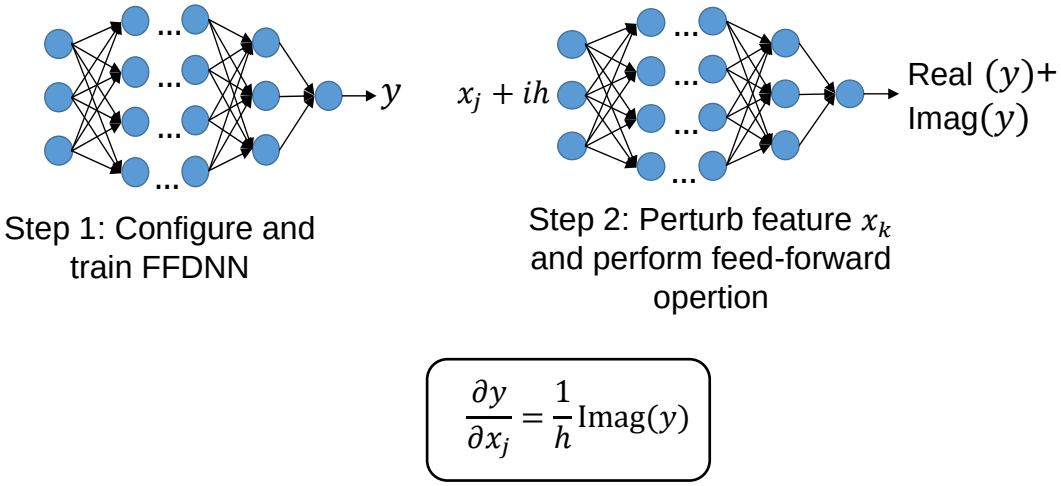


Figure 7.2. Schematic of steps involved for implementing CSDA in FFDNN framework.

7.3.1. Illustrative Example

For illustrating the effectiveness of the CSDA in computing the first order order derivative of FFDNN, a single variable function (see Eq. (7.8)) commonly employed in CSDA literature is chosen. A single hidden layer with 100 neurons is configured to train the FFDNN and the first order derivative is obtained at $x = \frac{\pi}{4}$ for step size of $h = 10^{-15}$. Both FDA and CFDA are also employed on the same trained FFDNN and first order derivative is obtained for same step size. The results along with the exact solution is provided in Table 7.1. From the Table 7.1 it is evident that the proposed methods result in least error (i.e., 2.9e-5) when compared to existing methods FDA (i.e., 0.145) and CFDA (i.e., 2.2e-3).

$$f(x) = \frac{e^x}{(\cos x)^3 + (\sin x)^3} \tag{7.8}$$

Furthermore the derivatives are evalauted for all the x values using CSDA, FDA and CFDA and is provided in Figure 7.3. Comparison of exact solution and the first order derivatives

evaluated using CSDA, FDA and CFDA.. From Figure 7.3 it can inferred that the proposed CSDA method predicts the analytical quality derivative that coincides with the exact solution. However in the case of FDA and CFDA the derivatives are found to be inaccurate due to subtractive cancellation errors.

Table 7.1. Comparison of error between CSDA and other existing methods.

@ x=pi/4	**Exact**	**CSDA**	**FDA**	**CFDA**
Output	3.10176	3.10167	3.55271	3.10862
Error	-	2.9e-5	0.1454	2.2e-3

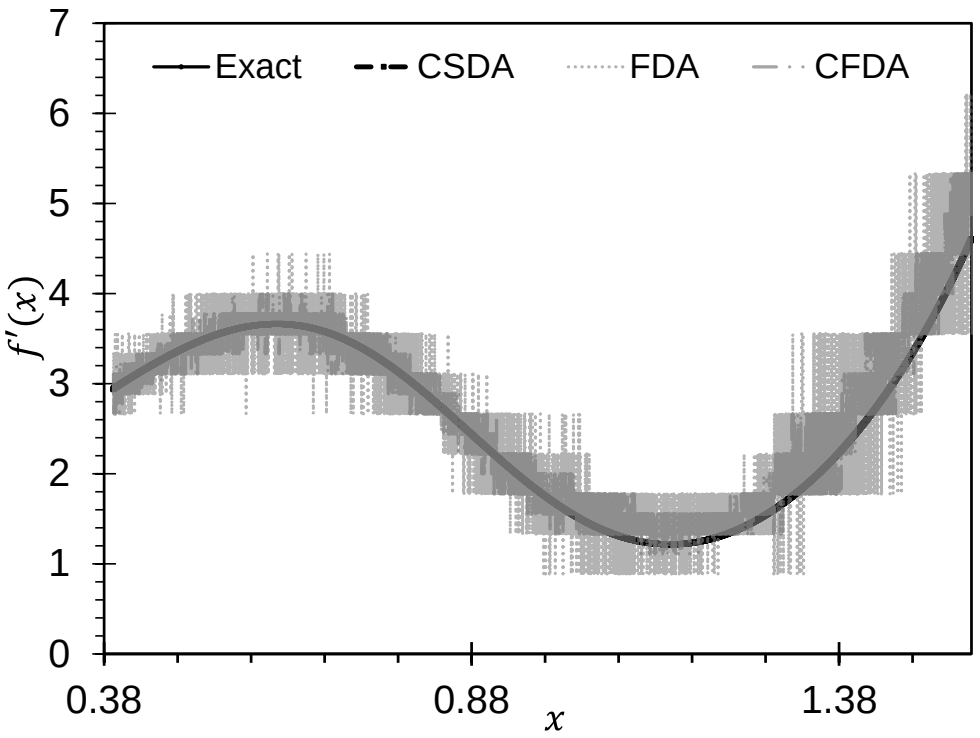


Figure 7.3. Comparison of exact solution and the first order derivatives evaluated using CSDA, FDA and CFDA.

In what follows, the implementation of CSDA is demonstrated for regression and classification tasks using artificial datasets consisting of more than one variable.

7.3.2. Regression

The process of generating artificial datasets (from a known analytic function) for performing regression is described in this subsection. The first-order derivative results are then obtained from the CSDA implemented FFDNN (see Eq. (7.7)) and are compared with the exact analytical derivatives of the known function.

7.3.2.1. Datasets and FFDNN Configurations

Three different single scalar-valued functions are employed in this study to generate artificial datasets for the regression task (see Table 7.2). While the first two functions R_1, R_2 have 3 input features x_1, x_2 and x_3, the third function R_3 is chosen to have 4 input features x_1, x_2, x_3 and x_4, wherein the feature x_4 represents the uniformly distributed random noise added to the function R_2. Since the added noise x_4 has no significant contribution in evaluating the output of the function R_3, the mean of the first-order derivative with respect to x_4 computed using CSDA would be expected to be zero. In other words, the purpose of adding noise is to verify the proposed method's ability to identify the least relevant feature. The input features employed in the dataset are real-valued and are independent of each other. In total, 2000 instances are randomly generated for each dataset from a uniform distribution of the feature values. The range of the values chosen for each input feature for all three datasets is summarized in Table 7.3. These randomly generated input features are then substituted in the respective functions R_1, R_2 and R_3 to obtain the associated target variables y for each dataset.

Table 7.2. Functions used to generate artificial datasets for regression.

Function	Exact Derivatives
R_1: $y = x_1^4 + 2x_2^3 + 3\sqrt{x_3}$	$\frac{\partial y}{\partial x_1} = 4x_1^3;\ \frac{\partial y}{\partial x_2} = 6x_2^2;\ \frac{\partial y}{\partial x_3} = \frac{3}{2\sqrt{x_3}}$
R_2: $y = \sin(\pi x_1) + e^{x_2} + x_3^2$	$\frac{\partial y}{\partial x_1} = \pi\cos(\pi x_1)\,;\ \frac{\partial y}{\partial x_2} = e^{x_2};\ \frac{\partial y}{\partial x_3} = 2x_3$
R_3: $y = \sin(\pi x_1) + e^{x_2} + x_3^2 + 0.00001x_4$	$\frac{\partial y}{\partial x_4} = 1e - 5$

For obtaining a suitable FFDNN configuration for each dataset, numerous trial configurations with varying numbers of neurons and hidden layers were examined beforehand. The trial configuration that resulted in a mean squared error (MSE) less than 1e-6 on the validation dataset is chosen as the suitable configuration for training the datasets. The final configuration of FFDNN that was adapted to train dataset 1 is 1st hidden layer (HL) (8 neurons) – 2nd HL (5 neurons); dataset 2 is 1st HL (10 neurons) – 2nd HL (5 neurons); and dataset 3 is 1st HL (10 neurons) – 2nd HL (5 neurons). Note that a soft plus function (see Figure 7.4(a)) ($\ln(1 + \exp(\Sigma))$, where, Σ is the net input function of a neuron) is used as an activation function for all the neurons in the hidden layers. The MSE of trained FFDNN associated with dataset 1, dataset 2, and dataset 3 are determined to be 8.2e-7, 5.6e-8, and 4.3e-7, respectively.

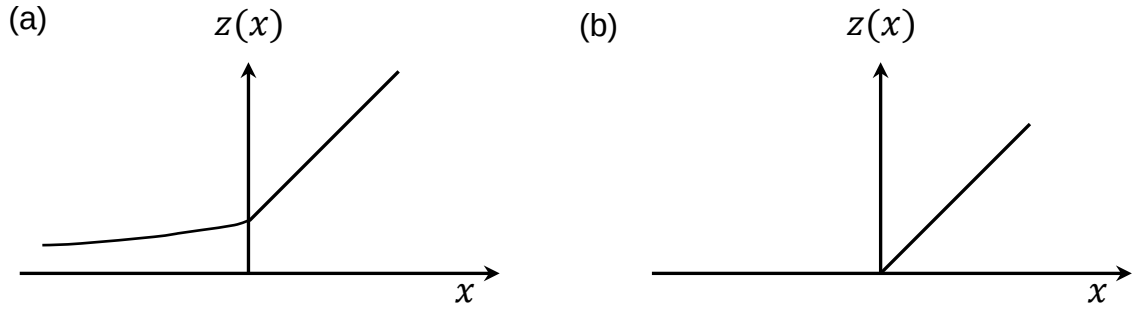


Figure 7.4. Activation function (z) employed for training FFDNNs (a) Softplus (for regression) and (b) ReLU (for classification).

7.3.2.2. Comparison of CSDA-FFDNN Output and the Exact Analytical Derivative

CSDA is implemented on the trained FFDNNs to evaluate the change in the predicted output variable $\hat{y}$ with respect to the input feature x_j where $j = 1, 2,$ and 3 for dataset 1 and dataset 2; and $j = 1, 2, 3,$ and 4 for dataset 3. Note that in CSDA implemented FFDNN, the predicted output ($\hat{y}$) is a complex variable. According to Eq. (7.7), only the imaginary component of $\hat{y}$ is required for obtaining the first-order derivative. More precisely, if g_1, g_2 and g_3 indicates the approximate function (mapping x to $\hat{y}$) learned by FFDNN for dataset 1,2 and 3, respectively, then the first-order derivative of g_1, g_2 and g_3 with respect to the feature x_j are computed as

$$g_1' = \frac{\text{Imag}\left(g_1(x_1, .., x_j + ih, .. x_q)\right)}{h};\ g_2' = \frac{\text{Imag}\left(g_2(x_1, .., x_j + ih, .. x_q)\right)}{h};\ g_3' = \frac{\text{Imag}\left(g_3(x_1, .., x_j + ih, .. x_q)\right)}{h}$$

where, $q = 3$ for dataset 1 and dataset 2, and $q = 4$ for dataset 3. Since there are 2000 instances in each dataset, the number of first-order derivatives evaluated with respect to each feature x_j is also 2000. The comparison between the first-order derivative evaluated (for all 2000 instances) using CSDA implemented FFDNN, and the exact analytical derivatives are provided in Figure 7.5 and Figure 7.6. From Figure 7.5, it is evident that the derivatives of the approximation function g_1 (for dataset 1) evaluated with respect to features x_1, x_2 and x_3 using CSDA are in good agreement with the exact analytical derivatives $\frac{\partial R_1}{\partial x_1}, \frac{\partial R_1}{\partial x_2}$ and $\frac{\partial R_1}{\partial x_3}$, respectively. Among all the data points for features x_1, x_2 and x_3, the maximum absolute error (ε) (see Eq. (7.6)) was found to occur at $x_1 = 1, x_2 = 0.006074$ and $x_3 = -1$. Similarly, from Figure 7.6(a)-(c), it is evident that the derivatives of the approximation function g_2 evaluated with respect to

x_1, x_2, and x_3 using CSDA are also in good agreement with the exact analytical derivatives $\frac{\partial R_2}{\partial x_1}, \frac{\partial R_2}{\partial x_2}$ and $\frac{\partial R_2}{\partial x_3}$. Among all the data points for features x_1, x_2 and x_3, the maximum absolute error (ε) was found to occur at $x_1 = 0.9855, x_2 = 1$ and $x_3 = 0.0005$ As mentioned earlier, in the case of function R_3 (see Figure 7.6 (d)) where the input feature x_4 is least relevant, the first derivative with respect to all values of x_4 are found to be scattered above and below the exact analytical derivative which is zero.

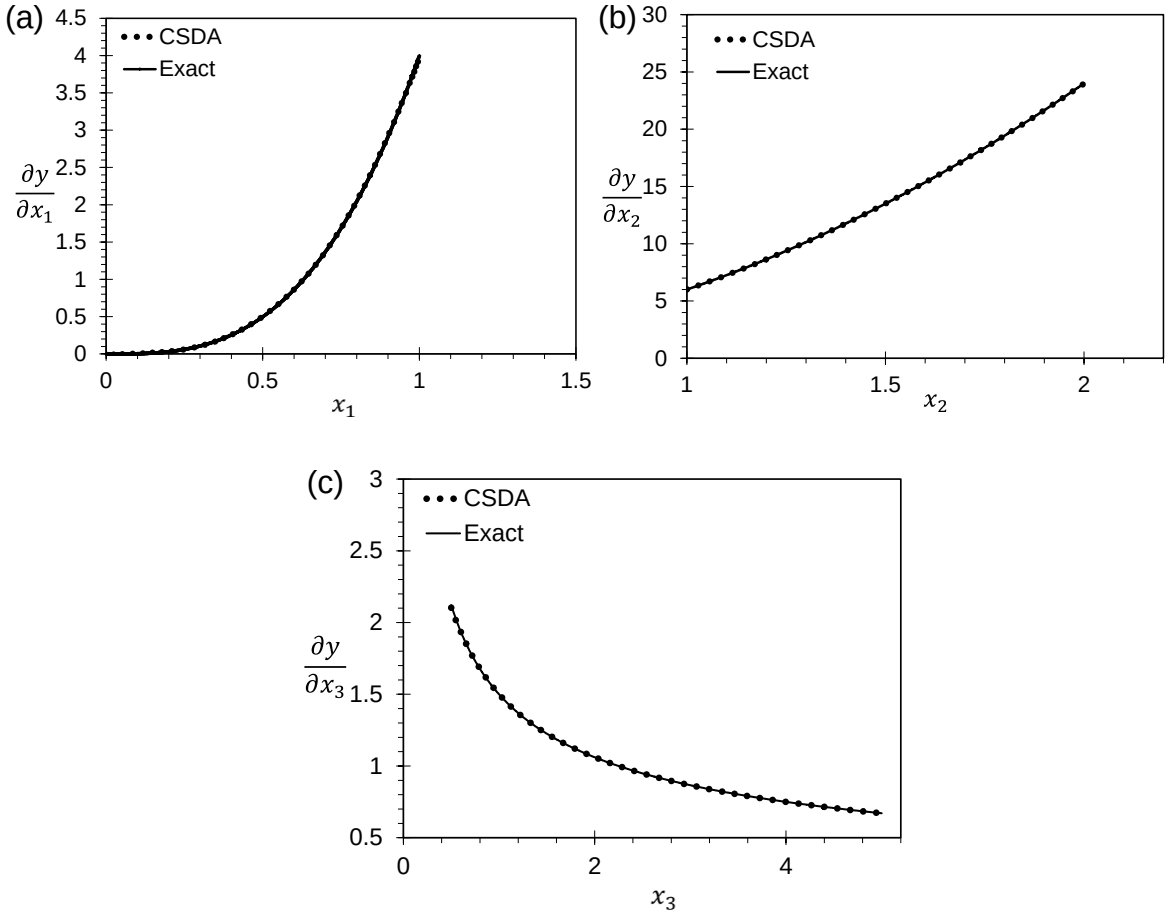


Figure 7.5. Comparison of the exact analytical solution and the first derivative evaluated using CSDA implemented FFDNN for Dataset 1.

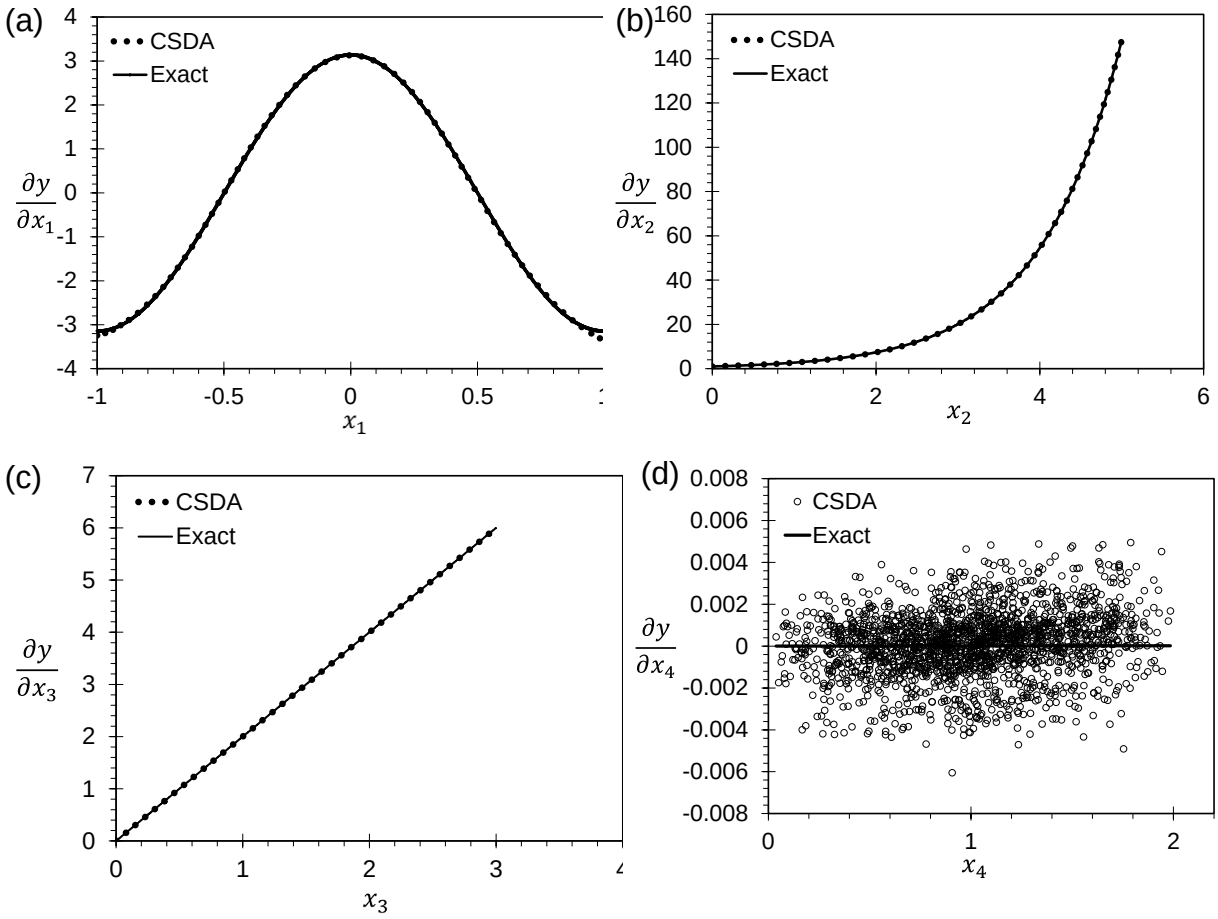


Figure 7.6. Comparison of the exact analytical solution and the first derivative evaluated using CSDA implemented FFDNN for Dataset 2 (a, b, and c) and Dataset 3 (d).

7.3.3. Classification

Unlike the regression task, evaluating the derivatives in the case of the classification task may not be feasible since the output of the FFDNN is discrete (e.g., SoftMax activation function outputs). However, considering the fact that the inputs fed to the SoftMax activation neurons in the output layer are not discrete, the first-order derivatives of such inputs could still be evaluated. These first-order derivatives will aid in providing information about the importance of the input features. In this subsection, the process of generating an artificial dataset for demonstrating the implementation of CSDA for classification tasks is described, and its significance in determining the top features is illustrated.

7.3.3.1. Dataset and CSDA Implementation

A binary class artificial dataset with input features x_1, x_2 and x_3 is generated such that all the instances belonging to class label 1 are enclosed within a cylinder of unit radius, and the rest of the instances belonging to class label 2 are outside the cylinder (see Figure 7.7). In total, 1000 instances are generated for each class label. Note that the feature x_3 is randomly chosen from a uniformly distributed noise with a zero mean, which has the least relevance in determining the class label. The purpose of including feature x_3 is to demonstrate that the proposed approach has the ability to identify the least significant features. The parametric equations used to generate the datasets are

$$\textit{Class Label 1}: x_1 = r_1 \cos(\theta_1)\,;\; x_2 = r_1 \sin(\theta_1)\,;\; x_3 \sim U(0, 0.0001)$$

$$\text{where } r_1 \sim U(0,1) \text{ and } \theta_1 \sim U(0, 2\pi)$$

$$\textit{Class Label 2}: x_1 = r_2 \cos(\theta_2)\,;\; x_2 = r_2 \sin(\theta_2)\,;\; x_3 \sim 0.0001 * U(0,1)$$

$$\text{where } r_2 \sim U(1,2) \text{ and } \theta_2 \sim U(0, 2\pi)$$

Table 7.3. Range of input features for generating regression dataset.

Function	**Range of input features**
R_1	$x_1 \sim U(0,1);\; x_2 \sim U(1,2);\; x_3 \sim U(0.5, 5)$
R_2	$x_1 \sim U(-1,1);\; x_2 \sim U(0,5);\; x_3 \sim U(0,3)$
R_3	$x_1 \sim U(-1,1);\; x_2 \sim U(0,5);\; x_3 \sim U(0,3);\; x_4 \sim U(0,2)$

Table 7.4. CSDA of net function in output neuron as a feature score.

Input feature, j =	**1**	**2**	**3**
$\frac{\partial \Sigma_{o_1}}{\partial x_j}$	0.5009	0.4935	0.0056
$\frac{\partial \Sigma_{o_2}}{\partial x_j}$	-0.5009	-0.4935	-0.0056

It is important to note that all three input features are independent of one another. Similar to the regression task, numerous trail configurations with a varying number of neurons and hidden layers were examined beforehand to obtain a suitable FFDNN configuration, i.e., a configuration that has prediction accuracy >98%. The configuration of FFDNN that was chosen to train the dataset is 1st HL (8 neurons) – 2nd HL (5 neurons). Note that Rectified Linear Unit (ReLU) (see Figure 7.4(b)) ($\max(0, \Sigma)$, where Σ is the net input function for a neuron) is used as an activation for all the neurons in the hidden layers and SoftMax function is used as an activation function for the neurons in the output layer.

The first derivative of the two net input functions in the output layer (i.e. Σ_{o_1} and Σ_{o_2}) in FFDNN with respect to input features x_1, x_2 and x_3 are obtained for all the data points using CSDA, and the sum of their absolute values (i.e., the sum of all 2000 data points) are provided in Table 7.4. Considering that the first derivative (i.e. $\frac{\partial \Sigma_{o_m}}{\partial x_j}$) with respect to each input feature x_j represents the proxy measure of its significance, the least relevant feature can be determined. In other words, the input feature that results in the lowest magnitude of the first derivative will be considered as the least relevant feature. From Table 7.4, it can be observed that the input feature x_3 has the lowest magnitude when compared to features x_1 and x_2. Therefore feature x_3 can be said to be the least relevant feature. In order to verify if the feature x_3 is irrelevant, the FFDNN is trained again with the exclusion of feature x_3 and the confusion matrix is shown in Table 7.5. From the confusion matrix, it is evident that the exclusion of feature x_3 does not influence the accuracy of classification. Furthermore, the precision and recall were also determined i.e., 0.99 and 0.98 respectively, and were noticed to be uninfluenced by the exclusion of feature x_3.

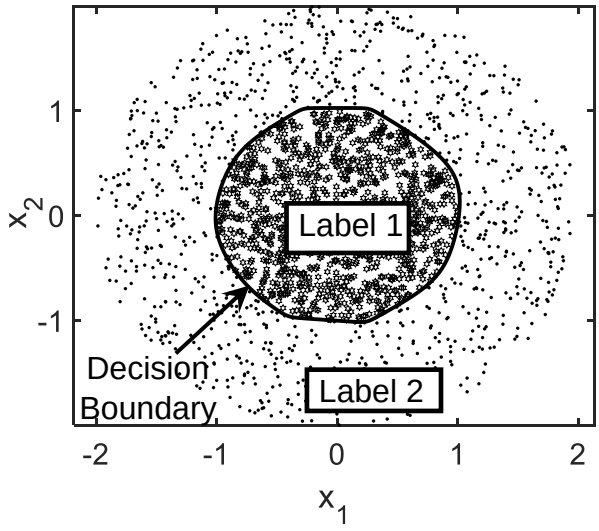


Figure 7.7. Decision boundary learned by FFDNN to classify the binary class artificial dataset.

Table 7.5. Confusion matrix excluding feature x_3.

		Predicted	
		Class Label 1	Class Label 2
Actual	Class Label 1	0.99	0.01
	Class Label 2	0.02	0.98

7.4. Feature Attribution Based XAI for Regression Using Complex-Step Sensitivity

The XAI feature attribution method (CS-FA-R) involving complex-step sensitivity is described in this section for the regression task. It consists of four steps (see Figure 7.8). While the first three steps lead to evaluation of first-order derivative as described in Section 7.3, the fourth step involves determination of features responsible for prediction. Note that step 2 and step 3 (see Figure 7.8) are repeated for all instances in the training dataset, and the average absolute magnitude of the first-order derivative of the target output with respect to the input feature is evaluated. For example, if y is the target output variable and x_{jk} is the k^{th} feature in the j^{th} observation that is complex-step perturbed (ih), then the first order derivative of the target

output with respect to the input feature averaged over all instances of the training dataset is expressed as (see Eq. (7.9))

$$\frac{\partial \mathrm{y}}{\partial \mathrm{x_k}} = \frac{1}{\mathrm{N}} \sum_{\mathrm{j=1}}^{\mathrm{N}} \left| \frac{\partial \mathrm{y}}{\partial \mathrm{x_{jk}}} \right| \tag{7.9}$$

where, N denotes the number of instances in the dataset, $\mathrm{k} = 1 \ldots \mathrm{q}$ indicates the input feature, and j represents the observation number in the dataset. In the fourth and final step, the rank of each input feature is determined based on the magnitude of the first-order derivatives evaluated, as shown in Eq. (7.9). The feature with a higher magnitude of the first-order derivative is assigned a higher rank and vice versa. Note that for training the feedforward neural network, a backpropagation algorithm, in conjunction with the Levenberg-Marquardt optimization technique, is employed in this study [45].

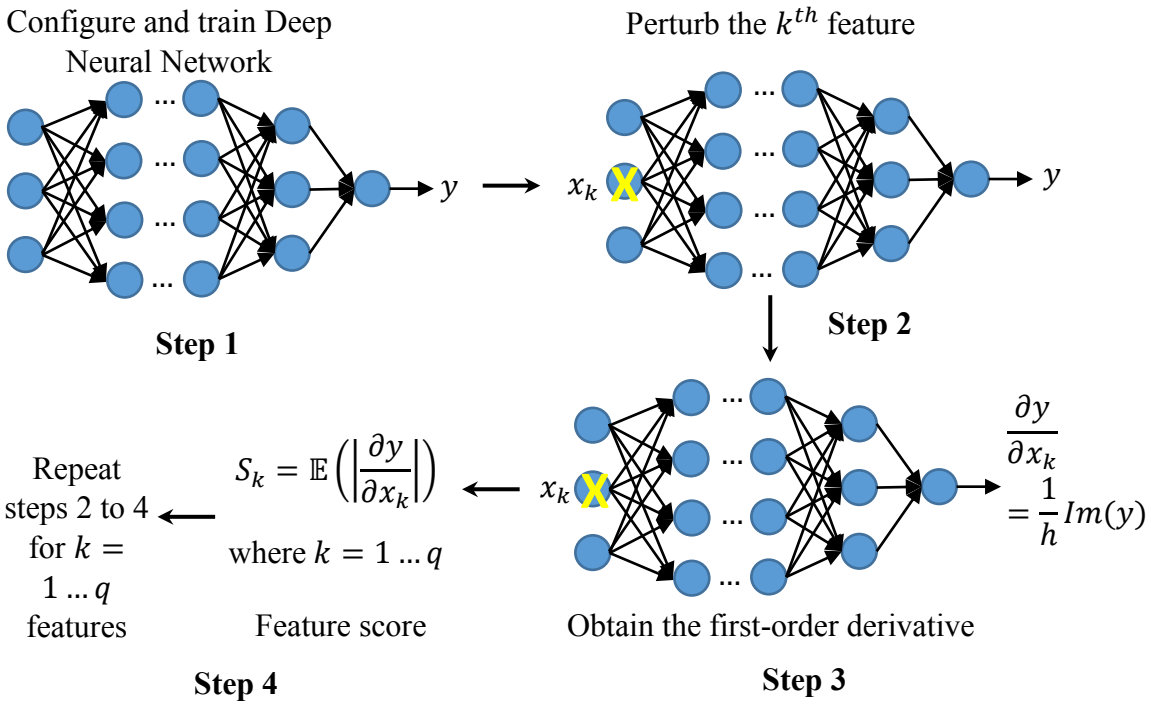


Figure 7.8. Steps involved in the complex-step sensitivity for regression task.

7.5. Feature Attribution Based XAI for Classification Using Complex-Step Sensitivity

The XAI feature attribution method (CS-FA-C) involving complex-step sensitivity is described in this section for the classification task. Unlike regression, a modification to step 3 (see Figure 7.8) is needed in the proposed method when feature attribution is determined for the classification task, i.e., evaluating the first-order derivative of target output with respect to perturbed input feature. The need for modification could be attributed to two reasons: (1) discrete output in the output layer and (2) multiple first-order derivatives yielded by the feed-forward neural network output layer (SoftMax layer) (see Figure 7.9). Considering the fact that the inputs fed to the SoftMax activation neurons in the output layer are not discrete, the first-order derivatives of such inputs could still be evaluated. These first-order derivatives will aid in providing information about the importance of the input features. If Σ_r represents the net function of r^{th} neuron in the SoftMax layer, then the first-order derivative of the net function Σ_r with respect to the k^{th} feature x_k is expressed as (see Eq. (7.10))

$$\left(\frac{\partial \Sigma_r}{\partial x_k}\right) = \frac{1}{h}\mathrm{Imag}\left(\Sigma_r(x_k + ih)\right) \tag{7.10}$$

where, $r = 1 \ldots\ldots m$ and m indicates the number of class labels. To quantify the change in the target output with respect to the k^{th} input feature x_k, the average of the first-order derivatives obtained for all neurons in the output layer is determined. This average magnitude is referred to as saliency (S_k) of k^{th} input feature [25] and is expressed as (see Eq. (7.11))

$$S_k = \frac{1}{N}\sum_{j=1}^{N}\left(\sum_{r=1}^{m}\left|\left(\frac{\partial \Sigma_r}{\partial x_{jk}}\right)\right|\right) \tag{7.11}$$

where r denotes the neuron in the SoftMax output layer, m represents the number of class labels, Σ_r represents the net function of r^{th} neuron in the SoftMax layer. The rank of each input feature

is then determined based on the magnitude of the first-order derivatives for each perturbed feature x_k determined as shown in Eq. (7.11).

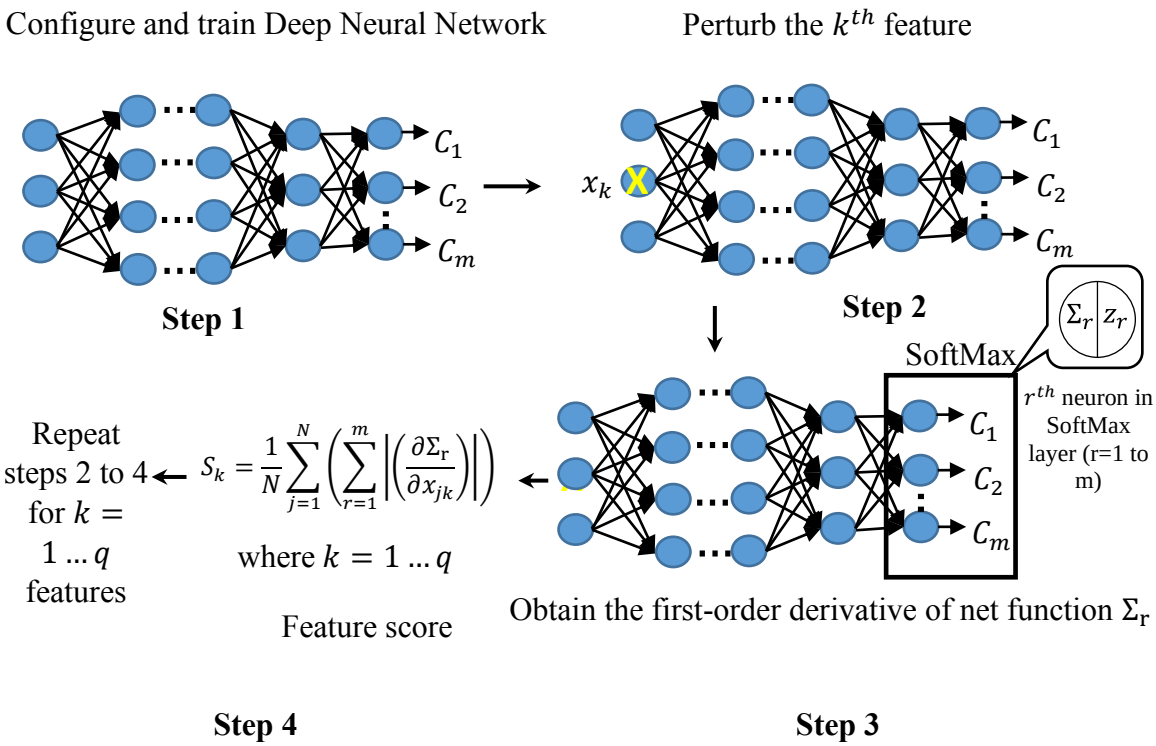


Figure 7.9. Steps involved in the complex-step sensitivity for the classification task.

The implementation of CS-FA for generation explanations of FFDNN predictions is illustrated in Chapter 8. For this purpose, both regression and classification datasets are considered.

7.6. References

1. Zhang Z, Beck MW, Winkler DA, Huang B, Sibanda W, Goyal H. Opening the black box of neural networks: methods for interpreting neural network models in clinical applications. Ann Transl Med. 2018;6:216–216. https://doi.org/10.21037/atm.2018.05.32.
2. Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. Neural Netw. 1989;2:359–66. https://doi.org/10.1016/0893-6080(89)90020-8.

3. Takahashi Y. Generalization and approximation capabilities of multilayer networks. Neural Comput. 1993;5:132–9. https://doi.org/10.1162/neco.1993.5.1.132.
4. Nourani V, Sayyah FM. Sensitivity analysis of the artificial neural network outputs in simulation of the evaporation process at different climatologic regimes. Adv Eng Softw. 2012;47:127–46. https://doi.org/10.1016/j.advengsoft.2011.12.014.
5. Cao M, Alkayem NF, Pan L, Novák D. Advanced methods in neural networks-based sensitivity analysis with their applications in civil engineering. Artif Neural Netw Model Appl. 2016. https://doi.org/10.5772/64026.
6. Kowalski PA, Kusy M. Sensitivity analysis for probabilistic neural network structure reduction. IEEE Trans Neural Netw Learn Syst. 2018;29:1919–32. https://doi.org/10.1109/TNNLS.2017.2688482.
7. Cortez P, Embrechts MJ. Using sensitivity analysis and visualization techniques to open black box data mining models. Inf Sci (Ny). 2013;225:1–17. https://doi.org/10.1016/j.ins.2012.10.039.
8. Engelbrecht AP, Cloete I. Sensitivity analysis algorithm for pruning feedforward neural networks. IEEE Int. Conf. Neural Networks - Conf. Proc., vol. 2, IEEE; 1996, 1274–8. https://doi.org/10.1109/icnn.1996.549081.
9. Nguyen-Thien T, Tran-Cong T. Approximation of functions and their derivatives: a neural network implementation with applications. Appl Model. 1999;23:687–704. https://doi.org/10.1016/S0307-904X(99)00006-2.
10. Hornik K, Stinchcombe M, White H. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. Neural Netw. 1990;3:551–60. https://doi.org/10.1016/0893-6080(90)90005-6.

11. Hashem S. Sensitivity analysis for feedforward artificial neural networks with differentiable activation functions, Institute of Electrical and Electronics Engineers (IEEE); 2003, 419–24. https://doi.org/10.1109/ijcnn.1992.287175.

12. Christopher MB. Neural networks for pattern recognition. Oxford: Oxford University Press; 1995.

13. Ruck DW, Rogers SK, Kabrisky M. Feature selection using a multilayer perceptron. J Neural Netw Comput. 1990;2:48–8.

14. Bo L, Wang L, Jiao L. Multi-layer perceptrons with embedded feature selection with application in cancer classification. Chin J Electron. 2006;15:832–5.

15. Gasca E, Sánchez JS, Alonso R. Eliminating redundancy and irrelevance using a new MLP-based feature selection method. Pattern Recognit. 2006;39:313–5. https://doi.org/10.1016/j.patcog.2005.09.002.

16. Montaño JJ, Palmer A. Numeric sensitivity analysis applied to feedforward neural networks. Neural Comput Appl. 2003;12:119–25. https://doi.org/10.1007/s00521-003-0377-9.

17. Güne¸ A, Baydin G, Pearlmutter BA, Siskind JM. Automatic differentiation in machine learning: a survey. J Mach Learn Res. 2018;18.

18. Jerrell ME. Automatic differentiation and interval arithmetic for estimation of disequilibrium models. Comput Econ. 1997;10:295–316. https://doi.org/10.1023/A:1008633613243.

19. Driscoll TA, Braun RJ. Fundamentals of Numerical Computation. 2017.

20. Boudjemaa R, Cox MG, Forbes AB, Harris PM. Report to the National Measurement Directorate, Department of Trade and Industry From the Software Support for Metrology

Programme Automatic Differentiation Techniques and their Application in Metrology. 2003.

21. Lyness JN, Moler CB. Numerical Differentiation of Analytic Functions. SIAM J Numer Anal. 1967;4:202–10. https://doi.org/10.1137/0704019.

22. Martins J, Sturdza P, Alonso J, Martins JR, Alonso JJ. The complex-step derivative approximation. ACM Trans Softw Assoc Comput Mach. 2003;29:245–62. https://doi.org/10.1145/838250.838251.

23. Conolly J, Lake M. Geographical information systems in archaeology. Cambridge: Cambridge University Press; 2006. 338.

24. Campbell AR. Numerical Analysis of Complex-Step Differentiation in Spacecraft Trajectory Optimization Problems. 2011.

25. Lai KL, Crassidis JL. Extensions of the first and second complex-step derivative approximations. J Comput Appl . 2008;219:276–93. https://doi.org/10.1016/j.cam.2007.07.026.

26. Kiran R, Khandelwal K. Automatic implementation of finite strain anisotropic hyperelastic models using hyper-dual numbers. Comput Mech. 2015;55:229–48. https://doi.org/10.1007/s00466-014-1094-1.

27. Kiran R, Li L, Khandelwal K. Complex perturbation method for sensitivity analysis of nonlinear trusses. J Struct Eng. 2017;143:04016154. https://doi.org/10.1061/(asce)st.1943-541x.0001619.

28. Kiran R, Khandelwal K. Complex step derivative approximation for numerical evaluation of tangent moduli. Comput Struct. 2014;140:1–13. https://doi.org/10.1016/j.compstruc.2014.04.009.

29. Lai KL, Crassidis JL, Cheng Y, Kim J. New complex-step derivative approximations with application to second-order Kalman filtering. Collect. Tech. Pap. - AIAA Guid. Navig. Control Conf., vol. 2, 2005, 982–98. https://doi.org/10.2514/6.2005-5944.

30. Squire W, Trapp G. Using complex variables to estimate derivatives of real functions. SIAM Rev. 1998;40:110–2. https://doi.org/10.1137/S003614459631241X.

31. Hagan MT, Demuth HB, Beale MH, De Jesus O. Neural network design. 2nd ed. Oklahoma: Martin Hagan; 2014.

32. Hagan MT, Menhaj MB. Training feedforward networks with the Marquardt Algorithm. IEEE Trans Neural Netw. 1994;5:989–93. https://doi.org/10.1109/72.329697.

8. NUMERICAL EXPERIMENTS[6]

In this chapter, the numerical experiments are performed to demonstrate the effectiveness of the proposed CS-FA-R and CS-FA-C method for generating FFDNN explanations. For this purpose, both the real-world datasets and hyperspectral dataset (see Chapter 5) are considered. The real-world datasets are chosen from UCI open-source data repository [1]. Furthermore, KernelSHAP XAI technique (see Chapter 6) is implemented, and the results are compared.

8.1. Real-World Datasets

Three real-world datasets, each for regression and classification problems, are employed. For regression problems, the body fat percentage dataset, abalone dataset, and wine quality dataset are chosen, and, for the classification task, a vehicle dataset, segmentation dataset, and breast cancer dataset are chosen. The descriptive features and target variables for each dataset are mentioned as follows.

Regression

Body fat percentage dataset [2]: Features – (1) Age (years), (2) Weight (kg), (3) Height (cm), (4) Neck (cm), (5) Chest (cm), (6) Abdomen (cm), (7) Hip (cm), (8) Thigh (cm), (9) Knee (cm), (10) Ankle (cm), (11) Biceps (cm), (12) Forearm (cm), (13) Wrist (cm); Target variable – percentage of body fat.

Abalone dataset [3]: Features – (1) Female, (2) Infant, (3) Male, (4) Length (gms.), (5) Diameter (gms.), (6) Height (gms.), (7) Whole weight (gms.), (8) Shucked weight (gms.), (9) Viscera weight (gms.), (10) Shell weight (gms.); Target variable – Number of rings.

[6]This chapter is based on the paper "A Novel Sensitivity-based Method for Feature Selection", J Big Data 8, 128 (2021). https://doi.org/10.1186/s40537-021-00515-w. The material in this chapter was co-authored by Dayakar Naik Lavadiya (DNL), and Ravi Kiran Yellavajjala (RK). Contributions of authors are as follows: RK: Conception, design of work, interpretation of results, revising the manuscript, and acquiring funding. DLN: execution, data generation, coding, first draft preparation, interpretation of results, and revision of manuscript.

Wine quality dataset [4]: Features – (1) fixed acidity, (2) volatile acidity, (3) citric acid, (4) residual sugar, (5) chlorides, (6) free sulfur dioxide, (7) total sulfur dioxide, (8) density, (9) pH, (10) sulfates, (11) alcohol; Target variable – quality score (1 to 10).

Classification

Vehicle dataset [5]: Features – (1) Compactness, (2) circularity, (3) radius circularity, (4) radius ratio, (5) axis aspect ratio, (6) maximum length aspect ratio, (7) scatter ratio, (8) elongatedness, (9) axis rectangularity, (10) maximum length rectangularity, (11) scaled variance major, (12) scaled variance minor, (13) scaled radius of gyration, (14) skewness major, (15) skewness minor, (16) kurtosis major, (17) kurtosis minor, (18) hollow ratio; Target variable – Class label 1 (van), Class label 2 (Saab), Class label 3 (bus), Class label 4 (Opel).

Segmentation dataset [1]: Features – (1) region-centroid-col (2) region-centroid-row (3) short-line-density (4) the results of a line extraction algorithm that counts how many lines of length (5) vedge-mean (6) vedge-sd (7) hedge-mean (8) hedge-sd (9) intensity-mean (10) rawred-mean (11) rawblue-mean (12) rawgreen-mean (13) exred-mean (14) exblue-mean (15) exgreen-mean (16) value-mean (17) saturatoin-mean (18) hue-mean; Target variable – Class label 1 (Window), Class label 2 (foilage), Class label 3 (brickface), Class label 4 (path), Class label 5 (cement), Class label 6 (grass), Class label 7 (sky).

Breast cancer dataset [6]: Features – (1) radius1, (2) texture1, (3) perimeter1, (4) area1, (5) smoothness1, (6) compactness1, (7) concavity1, (8) concave points1, (9) symmetry1, (10) fractal dimension1, (11) radius2, (12) texture2, (13) perimeter2, (14) area2, (15) smoothness2, (16) compactness2, (17) concavity2, (18) concave points2, (19) symmetry2, (20) fractal dimension2, (21) radius3, (22) texture3, (23) perimeter3, (24) area3, (25) smoothness3, (26)

compactness3, (27) concavity3, (28) concave points3, (29) symmetry3, (30) fractal dimension3; Target variable – Class label 1 (Benign), Class label 2 (Malignant).

Other details about regression and classification datasets are provided in Table 8.1 and Table 8.2, respectively.

Table 8.1. Description of the datasets used for regression task.

Dataset name	Instances	No. of features	No. of target variables
Bodyfat	252	13	1
Abalone	4177	10	1
Wine quality	1599	11	1

Table 8.2. Description of the datasets used for the classification task.

Dataset name	Instances	No. of features	No. of class labels
Vehicle	846	30	4
Segmentation	210	18	7
Breast cancer	569	18	2

8.1.1. Configuring Feed-Forward Neural Networks

Feed-forward deep neural networks (FFDNN) with three hidden layers (HL) are configured to train on the regression and classification datasets. While a configuration of 1^{st} HL – 20 neurons, 2^{nd} HL – 10 neurons, and 3^{rd} HL – 5 neurons is employed to train on regression datasets, a configuration of 1^{st} HL – 60 neurons, 2^{nd} HL – 40 neurons, and 3^{rd} HL – 20 neurons is employed to train on classification datasets. A Rectified Linear Unit (ReLU) nonlinear function (see Figure 7.4(b)) is used as an activation function for all the configurations [7]. For the purpose of training, validating, and testing the chosen configurations, the datasets are randomly partitioned into 70:15:15 ratio, respectively. Note that in the case of the classification task, the partition ratio is maintained consistently for each class label, i.e., 70:15:15 of training, validation, and testing data from each class label is chosen. To ensure that the chosen configurations yield repeatable results, the training operation is performed 100 times with the same partition ratio but

with the replacement of instances randomly selected in every iteration. The performance metric, namely mean squared error (MSE) and accuracy, are evaluated for regression and classification datasets, respectively, for chosen configurations. The average MSE error for body fat percentage, abalone, and wine quality datasets is determined to be 20.41, 4.6, and 0.53, respectively. The average accuracy for the vehicle, segmentation, and breast cancer dataset is determined to be 75%, 80% and, 90%, respectively. The addition of more hidden layers or neurons in each hidden layer to the chosen configuration was found to yield similar MSE errors or accuracies and hence are not considered in this study.

8.1.2. Results

The proposed complex-step sensitivity analysis (i.e., CS-FA-R and CS-FA-C) is performed on the trained FFDNN and the important features governing the predictions of each dataset are identified. Other popularly employed XAI technique (for feature attribution), known as KernelSHAP, is also considered in this study and the results are compared. SHAP library available on Python software is used for this purpose.

From Table 8.3, it can be inferred that both KernelSHAP and complex-step sensitivity analysis yielded Feature 6 (Abdomen) as the most important feature and Feature 9 (Knee) as the least relevant feature for determining the percentage of body fat. The order of importance of other features are found to vary. Furthermore, the MSE for body fat dataset with each feature's inclusion is evaluated for both CS-FA-R and KernelSHAP methods and is shown in Figure 8.1(a). From Figure 8.1(a), it is evident that the overall trend of MSE for FFDNN decreases with the inclusion of each feature. The proposed complex-step sensitivity method and the KernelSHAP were found to yield lower MSE with seven top-most features.

In the case of the abalone dataset, the Feature 7 (Whole Weight), Feature 8 (Shucked Weight) and Feature 10 (Shell Weight) are found to be the top three features and Feature 1 (female), Feature 2 (infant), and Feature 3 (male) are found to be the least relevant features by both CS-FA-R and KernelSHAP (see Table 8.3). However, the order of importance of remaining four features' Feature 4 (Length), Feature 5 (Diameter), Feature 6 (Height) and Feature 9 (Viscera Weight) was found to vary. Similar to the body fat dataset, the MSE of FFDNN with the inclusion of each feature is determined for CS-FA-R and KernelSHAP and is shown in Figure 8.1(b). From Figure 8.1 (b), it can be inferred that the trend of KernelSHAP and the proposed method are similar. A lower MSE is achieved with inclusion of top 3 features.

In the case of the wine quality dataset, the Feature 10 (sulfates), Feature 11 (alcohol), Feature 2 (volatile acidity) and Feature 7 (total sulfurdioxide) are found to be the top four features and Feature 3 (citric acid) and Feature 8 (density) are found to be the least relevant features by both CS-FA-R and KernelSHAP (see Table 8.1). The order of importance of remaining five features' Feature 1 (fixed acidity), Feature 4 (residual sugar), Feature 5 (chlorides), Feature 6 (free sulfurdioxide) and Feature 9 (pH) was found to vary. The MSE of FFDNN with the inclusion of each feature for wine quality dataset is determined for CS-FA-R and KernelSHAP and is shown in Figure 8.1 (c). From Figure 8.1 (c), it can be inferred that the trend of KernelSHAP and the proposed method are similar. Both KernelSHAP and the proposed method identified the Feature 10 (sulfates), Feature 11 (alcohol), Feature 2 (volatile acidity) and Feature 7 (total sulfurdioxide) that yield the lowest MSE.

Table 8.3. Important features identified by KernelSHAP and CS-FA-R (ranked in the descending order of their importance).

Bodyfat dataset		**Abalone dataset**		**Wine quality dataset**	
SHAP	CS-FA-R	SHAP	CS-FA-R	SHAP	CS-FA-R
6	6	7	7	11	10
11	3	8	8	10	11
5	13	10	6	2	2
7	4	5	10	7	7
12	8	9	9	9	5
2	2	6	4	5	1
13	1	4	5	6	9
10	7	3	2	4	6
3	5	2	3	8	4
8	12	1	1	1	8
1	11			3	3
4	10				
9	9				

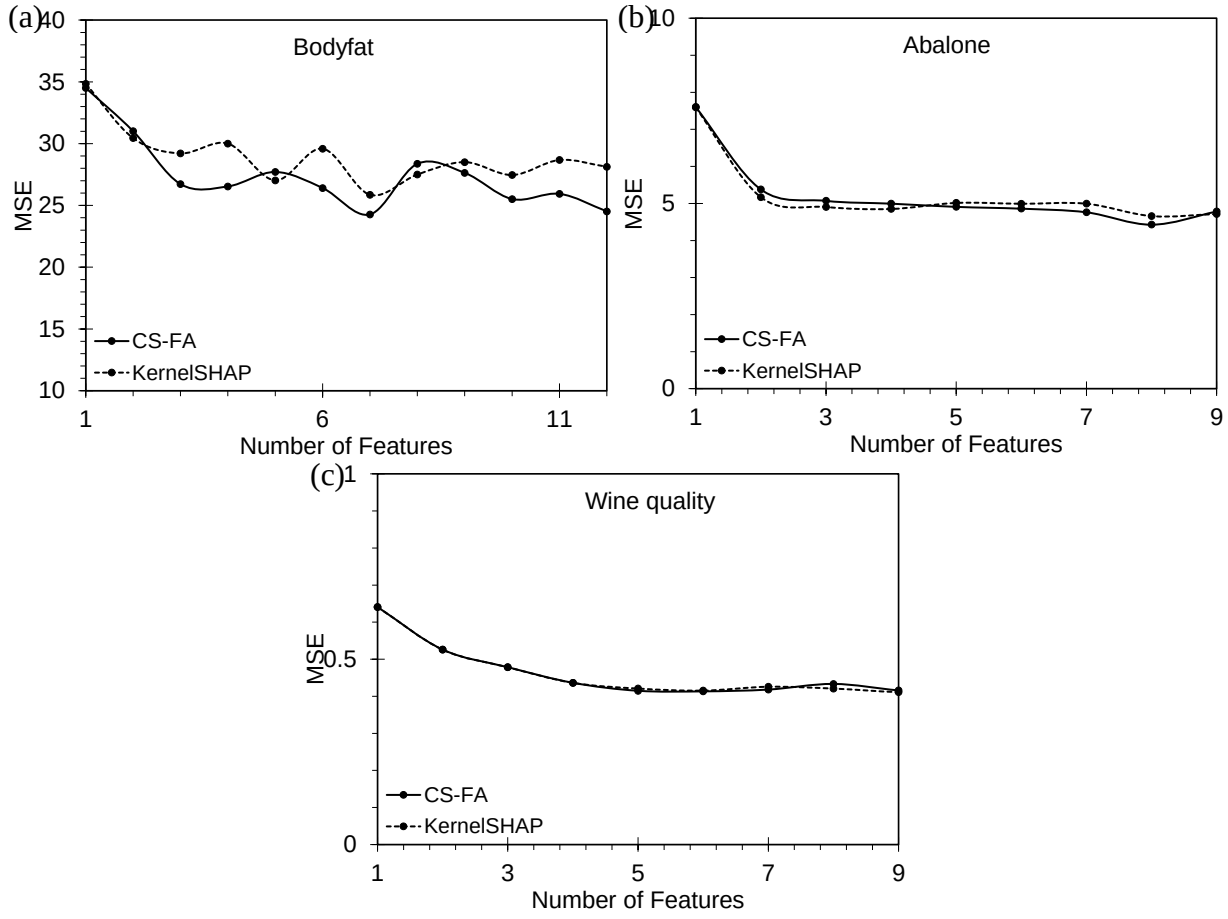


Figure 8.1. Comparison of the complex-step sensitivity method KernelSHAP for the classification task.

From Table 8.4, it can be inferred that both KernelSHAP and the proposed method dentified similar least relevant features for the vehicle dataset (Feature 2 (circularity), Feature 13 (scaled radius of gyration), Feature 15 (skewness minor) and Feature 16 (kurtosis major)). However, the rank of the remaining features was found to vary. Among the first top 6 features, four features were found to be common for Kernel SHAP and the proposed method namely Feature 5 (axis aspect ratio), Feature 10 (maximum length rectangularity), Feature 14 (skewness major) and Feature 18 (hollow ratio). Furthermore, the trend of the accuracy is determined for vehicle dataset for Kernel SHAP and the proposed method with the inclusion of each feature in succession and is shown in Figure 8.2(a). From Figure 8.2 (a), it is evident that the accuracy of the FFDNN increases with the addition of each feature for the vehicle dataset. The proposed method yielded an accuracy of 75% by selecting only the top 6 features and was found to slightly outperform KernelSHAP.

Similar to the vehicle dataset, KernelSHAP and the proposed method yielded similar least irrelevant features in the case of the segmentation dataset. The least relevant features are identified as (Feature 3 (short-line density), Feature 4 (lines of length), Feature 5 (vedge-mean), Feature 6 (vedge-sd), Feature 7 (hedge-mean), and Feature 8 (hedge-sd)). Interestingly the top 4 features are also found to be similar, Feature 2 (region-centroid-row), Feature 13 (exred-mean), Feature 15 exgreen-mean) and Feature 18 (hue-mean). The trend of the accuracy for the segmentation dataset is determined for KernelSHAP and the proposed method with the inclusion of each feature in succession and is shown in Figure 8.2(b). From Figure 8.2 (b), it is evident that the accuracy of the FFDNN increases with the addition of each feature for the segmentation dataset. Both methods were found to perform similar yielding the highest accuracy of 90% with only the top 6 features.

Interestingly, in the breast cancer dataset, KernelSHAP and the proposed method both resulted in similar top-most features, i.e., Feature 21 (radius3) and feature 23 (perimeter3). However, the order of importance of other features were found to vary. Unlike vehicle and segmentation dataset no common irrelevant features were identified. The trend of accuracy is obtained for the breast cancer dataset with the inclusion of each feature in each succession and is shown in Figure 8.2 (c). In the case of the breast cancer dataset, the trend of Kernel SHAP and the proposed method was found to be more or less similar. An accuracy of 93% is achieved by the inclusion of the top two features, i.e., Feature 21 (radius3) and Feature 23 (perimeter3).

Table 8.4. Important features identified by SHAP and CS-FA-C (ranked in the descending order of their importance).

	Method	**Feature Ranking**
Vehicle dataset	SHAP	3, 14, 18, 10, 5, 1, 12, 8, 17, 9, 7, 6, 4, 11, 2, 15, 13, 16
	CS-FA	10, 8, 5, 17, 14, 18, 11, 3, 6, 12, 7, 1, 9, 4, 13, 2, 15, 16
Segmentation dataset	SHAP	15, 2, 13, 18, 17, 12, 1, 16, 14, 10, 9, 7, 11, 5, 3, 6, 4, 8
	CS-FA	2, 18, 15, 13, 10, 16, 11, 12, 17, 9, 14, 6, 8, 7, 5, 4, 3, 1
Breast cancer dataset	SHAP	28, 23, 8, 22, 24, 3, 7, 25, 29, 1, 4, 14, 13, 2, 27, 26, 16, 18, 21, 20, 11, 10, 19, 17, 15, 5, 30, 6, 12, 9
	CS-FA	21, 23, 28, 20, 8, 4, 7, 11, 24, 17, 15, 2, 22, 30, 12, 26, 13, 16, 1, 14, 10, 9, 29, 25, 18, 19, 6, 3, 27, 5.

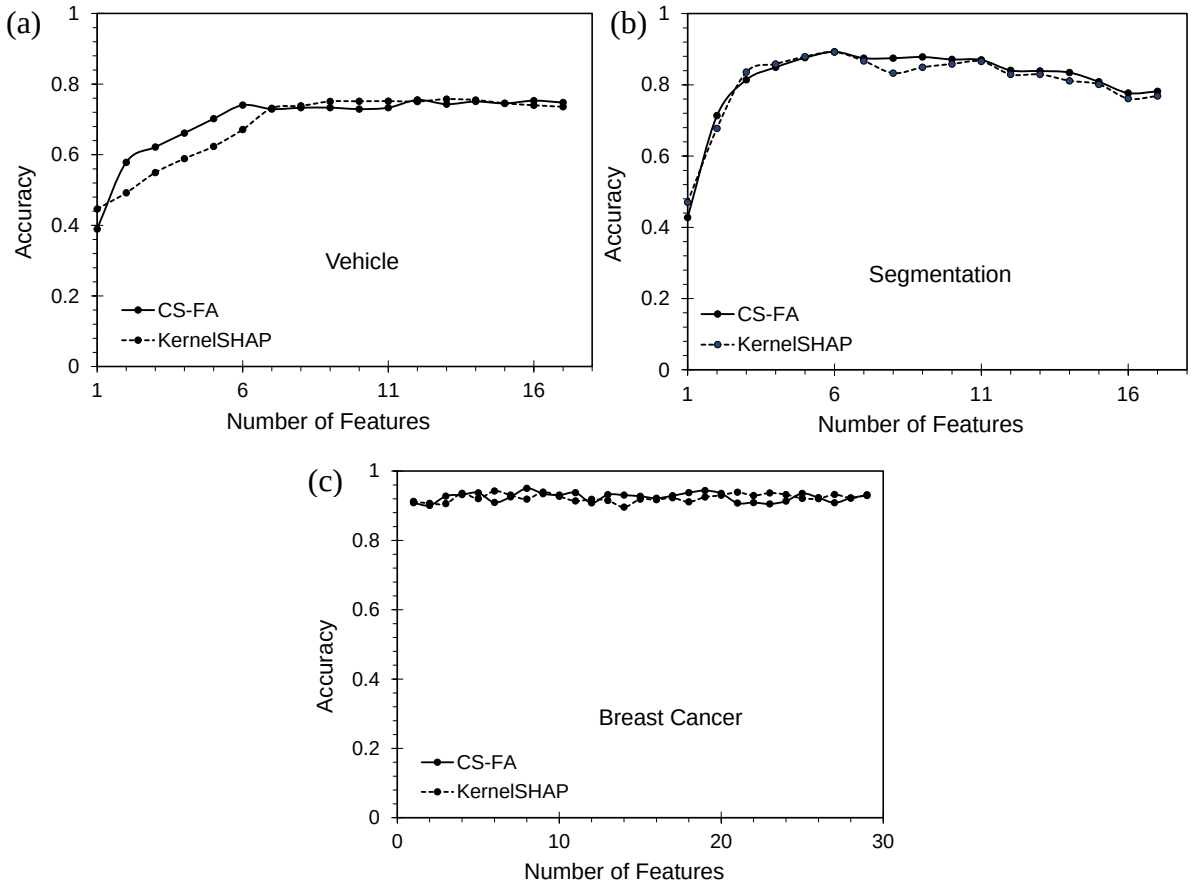


Figure 8.2. Comparison of the complex-step sensitivity method KernelSHAP for the classification task.

8.2. Hyperspectral Dataset of Corroded ASTM A572 Plates

The calibrated spectral reflectance data of the corroded (HCl, NaCl, Na_2SO_4), non-corroded, and coated pixels are extracted (see Chapter 5) and a total of $N = 50{,}000$ data instances are generated. The number of spectral bands (features), $q = 448$, and the number of classes, $m = 5$; wherein the class labels C_1, C_2, C_3, C_4 and C_5 are associated with 'Non-corrosion', 'Coating', 'Acid' (HCl corrosion), 'Salt' (NaCl corrosion), and 'Sulfate' (Na_2SO_4 corrosion), respectively. Note that there are 10,000 data instances associated with each class label.

8.2.1. Preprocessing

As mentioned in Section the hyperspectral data consists of information from the neighboring spectral bands that often convey the same information about the object i.e., the features are correlated. Since inclusion of highly correlated features may result in the poor performance of the classifier and higher computational effort [8,9], the correlated adjacent features were grouped and averaged together. For this purpose, Ward hierarchical clustering technique is employed in this study for grouping the features. Ward's method is an agglomerative clustering technique that minimizes the total within-cluster variance. The clusters which result in minimum information loss are then combined. Detailed description of Ward's method could be found elsewhere [10]. With implementation of Ward's method, the number of features in hyperspectral dataset is reduced from 448 to 39 (see Table 8.5).

Table 8.5. Clustering of features.

Feature	Wavelength
1	397.01, 398.32
2	399.63, 400.93
3	402.24, 403.55
4	404.86, 406.17
5	407.48, 408.79
6	410.1, 411.41
7	412.72, 414.03, 415.34
8	416.65, 417.96, 419.27, 420.58
9	421.9, 423.21, 424.52, 425.83, 427.15, 428.46
10	429.77, 431.09, 432.4, 433.71
11	435.03, 436.34, 437.66, 438.97, 440.29, 441.6, 442.92, 444.23
12	445.55, 446.87, 448.18, 449.5, 450.82, 452.13
13	453.45, 454.77, 456.09, 457.4, 458.72, 460.04, 461.36, 462.68, 464, 465.32, 466.64, 467.96
14	469.28, 470.6, 471.92, 473.24, 474.56, 475.88, 477.2, 478.52, 479.85, 481.17, 482.49, 483.81, 485.14, 486.46, 487.78, 489.11, 490.43, 491.75, 493.08, 494.4, 495.73, 497.05, 498.38, 499.7, 501.03, 502.35, 503.68, 505.01, 506.33, 507.66, 508.99, 510.31, 511.64, 512.97, 514.3, 515.63, 516.95, 518.28, 519.61, 520.94, 522.27, 523.6, 524.93, 526.26, 527.59, 528.92
15	530.25, 531.58, 532.91, 534.25, 535.58
16	536.91, 538.24, 539.57, 540.91, 542.24, 543.57, 544.9, 546.24
17	547.57, 548.91, 550.24

Table 8.5. Clustering of features (continued)

Feature	Wavelength
18	551.57, 552.91
19	554.24, 555.58
20	556.91, 558.25
21	559.59, 560.92, 562.26, 563.59, 564.93
22	566.27, 567.61, 568.94, 570.28, 571.62, 572.96, 574.3, 575.63, 576.97, 578.31, 579.65, 580.99, 582.33, 583.67, 585.01, 586.35, 587.69, 589.03, 590.37
23	591.71, 593.06, 594.4
24	595.74, 597.08, 598.42
25	599.77
26	601.11
27	602.45, 603.8, 605.14, 606.48, 607.83, 609.17, 610.52, 611.86, 613.21, 614.55, 615.9, 617.24, 618.59, 619.94, 621.28
28	622.63, 623.98, 625.32, 626.67
29	628.02, 629.37, 630.71, 632.06
30	633.41, 634.76, 636.11, 637.46, 638.81, 640.16
31	641.51, 642.86, 644.21, 645.56, 646.91, 648.26, 649.61, 650.96, 652.31, 653.67, 655.02, 656.37
32	657.72, 659.08, 660.43, 661.78, 663.14, 664.49, 665.84, 667.2, 668.55, 669.91, 671.26, 672.62, 673.97, 675.33, 676.68, 678.04, 679.4, 680.75, 682.11, 683.47, 684.82, 686.18, 687.54, 688.9, 690.25, 784.52, 785.9, 787.27, 788.65, 790.02, 791.4, 792.77, 794.15, 795.52, 796.9, 798.28
33	691.61, 692.97, 694.33, 695.69, 697.05, 698.41, 699.77, 701.13, 702.49, 703.85, 705.21, 706.57, 707.93, 709.29, 710.65, 712.02, 777.66, 779.03, 780.4, 781.78, 783.15
34	713.38, 714.74, 716.1, 717.47, 718.83, 720.19, 721.56, 722.92, 724.28, 725.65, 727.01, 728.38, 729.74, 731.11, 732.47, 766.68, 768.05, 769.42, 770.79, 772.17, 773.54, 774.91, 776.28
35	733.84, 735.2, 736.57, 737.93, 739.3, 740.67, 742.03, 743.4, 744.77, 746.14, 747.5, 748.87, 750.24, 751.61, 752.98, 754.35, 755.72, 757.09, 758.46, 759.83, 761.2, 762.57, 763.94, 765.31
36	799.65, 801.03, 802.41, 803.78, 805.16, 806.54, 807.92, 809.3, 810.67, 812.05, 813.43, 814.81, 816.19, 817.57, 818.95, 820.33, 821.71, 823.09, 824.47, 825.85, 827.23, 828.61, 830, 831.38, 832.76, 834.14, 835.53, 836.91
37	838.29, 839.67, 841.06, 842.44, 843.83, 845.21, 846.59, 847.98, 849.36, 850.75, 852.13, 853.52, 854.91, 856.29, 857.68, 859.06, 860.45, 861.84, 863.23, 864.61, 866, 867.39, 868.78, 870.16, 871.55, 872.94, 874.33, 875.72, 877.11, 878.5, 879.89, 881.28, 882.67, 884.06
38	885.45, 886.84, 888.23, 889.63, 891.02, 892.41, 893.8, 895.19, 896.59, 897.98, 899.37, 900.77, 902.16, 903.55, 904.95, 906.34, 907.74, 909.13, 910.53, 911.92, 913.32, 914.71, 916.11, 917.5, 918.9, 920.3, 921.69, 923.09, 924.49, 925.89, 927.28, 928.68, 930.08, 931.48, 932.88, 934.28, 935.68, 937.08, 938.48, 939.88, 941.28, 942.68, 944.08, 945.48, 946.88
39	948.28, 949.68, 951.08, 952.48, 953.89, 955.29, 956.69, 958.09, 959.5, 960.9, 962.3, 963.71, 965.11, 966.52, 967.92, 969.33, 970.73, 972.14, 973.54, 974.95, 976.35, 977.76, 979.16, 980.57, 981.98, 983.38, 984.79, 986.2, 987.61, 989.02, 990.42, 991.83, 993.24, 994.65, 996.06, 997.47, 998.88, 1000.29, 1001.7, 1003.11, 1004.52

8.2.2. Configuring Feed-Forward Neural Networks

A feed-forward deep neural network (FFDNN) with a configuration of five hidden layers is employed to train the hyperspectral dataset. The details of the configuration is as follows: 1st

HL – 200 neurons, 2nd HL – 100 neurons, 3rd HL – 50 neurons, 4th HL – 30 neurons, and 5th HL – 10 neurons. A tangent hyperbolic (tanh) nonlinear function is used as an activation function for all the neurons in the hidden layers. For the purpose of training, validating, and testing the chosen configurations, the datasets are randomly partitioned into 70:15:15 ratio, respectively. Note that the partition ratio is maintained consistently for each class label, i.e., 70:15:15 of training, validation, and testing data from each class label is chosen. The average accuracy is determined to be 97.4% after repeating the training for 100 times with randomly chosen instances in each iteration (see Table 8.6). From the confusion matrix it can be found that the 8% of 'Salt' is misclassified 'Sulfate' and 5% of 'Sulfate' is misclassified as 'Salt'.

Table 8.6. Confusion matrix of correctly and incorrectly classified corrosion source (FFDNN).

Class		**Actual Class Label**				
		Non-Corrosion	Coating	Acid	Salt	Sulfate
Predicted Label	Non-Corrosion	1	0	0	0	0
	Coating	0	1	0	0	0
	Acid	0	0	1	0	0
	Salt	0	0	0	0.92	0.05
	Sulfate	0	0	0	0.08	0.95

8.2.3. Results

Complex-step sensitivity analysis is performed (see Figure 7.9) and the features responsible for the predictions of class labels (HCl, NaCl, Na_2SO_4, non-corroded, and coated pixels) are determined (see Table 8.7). Furthermore, the KernelSHAP is also implemented, and the important features are obtained.

Table 8.7. Important features identified by SHAP and CS-FA-C (ranked in the descending order of their importance).

	Method	Important Features
Hyperspectral dataset	SHAP	39, 38, 34, 33, 37, 27, 35, 22, 20, 26, 17, 32, 18, 28, 13, 19, 30, 16, 24, 23, 31, 21, 11, 12, 29, 14, 10, 15, 25, 6, 9, 36, 1, 8, 2, 4, 5, 3, 7
	CS-FA	38, 39, 32, 34, 22, 37, 9, 27, 35, 33, 21, 13, 29, 10, 20, 26, 8, 18, 17, 23, 12, 2, 31, 4, 7, 30, 19, 5, 11, 15, 24, 36, 25, 28, 1, 16, 3, 14, 6

From Table 8.7 it is evident that eight among top 10 features are similar for CS-FA-C and KernelSHAP methods. These features are identified as Feature 22 (566nm-590nm), Feature 27 (602nm-621nm), Feature 33 (691nm-712nm, 777nm-783nm), Feature 34 (713nm-732nm, 766nm-776nm), Feature 35 (733nm-765nm), Feature 37 (838nm-884nm), Feature 38 (885nm-946nm) and Feature 39 (948nm-1004nm). Nevertheless, the order of importance of other features are found to vary. Interestingly the irrelevant features (i.e., last few features) are also found to differ except Feature 1 and Feature 3. The spectral bands of top 10 features obtained from CS-FA-C indicates an agreement with the spectral profile shown in Figure 5.6. To further verify the efficacy of the identified top 10 features, the hyperspectral test dataset are employed (see Section 5.3.2).

The prediction ability of the configured FFDNN with inclusion of top 1 feature, top 5 features, top 10 features and all features is demonstrated for the hyperspectral test image dataset. Note that there are three test datasets namely ‘Acid’, ‘Salt’ and ‘Sulfate’ (see Figure 5.3). The predictions for ‘Acid’, ‘Salt’ and ‘Sulfate’ datasets are shown in Figure 8.3-Figure 8.5, respectively. From Figure 8.3 it is evident that the prediction accuracy improves as the number of features are included in succession for ‘Acid’ test image. When top 10 features are employed the trained FFDNN is found to improve the predictions. Similarly, from Figure 8.4 and Figure 8.5 it is evident that inclusion of top 10 features improves the prediction accuracy of ‘Salt’ and

'Sulfate' test images, respectively. However, with inclusion of all features slight misclassifications were observed in all three test images i.e. the non-corroded portions were misclassified as 'Acid'.

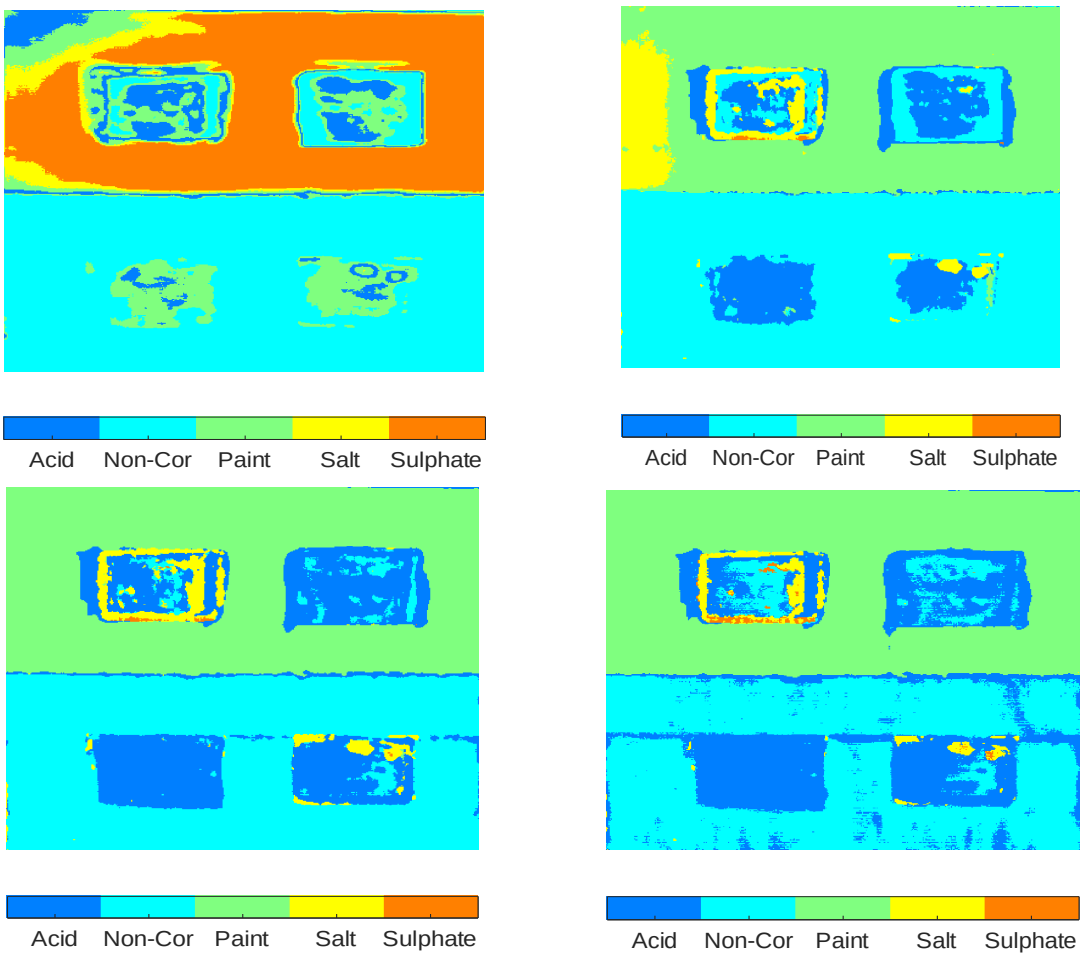


Figure 8.3. Prediction of 'Acid' corrosion using (a) top 1 feature, (b) top 5 features, (c) top 10 features, and (d) all features.

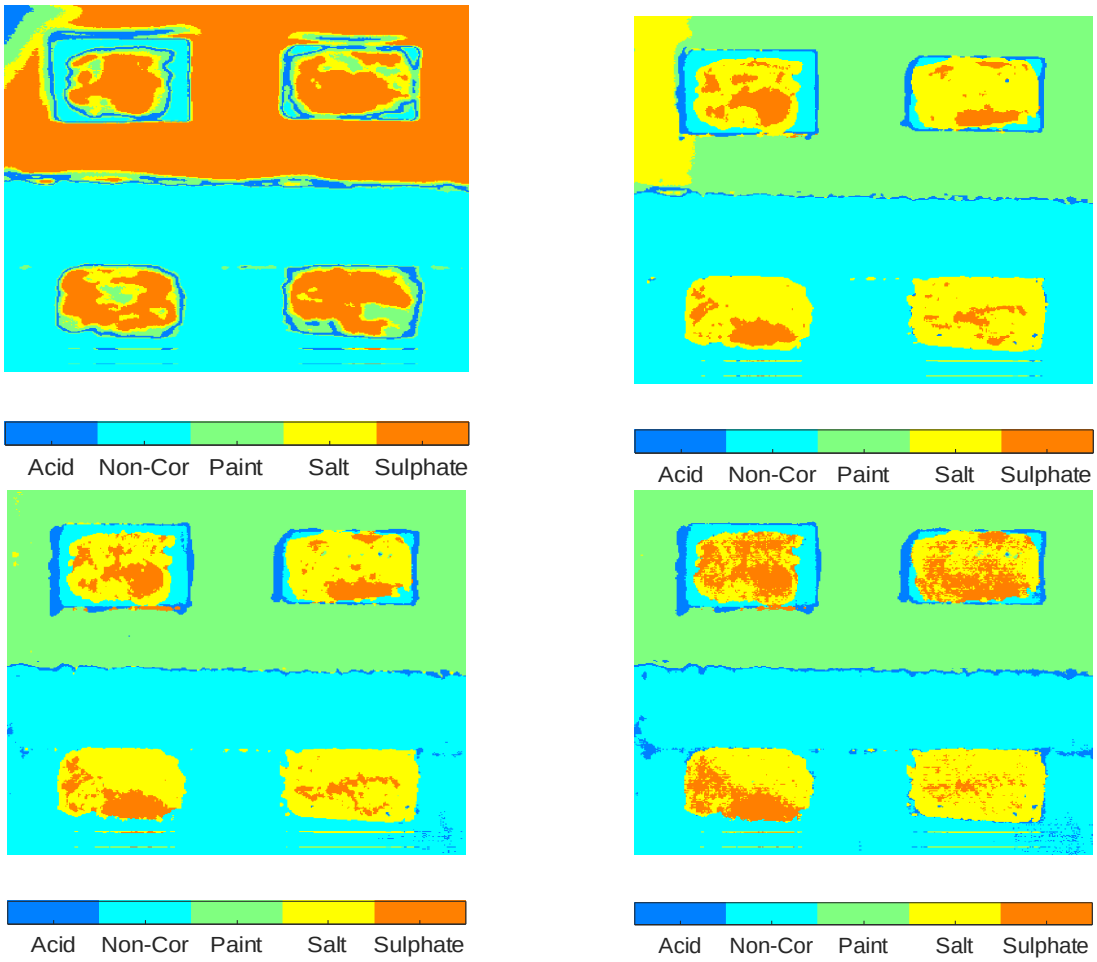


Figure 8.4. Prediction of 'Salt' corrosion using (a) top 1 feature, (b) top 5 features, (c) top 10 features and (d) all features.

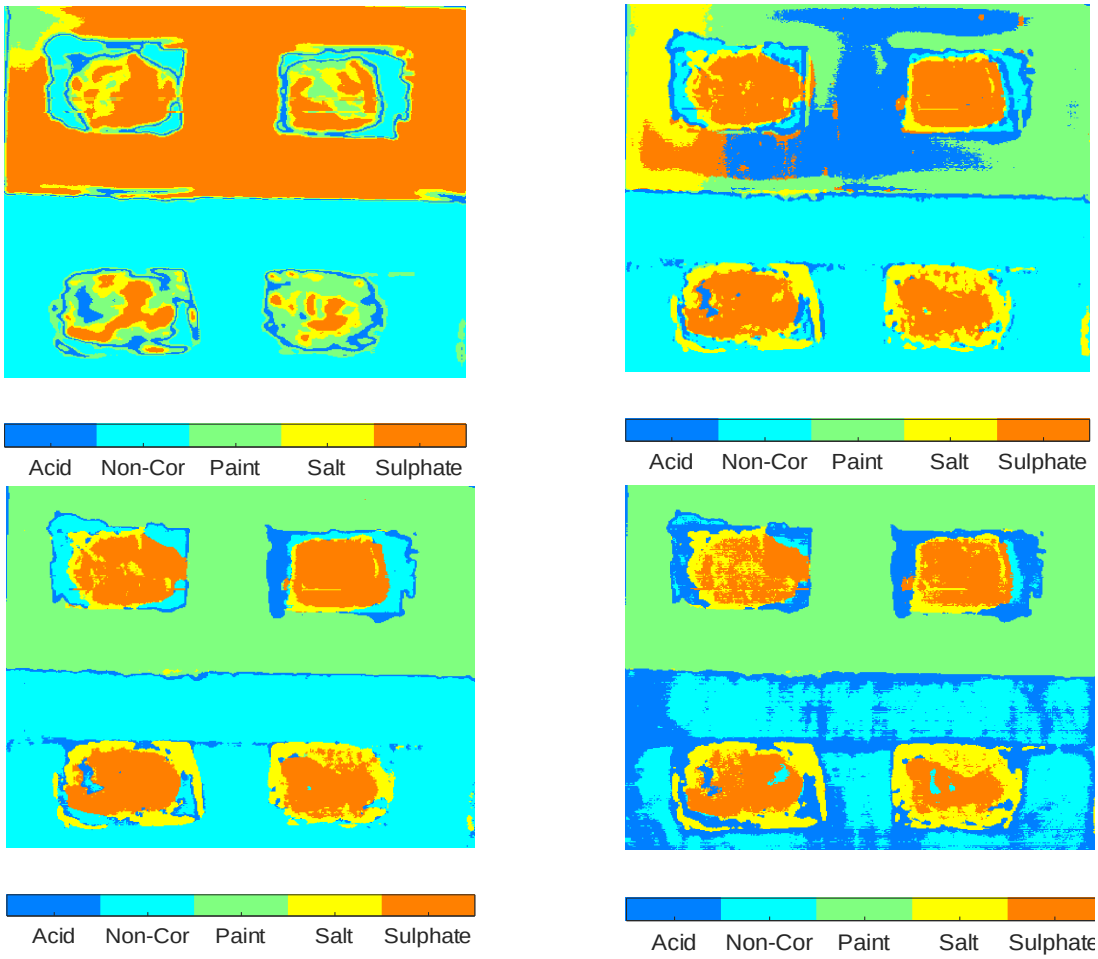


Figure 8.5. Prediction of 'Sulfate' corrosion using (a) top 1 feature, (b) top 5 features, (c) top 10 features and (d) all features.

8.3. Summary

In this study the efficacy of the complex-step sensitivity analysis is illustrated in the framework of feed-forward deep neural networks. For this purpose, the real-world datasets (regression and classification) and hyperspectral dataset are employed. The explanations for FFDNN are generated in the form of feature attribution and important features that contribute to the predictions are identified. With inclusion of each top feature in succession the trend of accuracy (for classification task) was found to increase and the MSE (for regression task) was found to decrease for all real-world datasets. In the case of hyperspectral dataset, the predictions are obtained on the test images ('Acid', 'Salt' and 'Sulfate') for top 1, 5, 10 features and all

features. The results revealed that the predictions improved with inclusion of more features and found to be accurate for top 10 features. For the sake of comparison, other popularly used XAI technique KernelSHAP is also employed. The results obtained from the proposed complex-step sensitivity analysis and KernelSHAP were found to be similar.

8.4. References

1. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/index.php (accessed April 7, 2021).
2. Johnson RW. Fitting Percentage of Body Fat to Simple Body Measurements. J Stat Educ 1996;4. https://doi.org/10.1080/10691898.1996.11910505.
3. Nash WJ. 7he Population Biology of Abalone (_Haliotis_ species) in Tasmania. I. Blacklip Abalone (_H. rubra_) from the North Coast and Islands of Bass Strait. 1994.
4. Cortez P, Cerdeira A, Almeida F, Matos T, Reis J. Modeling wine preferences by data mining from physicochemical properties. Decis Support Syst 2009;47:547–53. https://doi.org/10.1016/j.dss.2009.05.016.
5. Siebert JP. Vehicle Recognition Using Rule Based Methods; 1987.
6. Mangasarian OL, Street WN, Wolberg WH. Breast Cancer Diagnosis and Prognosis Via Linear Programming. Oper Res 1995;43:570–7. https://doi.org/10.1287/opre.43.4.570.
7. Ding B, Qian H, Zhou J. Activation functions and their characteristics in deep neural networks. Proc. 30th Chinese Control Decis. Conf. CCDC 2018, Institute of Electrical and Electronics Engineers Inc.; 2018, 1836–41. https://doi.org/10.1109/CCDC.2018.8407425.
8. Datta A, Ghosh S, Ghosh A. Unsupervised band extraction for hyperspectral images using clustering and kernel principal component analysis. Int J Remote Sens 2017;38:850–73. https://doi.org/10.1080/01431161.2016.1271470.

9. Su J, Yi D, Liu C, Guo L, Chen W-H. Dimension Reduction Aided Hyperspectral Image Classification with a Small-sized Training Dataset: Experimental Comparisons https://doi.org/10.3390/s17122726.

10. Rencher AC., Christensen WF. Methods of Multivariate Analysis: Wiley; 2009.

9. CONCLUSIONS AND FUTURE WORK

In this chapter, a brief summary, the conclusions, and the possible future work for each research objective are provided.

9.1. Metallurgical Phase Identification

In this dissertation, a supervised machine learning approach is proposed to automatically identify the metallurgical phases in heat-treated steels, namely ferrite, pearlite, and martensite. Unlike traditional techniques wherein pixel intensity is used as a descriptive feature for identifying the metallurgical phases, textural features are employed in this study along with the pixel intensity. Five different window sizes are considered for extracting the textural features of each metallurgical phase. The influence of each window size and the textural features are investigated, and the most suitable window size and the important features are determined.

9.1.1. Conclusions

Among the 20 descriptive features, ‘pixel intensity’, ‘maximum probability’, ‘auto-correlation’, ‘sum of squares’ ‘cluster shade’, ‘sum variance’, ‘sum average’ and ‘energy’ are found to be the most relevant features for all five window sizes. Unlike the threshold-based segmentation approaches, the proposed approach was found to avoid the misclassification of grain boundaries into pearlite. Interestingly, the proposed approach does not require the end-user to input the number of metallurgical phases present in the microstructure, which is advantageous when investigating new microstructures. Based on the current study, the following two recommendations are provided: (1) a sufficient number of data points must be acquired (under similar conditions) to train the classifier, and (2) an optimal window size must be determined in conjunction with the subset of relevant features for accurate prediction of metallurgical phases.

9.1.2. Future Direction

While only heat-treated ASTM A36 structural steel is considered in this study, the methodology could be implemented on the microstructures of other metals too. Further, for a case wherein the label of the phase may be unavailable, the unsupervised algorithms could be used in conjunction with textural features to cluster/segment the similar metallurgical phases.

9.2. Fracture Type Identification

Automatic identification of the brittle and ductile fracture regions in fractographic images of structural steel with varying grayscale and illumination is developed in this dissertation. For this purpose, a textural feature extraction algorithm, Local Binary Pattern (LBP), is employed in conjunction with the supervised machine learning classifier Linear Discriminant Analysis (LDA). Both block-wise and pixel-wise classification is performed on the test images.

9.2.1. Conclusions

Textural features extracted using LBP were found to identify the types of fracture accurately. While features/patterns ‘0’ and ‘9’ were determined to important for brittle fracture, features/patterns ‘4’ and ‘5’ were identified to be important for ductile fracture. Physically the features/patterns ‘0’ and ‘9’ signify the presence of random river-like patterns, and features/patterns ‘4’ and ‘5’ signify the presence of edges. When compared to the block-wise classification, pixel-wise classification resulted in more accurate classification. However, block-wise classification is computationally inexpensive, and the area fractions obtained from both methods are more or less the same.

9.2.2. Future Directions

This methodology could be extended to identify fatigue fractures from fractographic images. Fatigue fractures are characterized by the presence of striations (high-cycle fatigue) or

elongated cups and cones (ultra-low cycle fatigue) at microscale exhibiting a distinct texture. However, a sufficient number of fatigue fracture training images are required to train ML algorithms to identify fatigue fracture in addition to the brittle and ductile fracture in fractographic images.

9.3. Detection of Corrosion and its Source

Color spaces in conjunction with different MLP configurations are explored to detect corrosion initiation in steel structures under ambient lighting conditions. To this end, sixteen different combinations of color spaces and MLP configurations are explored. The performance of each combination is then assessed through the validation dataset obtained from lab-generated images, and the best combination is determined. Subsequently, the obtained combination is deployed on the test image database, and the efficacy of trained MLP to detect corrosion in real-world scenarios is demonstrated. Furthermore, the use of hyperspectral images for the elimination of visual ambiguity in corrosion detection and identification of corrosion source (‘Acid’, ‘Slat’ and ‘Sulfate’) is also demonstrated. Hyperspectral data of ‘Non-corrosion’, ‘Coating’, ‘Acid’, ‘Salt’ and ‘Sulfate’ labeled specimens are acquired in the VNIR range of the EM spectrum and are used to train, test and validate the SVM classifier.

9.3.1. Conclusions

The combination of ‘rgb’ color space and an MLP configuration of a single hidden layer with 4 neurons (1^{st} HL (4N)) yielded the highest accuracy for corrosion detection. Improved accuracy in the case of ‘RGB’ color space can be attributed to the increased non-linearity of the decision boundary generated by the MLP, which will ultimately lead to overfitting issues. Under shadows and wetting conditions, the trained MLP is still found to yield correct predictions when ‘rgb’ color features are used. Especially, the detection of corrosion in the bottom side of the deck

of a bridge under dark shadows is noteworthy. Interestingly, the proposed method is insensitive to the camera sensor employed for the image acquisition, i.e., irrespective of images being acquired from a mobile camera or a DSLR camera, the efficacy of trained MLP to detect corrosion was not affected. MLP trained on varying illumination dataset alone is sufficient for detecting the corrosion under shadows and wetting conditions.

However, for eliminating the visual ambiguity between the coatings and corroded surfaces and identifying the corrosion source, the VNIR spectra are employed. In this study, the top two principal components of the reflectance of VNIR spectra, along with an SVM, are used to eliminate visual ambiguity between the coating and corroded surfaces. The source of corrosion, i.e., 'Acid', 'Salt' and 'Sulfate', is also identified using the top two principal components of the reflectance of VNIR spectra. The trained SVM classifier was able to identify the source of corrosion accurately. However, slight misclassifications were observed in the case of 'Salt' and 'Sulfate' data. Misclassifications of 'Salt' and 'Sulfate' class labels may be attributed to the presence of similar iron oxide corrosion products. XRD characterization tests revealed the presence of similar corrosion product Goethite on both 'Salt' and 'Sulfate' corroded surfaces which may have led to the confusion. Among the 448 spectral bands that were acquired from the hyperspectral images, only a few spectral bands were found to play an important role in the identification of corrosion sources and the elimination of visual ambiguity. The important ranges of the wavelengths of the spectral bands identified for the classification of coating and corroded surfaces are 500-520 nm, 660-680 nm, 760-770 nm, and 830-850 nm.

9.3.2. Future Directions

The proposed method does not account for the detection of other corrosion types such as pitting corrosion, crevice corrosion, etc. Agglomeration of other feature extraction techniques

and ML algorithms could be explored to distinguish the type of corrosion and its severity. Corrosion of steel depends on so many other factors which in turn governs the formation of different oxidation products—for example, the availability of oxygen, humidity, pH value, etc. More information about the corrosion chemistry could be incorporated into ML algorithms to characterize the corrosion products. Hyperspectral data for an extended period of corrosion should also be taken into account since the color appearance, and oxidation products may change. Performing SEM, EDX on the corroded surface would provide supplemental information about the morphology and elemental composition. In a practical scenario, false negatives for corrosion prediction should be minimized. Performing other spectroscopy analyses such as FTIR may provide more information that could be linked with the hyperspectral (VNIR) data.

Push broom hyperspectral imaging systems are expensive and are primarily developed for benchtop applications limiting their use in field applications. Building a customized multispectral imagining sensor with the ability to capture spectral information in the desired range of wavelengths may be a feasible and economical option. The multispectral sensors are portable and hence could be easily mounted on UAVs for easy navigation and maneuvers in the field. The key wavelengths identified in this study can be used to build a multispectral imagining sensor that can eliminate the visual ambiguity and detect the chemically distinctive corroded surfaces in civil, structural, aerospace, and offshore structures.

9.4. Complex-Step Sensitivity Analysis

The complex-step derivative approximation (CSDA) approach is introduced in the context of deep neural networks to perform the sensitivity analysis. The implementation of the proposed method is described, and the explanations for neural network predictions are generated

for regression and classification tasks. To verify the efficacy of the proposed method, the real world datasets and hyperspectral datasets are employed.

9.4.1. Conclusions

The first-order derivatives evaluated for FFDNN using the CSDA approach were found to eliminate the subtractive cancellation errors and yield analytical quality first-order derivatives. Unlike the existing technique in which backpropagation is required to determine the first-order derivatives, the proposed method incorporates only feed-forward operation. The feature attribution-based explanations generated for real-world datasets are found to be comparable to that of KernelSHAP technique. In other words, the top-most important features and the irrelevant features were found to be similar for both complex-step sensitivity analysis and KernelSHAP. Furthermore, the proposed method, when implemented on the hyperspectral dataset, resulted in the identification of the top 10 spectral bands yielding accurate predictions of corrosion source.

9.4.2. Future Directions

Besides elementary effects, the interaction of features may also play a vital role in the prediction models. The proposed methods' scope could be extended to determine the first-order interaction effects. Further, to reduce the computational time the parallel computing could also be incorporated. On the other hand, the implementation of the complex-step sensitivity analysis in the framework of convolutional neural networks could also be explored.