DEVELOPMENT OF A DISTRIBUTED BIG DATA FUSION ARCHITECTURE FOR MACHINE-TO-MACHINE COMMUNICATION USING ENSEMBLE LEARNING

BY

Ngbede Odaudu SALEFU

P15EGCP8083

DEPARTMENT OF COMPUTER ENGINEERING

FACULTY OF ENGINEERING

AHMADU BELLO UNIVERSITY, ZARIA

NIGERIA

FEBUARY, 2019

DEVELOPMENT OF A DISTRIBUTED BIG DATA FUSION ARCHITECTURE FOR MACHINE-TO-MACHINE COMMUNICATION USING ENSEMBLE LEARNING

BY

Ngbede Odaudu SALEFU BSc (2010), PGDCP (2015) ABU

P15EGCP8083

ngbede4@gmail.com

A DISSERTATION SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES,

AHMADU BELLO UNIVERSITY, ZARIA

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF

MASTER OF SCIENCE (M.Sc.) DEGREE IN COMPUTER ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

FACULTY OF ENGINEERING

AHMADU BELLO UNIVERSITY, ZARIA

NIGERIA

FEBUARY, 2019

DECLARATION

I Ngbede Odaudu SALEFU declare that the work in this project report/thesis/dissertation entitled_ DEVELOPMENT OF A DISTIBUTED BIG DATA FUSION ARCHITECTURE FOR MACHINE TO MACHINE COMMUNICATION USING ENSEMBLE LEARNING_ has been performed by me in the Department of Computer Engineering Ahmadu Bello University, Zaria. The information derived from the literature has been duly acknowledged in the text and a list of references provided. No part of this project report/thesis/dissertation was previously presented for another degree or diploma at this or any other Institution.

Odaudu, N. SALEFU _____ _____

Signature Date

CERTIFICATION

This Dissertation entitled DEVELOPMENT OF DISTRIBUTED BIG DATA FUSION ARCHITECTURE FOR MACHINE TO MACHINE COMMUNICATION USING ENSEMBLE LEARNING by ODAUDU, Ngbede Salefu meets the regulations governing the award of the degree of Master of Science (MSc) in Computer Engineering of the Ahmadu Bello University and is approved for its contribution to knowledge and literary presentation.

Dr. I. J. Umoh Chairman, Supervisory Committee	(Signature)	Date
Dr. E. A. Adedokun Member, Supervisory Committee	(Signature)	Date
Prof. M. B. Mu'azu Head of Department	(Signature)	Date
Prof. S. Z. Abubakar Dean. School of Postgraduate Studies	(Signature)	Date

DEDICATION

This research work is dedicated to my family and to the Church of Jesus.

ACKNOWLEDGEMENT

To Him that sit upon the throne be Glory, Honor, Blessing, Power, Dominion, Wisdom, Might and Thanksgiving forever Amen.

My profound appreciation goes to the Almighty God who is the creature of all Being and the Judge of both the living and the dead. Lord, I am eternally grateful.

I had love to use this medium to appreciate everybody God has used to see me through to this end. My warm gratitude is lavish on my team of supervisors Dr. I. J. Umoh and Dr. E.A. Adedokun chairman and member supervising committee and also, Prof. M. B. Mu'azu for their supports, encouragement and guidance in the course of this research.

To the academic staff, non-academic staff and research students of the department of Computer Engineering Ahmadu Bello University, Zaria amongst whom are my lecturers; Dr. T. H. Sikiru, Dr. B.O. Sadiq, Dr. A. T. Salawudeen, Dr. A. Sha'ban, Dr. A.I. Abdulrazak, Engr. I. Yusuf, Engr Shehu to mention but few I remain grateful.

My gratitude goes to every member of Control and Computer research group for the love, interest, correction and constructive criticisms during each of my appearance before the group.

I acknowledge with thanks the contribution of my friends, colleagues and classmates whose list is endless, I am glad I identified with you all.

This page won't stop scrolling if I fail to acknowledge those that matter to my heart the most especially my family. To my elder brother and his family Mr. & Mrs. Salefu O.A., thank you for believing in me and for the investment of time, encouragement, resources, prayers and love I say God bless you real good sir. My elder Sister, Fatima Usman (Mrs.), Mr. Abdulahi Salefu, Hadiza Salefu (Mrs.), and Mr. Salefu Felix Ahassan (late) wish you are here. My lovely mother, Mrs.

Salefu Zainab, mummy thanks for the prayers. In the same vain, I salute Adejoh Ella Nancy, Walter Inalegwu Salefu, the church of Jesus (RCCG, Restoration Assembly) and a host of others for their patience, love, prayers and faith in me.

Salefu Ngbede Ernest ODAUDU

September, 2018

ABSTRACT

This research developed a distributed big data fusion architecture for machine to machine communication using ensemble learning. This is implemented to mitigate the challenges that characterize centralized big data fusion architecture commonly adopted through the use of Hadoop MapReduce platform. These challenges include bandwidth consumption, latency, and high computational cost. Fog computing technique approach was adopted through the implementation of ensemble learning; feature engineering was implemented to extract information (pixel values, number of layers (nlayers), number of cell (ncell), number of row (nrow), and coordinates) from the data, water bodies and vegetation index (NDWI and NDVI) were calculated. The extracted information was used as a training dataset for both centralized and distributed architecture using adaboost as bases of comparison between centralized and distributed architecture. Performance evaluation was based on Bandwidth consumption and Latency. Results were presented in the form of confusion matrix. The developed architecture achieved a 31.44 minutes and 1.9% improvement in latency and accuracy between the centralized and the distributed architecture respectively. The result also showed 5.8% and 4.81 minutes improvement in accuracy and latency were recorded in performance comparison of base learner and ensemble Adaboost.

TABLE OF CONTENTS

DECLARATION **III**

CERTIFICATION **IV**

DEDICATION **V**

ACKNOWLEDGEMENT **VI**

ABSTRACT **VIII**

TABLE OF CONTENTS **IX**

LIST OF FIGURES **XIV**

LIST OF TABLES **XVI**

LIST OF ABBREVIATIONS **XVII**

CHAPTER ONE: INTRODUCTION

1.1 Background of Study 1

1.2 Statement of Problem 3

1.3 Aim and Objectives 4

1.4 Significance of Research 4

1.5 Scope of Study 4

CHAPTER TWO: LITERATURE REVIEW

2.1 Introduction 7

2.2 Review of Fundamental Concepts 7

2.2.1 Big data 7

2.2.2 Big data processing and management. 9

2.2.3 Big data fusion architecture 10

2.2.3.1 Centralized big data fusion architecture (CBDFA) 12

2.2.3.2 Distributed big data fusion architecture (DBDFA) 13

2.2.4 Hadoop Architecture and Design 14

2.2.4.1 Hadoop distributed file system 14

2.2.4.2 MapReduce 15

2.2.5 The U.S. Joint directors of laboratories data fusion model 16

2.2.6 Fog Computing 19

2.2.7 Fog Computing System Architecture 22

2.2.8 Machine-to-Machine Communication 22

2.2.9 Ensemble Learning System 24

2.2.9.1 Combining Classifiers. 25

2.2.9.2 Combining class labels 25

2.2.9.3 Voting based fusion techniques 26

2.2.9.4 Majority voting 26

2.10 Algorithms 27

2.10.1 Boosting 27

2.10.2 Stack Generalization 29

2.10.3 K- Means 31

2.10.4 Haar-like feature 32

2.11 OpenCV 33

2.12 Weka 33

2.13 Classification 34

2.13.1 NDVI and NDWI 35

2.13.2 Dataset 36

2.13.3 System specification 37

2.13.4 Storage **Error! Bookmark not defined.**

2.3 Review of similar works 37

CHAPTER THREE: MATERIALS AND METHODOLOGY

3.1 Introduction 44

3.2 Materials 44

3.2.1 Software 44

3.2.2 Hardware 45

3.3 Methodology 45

3.3.1 Development of Centralized Big data Fusion Architecture 46

3.3.1.1 Data Acquisition 47

3.3.1.2 Pre-Processing 47

3.3.1.3 Data reduction 48

3.3.2 Feature engineering 49

3.3.2.1 Load data. 50

3.3.2.2 Extract Pixel values. 51

3.3.2.3 Extract image coordinate and layers in X, Y table. 51

3.3.2.4 Crop and extract pixel values of Region of Interest (RoI) 52

3.3.2.5 Extract NDVI and NDWI 52

3.3.3 Implementation of U.S. Joint Director of Laboratory 53

3.3.3.1 Object Alignment 53

3.3.3.2 Object Refinement 53

3.3.3.3 Object Identification and Detection using Haar-like feature (Adaboost) Algorithm. 54

3.4 Development of Distributed Big Data Fusion Architecture using Ensemble Learning. 55

3.4.1 Data Description: 56

3.4.2 Data Pre-processing: 56

3.5 Classification 57

3.5.1 WEKA implementation of Ensemble Learning (Adaboost) 57

3.6 Performance Evaluation 59

3.6.1 Bandwidth 59

3.6.2 Latency 59

3.6.3 Confusion Matrix 60

CHAPTER FOUR: RESULTS AND DISCUSSION

4.1 Introduction **Error! Bookmark not defined.**

4.2 Centralized Data Fusion Architecture 61

4.2.1 Data pre-processing 61

4.2.1.1 Data Visualization 62

4.2.1.2 Principal component Analysis 62

4.2.1.3 Satellite image Bands Extraction 63

4.2.1.4 Implementation of NDVI and NDWI 64

4.2.2 Classification 65

4.3 Analysis of centralized data fusion implementation 66

4.3.1 Adaboost (Haar) 66

4.3.2 Confusion matrix 69

4.4 Results from Distributed Big Data Fusion Architecture Using Ensemble Learning. 70

4.4.1 Data pre-processing 70

4.4.1.1 Statistical parameter 71

4.4.1.2. Data Visualization using WEKA. 72

4.4.2 Implementation of Ensemble Learning Adaboost 73

CHAPTER FIVE: CONCLUSION SUMMARY AND RECOMMENDATION

5.1	Conclusion	77
5.2	Significant contributions	77
5.3	Recommendations for further work	77
5.4	Limitation	78
REFERENCES		**79**
APPENDICES		**87**

LIST OF APPENDICES

Appendix A 83

Appendix B 120

LIST OF FIGURES

Figure 2.1: Flow Chart of Big Data Fusion Architecture (Ahmad *et al*., 2016). 11

Figure 2.2 Centralized Data Fusion Architecture (Natale, 2015) 12

Figure 2.3: Distribute big data fusion Architecture (Natale, 2015). 13

Figure 2.4: The relationship between nodes (Uzunkaya *et al.,* 2015) 15

Figure 2.5: MapReduce Architecture (Uzunkaya *et al*., 2015) 16

Figure 2.6: A simple M2M Architecture (Sneps-Sneppe & Namiot, 2012) 24

Figure 2.7: Stacked generalization (Rokach, 2005) 30

Figure 3.1:WEKA interface 45

Figure 3.2: Landsat8 satellite images (USGS, 2018**)** 47

Figure 3.3: Feature engineering flow chat 50

Figure 3.4: Process flow chart for developing a Distributed Big Data Fusion Architecture. 55

Figure 3.5: WEKA Interface for data pre-processing 57

Figure 3.6: Snapshot of the implementation of Boosting single and ensemble learning 58

Figure 4.1: De-noised satellite image using median Gaussian filter algorithm. 62

Figure 4.2: Plot of Principal Component Analysis 63

Figure 4.3: Plot of image visualization in Rstudio 64

Figure 4.4: Plot of NDVI and NDWI feature 65

Figure 4.5: Feature clustering using K-means 66

Figure 4.6: Adaboost correctly identified water body. 67

Figure 4.7: Features incorrectly identified. 68

Figure 4.8: Hadoop implementation of feature detection and identification 68

Figure 4.9: Plot of Confusion Matrix 69

Figure 4.10: WEKA Implementation of Principal Component Analysis 71

Figure 4.11: Statistical parameters 72

Figure 4.12: Data Visualization 73

Figure 4.13: Boosting; time of execution 74

Figure 4.14: Comparison of Adaboost implementation on centralized and distributed architecture.

75

LIST OF TABLES

Table 1.1: Tradeoff for different levels of data fusion (Esteban *et al.*, 2005) 2

Table 2.1: Landsat8 digital information (USGS, 2018) 36

LIST OF ABBREVIATIONS

Acronym	Meaning
BDF	Big Data Fusion
CBDFA	Centralized Big Data Fusion Architecture
GUI	Graphic Users Interface.
JDL	Joint Directors of Laboratory
M2M	Machine to machine
NDVI	Normalized Difference of Vegetation Index
NDWI	Normalized difference of Water Index
RDMS	Relational Data Management System
ROI	Region of Interest
SQL	Structured Query Language
USGS	US Geological survey
WEKA	Waikato Environment for knowledge Analysis

CHAPTER ONE

INTRODUCTION

1.1 Background of Study

Data fusion is a recognized structure used to express the union of data from various sources in which its communication imparts the methods and instruments for the cooperation of data that originated from different sources (Akita 2002; Wald, 1998). This approach is centered on the synergy offered by the various sources. The overall idea is to increase training data for a given learning problem using data obtained from other associated problem so as to learn from multiple sources. The process is useful in a situation where the information contained in the single source is partially relevant to the problem of interest. Having this knowledge, it is advantageous to extract complementary information from scores of data source. Data fusion is also not profitable when different data source available for a given problem are incomplete and noisy. The fundamentals of data fusion are to focus on increasing the accuracy of classification thereby increasing the chances of getting valuable information (Wald, 1998). Defined data fusion as a formal framework for combining different data obtained from a different source with the sole aim of obtaining information of superior quality which necessarily be correlated with the production of more relevant information (Wald, 1998). The main reason that necessitates the development of data fusion is the increasing requirement for pertinent information, a robust procedure to manage data efficiently and reliability and performance.

Data fusion finds relevance in science, engineering, biometric, finance, governance, remote sensing and geographic data, wireless sensor network amongst others. The process of combining information from several other sources are categorized into lower level data fusion, intermediate data fusion, and high-level data fusion. The categorization, however, is dependent on the various

stage in which data is fused (Esteban *et al.,* 2005). The Low-level fusion also known as raw data fusion is the combination of raw data from multiple origins into new raw data that should be explanatory, while intermediate level or feature, is the fusion of data of different origin (Esteban *et al.,* 2005). The intermediate level of fusion requires the extraction of a different feature from the source of raw data to be aggregated into a unified feature. Hence this level can be referred to as fusion that is based on feature extraction which is object-dependent while the high level is a fusion approach that depends on decision thereby combining result from different algorithm to reach a final fused decision. In the field of research, several approaches to this method of fusion exist such as statistical method, majority voting, and ensemble method amongst others. Consequently, there is a certain trade-off associated with each level of fusion. The chief of which is choice of the most appropriate fusion level for the data available. Other identified cons are heightened in the table below (Esteban *et al.,* 2005).

Table 1.1: Tradeoff for Different Levels of Data Fusion (Esteban *et al.,* 2005)

Parameters	**Data**	**Feature**	**Decision**
Bandwidth	Possibly very large	Medium	Very small
Information	No loss	Same	Medium
Performance	No loss	Some	Some
Operational complexity	High	Medium	Low

Several data fusion architectures have been developed and implemented. However, in the multisensor system, the purpose is to extract information from several available sources using an

appropriate method like Bayesian inference, ensemble learning, Dumpster-Shafer evidence theory, fuzzy logic and neural network (Alsheikh, 2014).

Big Data Fusion (BDF) learn and update data models as they process data. This holds for both static and real-time data. At each level of processing, the models are updated and discovered relations or patterns are applied almost immediately to the fusion task (Wheeler, 2014). Cloud computing requires the data to be directly preloaded in the data centre before a user runs its application in the cloud. This architecture simply is one that operates based on the parallelism of federated systems/server communicating and performing a given task (Abhishek & Benjamin, 2012; Calheiros, *et al.*, 2011). Demand and needs for optimizing network resources economically and technical cost of managing data and IT infrastructure remotely necessitated cloud computing technique. BDF has been implemented in cloud computing architecture as reported in several kinds of literature (Abhishek & Benjamin, 2012; Calheiros *et al*., 2011; Calheiros, 2011). This process account for large data management, processing, storage, analytic and fusion. Major Service provider of cloud computing includes Microsoft, Facebook, Google, Amazon (Evans, 2011) managing millions of gigabytes of data generated daily by billions of network-enabled devices connected to the internet thereby birthing an era of big data, Internet of
Things (Sneps-Sneppe & Namiot, 2012) and machine to machine communication (M2M) (X.-W. Chen & Lin, 2014; Evans, 2011).

1.2 Statement of Problem

Popular centralized big data fusion employs the use of Hadoop MapReduce architecture which is characterized by large latency for communication between network devices and communication bandwidth. These latency and bandwidth is not computationally cost effective. Therefore, it is

imperative to developed a distributed architecture that will fuse big data in manner that the processing and fusion is done closer to the point where it is generated thereby mitigating the challenges observed in centralized architecture.

1.3 Aim and Objectives

The aim of this research is develop a distributed big data Fusion Architecture for machine-to-machine (M2M) communication using Ensemble Learning.

The objectives of this research are as follows:

1) To develop a centralized big data fusion architecture.
2) To develop a distributed big data fusion Architecture leveraging on Fog Computing technique using ensemble learning.
3) To evaluate the performance of the developed architectures using Accuracy and latency as performance metrics by:

   I. Comparison of the performance of Adaboost in centralized data fusion architecture and Adaboost in distributed data fusion architecture.

   II. Comparison of performance of AdaBoost in distributed Architecture and Ensemble Adaboost on distributed Data fusion Architecture.

1.4 Significance of Research

The significance of this research is to ensure that data is fused distributed at the edge using fog computing techniques in a distributed architecture. To demonstrate the use of Weikato Environment for Knowledge Analysis (Weka) as a tool for data fusion through the use of ensemble methods.

1.5 Scope of the Study

The scope of this study is to use WEKA data mining suit to implement distributed data fusion architecture using one terabytes of spatial image data.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter comprises of the review of fundamental concepts and review of similar works.

2.2 Review of Fundamental Concepts

In this section, concepts that are fundamental to this research work such as big data, Hadoop, MapReduce, U.S Joint Directors of Laboratories data fusion model, M2M communication, Fog Computing and Ensemble Learning System were reviewed.

2.2.1 Big Data

The combination of the big data and internet of things paradigm is the most successful shift in business intelligence. Thereby, changing the way we do business, and increasing productivity and profitability (Dofe *et al.,* 2016). The concept of big data has existed for over a decade but remained within the domain of advance university research and national laboratories. The epoch of big data was announced by Cisco to be a time of business intelligence as reported in (Wu *et al.,* 2014). At the moment, the different company have aligned to this paradigm of data revolution thereby resulting into data of about 2.5 quintillion bytes of data (1 exabyte = I quintillion and 1 exabyte = 1 billion gigabytes) is generated daily from different sources. These data are critical to Business, governance, industry and education in decision making and planning, technology, and the process of handling big data embody collection. Storage, transportation and exploitation. The process of collection, storage and transportation are the important antecedents for mining data using certain analytical tools (Ting *et at.,* 2014).

Big data lack a generalized definition (Chen & Lin, 2014). Most researchers defined big data as any data that is beyond the handling of the traditional Relational Database management System (RDMS) such as SQL (Chen *et al.,* 2014). Typically, big data includes massive unstructured data which are generated at a very fast speed and in a different form. Also, big data can be defined based on its characteristic which are referred to as the 4vs (volume, velocity, variety and veracity) of big data (Emani *et al.,* 2015).

a. **Volume:** Data Volume represents the size of data available to an organization, which does not have to belong to the organization provided it is accessible. The accumulated data consist of both structure, semi-structured and unstructured data type of different origin (Walunj Swapnil K, 2016). It is the granular idea of the data that is extraordinary. Big Data requires handling high volumes of low-density data, that is, data of obscure (unknown) value, for example, twitter data feeds, click on a website page, organize an activity, sensor-empowered hardware catching data almost instantaneously, and some more. It is the task of Big Data to change over low-density data into high-density data, that is, data that has value. For a few organizations, this may be several terabytes, for others, it might be many petabytes (Sagiroglu & Sinanc, 2013)
b. **Velocity:** data velocity as the name implies is a means to gauge the speed of data produced, streaming and arithmetic operations (Walunj Swapnil K, 2016). Also, it's the fast rate that data is gotten and acted upon. The highest velocity data regularly streams straightforwardly into memory as opposed to being composed to disk. Some Internet of Things applications have health and security ramification that require real-time assessment and activity (Sneps-Sneppe & Namiot, 2012). Other web-empowered brilliant devices to work in real-time or close

realtime. Operationally, mobile application encounters have huge client populaces, expanded network traffic, and the desire for prompt reaction (Sagiroglu & Sinanc, 2013).

c. **Variety:** Data Variety represents the way and manner in which the various type of data is represented in the database- text, video, audio, video etc. because of the heterogeneity nature of big data source, big data platform must be able to analyze multiple categories of data stream so as to effectively carry out mining to uncover hidden pattern (Sagiroglu & Sinanc, 2013).

d. **Value:** Data has a hidden value—however, it must be found. This value is what define the utility properties of big data. There is a scope of quantitative and investigative procedures to get value from data – from finding a purchaser inclination or sentiment, to making a significant offer by area. In any case, discovering value likewise requires new revelation forms including smart and clever analyst, business clients, and administrators. The challenges faced in Big data is a human one, which is figuring out how to ask the correct inquiries, recognizing the pattern, making assumptions, and predicting the outcome (Sagiroglu & Sinanc, 2013)

2.2.2 Big data Processing and Management.

Data processing is the unique characteristics of big data as opposed to the 4Vs because it involves discovering hidden patterns in the form of knowledge discovery in database (Georg Krempl *et al.*, 2015). By definition, it is the practice of carrying out an operation on data usually to transform, retrieve, store and classify information. In the case of big data, data generated from multisource are collected, stored and processed. Data can be classified into Real-time data and static Data (Georg Krempl *et al.*, 2015).

Processing big data is a serious challenge because of the voluminous data generated and the speed at which the data arrive in the database this prompt the development of NOSQL a special kind of

database management system specifically for big data (Gao, 2014). The motive for the development was to deliver a database system that is inexpensive, avoid unneeded complexity by providing a variety and complex features and strict data consistency, high throughput because, It does not align to the conventional row and column pattern of (RDMS) making it possible for Google to process about 20 petabytes volume of data via its MapReduce approach, avoidance of expensive object-relational mapping; this characteristics of NoSQL, is that it was deliberately designed to store data structures that are similar in nature and mapping the data into keys and values (Gao, 2014). Owing to its scalability and reliability for better performance, NoSQL has become the needful for cloud computing (Gao, 2014).

2.2.3 Big Data Fusion Architecture

The characterizing processing abilities for big data architecture are to meet the 4Vs prerequisites. Interesting distributed (multi-node) parallel processing architecture has been made to parse these huge datasets. There is the contrasting technological approach for real-time and batch processing storage necessities. For real-time, key-value data stores, for example, NoSQL, take into consideration performance. For batch processing, a procedure known as "Map-Reduce," filter data as indicated by a particular data discovery methodology. After the filtered data is found, it can be dissected specifically, stacked into other unstructured or semi-structured databases, sent to the mobile device, or converged into traditional data warehousing condition and related to structured data (Gao, 2014). Figure 2.1 shows the flowchart of MapReduce procedures for processing batch data.

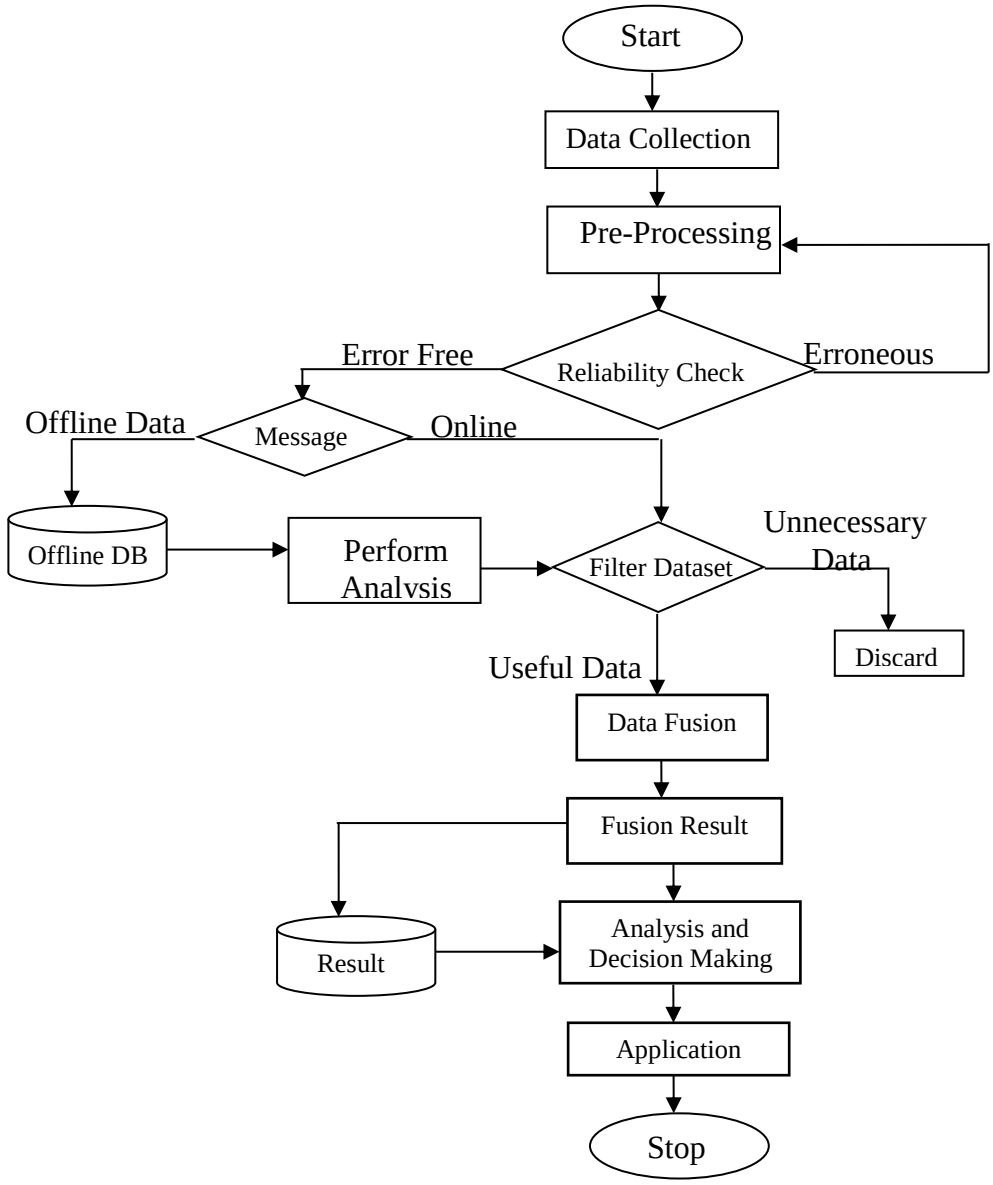


Figure 2.1: Flow Chart of Big Data Fusion Architecture (Ahmad *et al*., 2016).

In batch data processing, data collected is pre-processed to remove noise and ensure even distribution of information in the data then reliability check is carried out on the data to ascertain the purity of the data and with the advent of erroneous or missing information, it undergoes through the pre-processing else the data is saved in database and then fetch thereafter for filtration. The results from these stage of processing is fused (Ahmad *et al.*, 2016). The fusion stage in

MapReduce is known as the reduction process while from the preprocessing, data is broken into blocks and respective block is fed or mapped unto mappers or processing elements, this is done arbitrarily without human intervention, then, information is rearranged through the process called shuffling (Ahmad *et al.,* 2016). Shuffling is based on similarities in features properties of data. Each stage discussed above employed the use and application of machine learning algorithms. Big data processing requires special techniques and tools (Ahmad *et al.,* 2016). Some of the tools, techniques and technologies that are applicable to implementing big data processing are discussed below.

2.2.3.1 Centralized Big Data Fusion Architecture (CBDFA)

A centralized data fusion architecture popularly used among sensors of similar configuration, time synchronization and bias correlation of sensor data, the transformation of the data from sensor based units and coordinates into convenient coordinates and units for central processing (Esteban *et al.,* 2005; Fasano *et al.,* 2015). In this architecture, data from all sensor are first preloaded into a centralized database then processing is done through partitioning of the task amongst federated system architecture as implemented in MapReduce, Apache spark and other related processing tools. Decision at CBDFA rest on the maximum information obtained from a network of sensors (Natale, 2015). This architecture is depicted in Figure2.2.

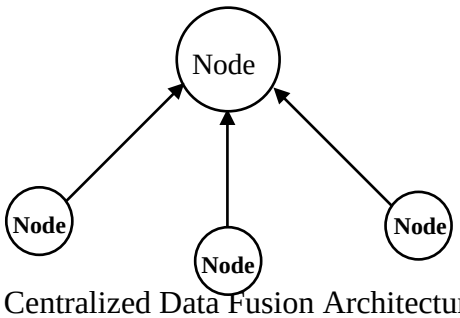


Figure 2.2 Centralized Data Fusion Architecture (Natale, 2015)

2.2.3.2 Distributed Big Data Fusion Architecture (DBDFA)

The architecture is popularly used for a sensor of a dissimilar configuration such as infrared and radar. Data originating from the similar sensor can be processed using this method because it combines data of different coordination and structure (data alignment) within a given location (location awareness) and time in which data was generated (Esteban *et al.,* 2005). This architecture is largely applied to streaming as it necessary not requires that the data must be housed before processing. In distributed Architecture, multi-sensor data are processed by individual processing method depending on the classification, configuration and geospatial information of the data. Such data are locally processed at its processing unit thereby eliminating centralized processing challenges and cost (Esteban *et al.,* 2005). It finds application in large flexible structure, monitoring of aircraft, weather and health-related condition. The choice of Distributed Big Data fusion architecture for the internet of things (Sneps-Sneppe & Namiot, 2012) and fog computing owes to its advantages over centralized architecture (Natale, 2015). The architecture is represented in Figure 2.3.

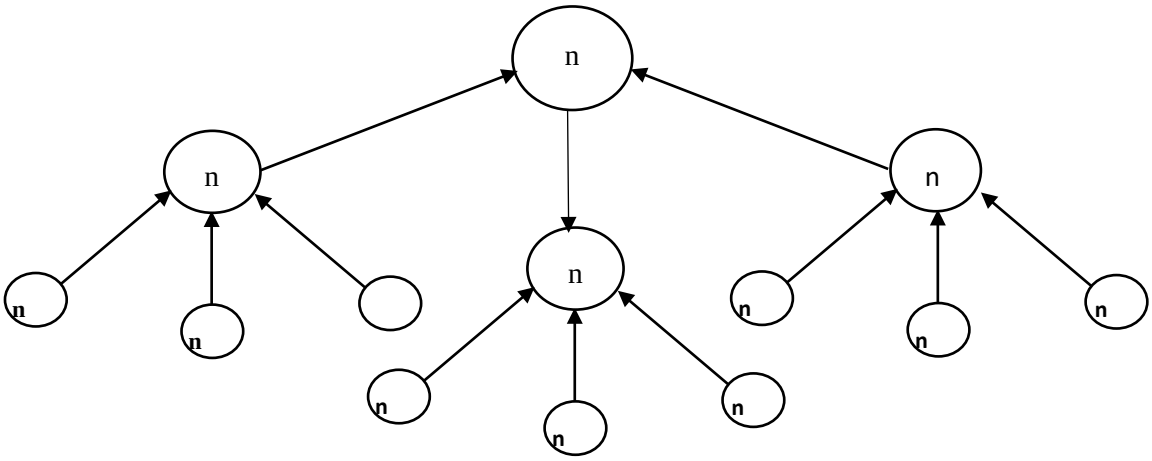


Figure 2.3: Distribute Big Data Fusion Architecture (Natale, 2015).

2.2.4 Hadoop Architecture and Design

Hadoop is an Apache open source framework written in Java; that supports the processing of bulky dataset in distributed computing environments using simple programming models (Uzunkaya *et al.,* 2015). Hadoop framework work in an environment that parallelized both computation and distribution across a federated network of computers. The fundamental principle behind Apache Hadoop is to break up unstructured data and distribute it into many parts for concurrent data analysis (Gao., 2015). It was engineered to scale up from single user server to thousands of federated machines each offering computation and storage. It consists of MapReduce, Hadoop Distributed File System, and a number of the related project such as Apache Hive, HBase and Zookeeper (Azzedin, 2013). MapReduce and HDFS are the key components of Hadoop.

2.2.4.1 Hadoop Distributed File System

It delivers scalable, fault-tolerant, rack-aware data storage and creates several replicas of the data block to be distributed across the different cluster for reliable and quick data access. HDFS runs on master-slave architecture which has a single NameNode, and more than one DataNode (Gollapudi, 2016). The name node is the point through which all client application access data, and is responsible or in control of managing the file system maintains all metadata designated for the RAM. DataNodes more of workers of the fine system (Uzunkaya *et al.,* 2015) When files are split into more different blocks, they are stored in the Datanodes. There is communication between the SecondaryNameNode and the NameNode for the purpose of evaluating the HDFS metadata within the giving time interval. This communication is defined by the cluster configuration but does not backup nameNode. The evaluated or checkpointed image is then read by NameNode

which is usually run on a separate server (Uzunkaya *et al.*, 2015). The relationship between the primary NameNode, DataNode and SecondaryNameNode is shown the Figure 2.4 below.

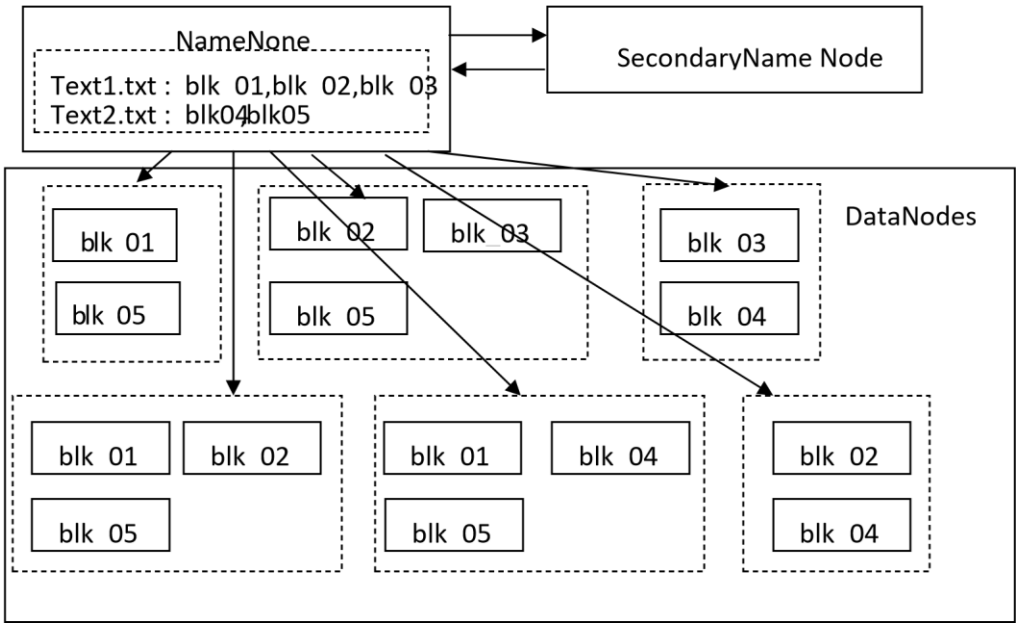


Figure 2.4: Relationship between Nodes (Uzunkaya *et al.*, 2015)

2.2.4.2 MapReduce

MapReduce is a Java-based system produced by Google where the real data from the HDFS store gets processed efficiently. MapReduce splits a big data processing job into the smaller task. The working idea of MapReduce is that its processing is done in two phase which is a map and reduces. Each phase operates on key-value pairs for input and output. The function of the mapper and reducer are program defined. Based on Hadoop architecture, it divides the input data into input split or just splits (Uzunkaya *et al.*, 2015). Each mapper is fed with a spilt by the Hadoop resulting into a process where firstly, the key and values pairs are same. Then all keys are combined together with similar value for the different mapper. The output from this process moves to reduce phase as input.at the instance of the failure of the node running the mapper then a new mapping task is started on another machine operating on the data. The reduce function does the operation of

combining the values of for a key. Its task has three primary phases: shuffling and sorted. The reduce node communicate only with other nodes is a shuffle step. Shuffling is the process of rearranging elements of data from the mappers to the reducers. The sort phase group the reducer inputs using the keys. The shuffle and sort operation occur simultaneously (Uzunkaya *et al.*, 2015). MapReduce phase has two types of nodes for controling the job execution process which is JobTracker TaskTracker. The JobTracker is responsible for handling every job on the system by the scheduling task that run on the task tracker. In this process, when a task fails, the Jobtracker will relaunch the task without human intervention. Hadoop cluster has one Jobtracker and is run on a server as a master node (Uzunkaya *et al.*, 2015). MapReduce Architecture comprising Jobtracker and task tracker is shown in the order of operation in Figure 2.5 below.

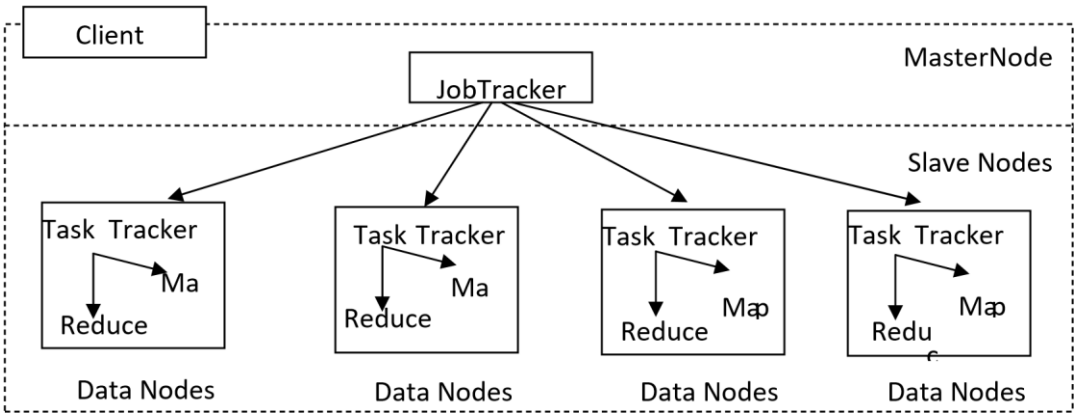


Figure 2.5: MapReduce Architecture (Uzunkaya *et al*., 2015)

2.2.5 The U.S. Joint directors of laboratories data fusion model

Many data fusion frameworks/ architectures have been produced both inside the research and business environment for the purpose of combining database on current and previous events. These frameworks have been utilized by various ventures to help the improvement of fusion frameworks by setting up the most suitable algorithm for the characterized problem of which the US Joint

Directors of Laboratory (JDL) data fusion model is commonly and widely used (Esteban *et al.,* 2005; Llinas *et al.,* 2004).

The Joint Directors of Laboratories data fusion sub-board inside the US Department of Defense initially defined this framework in the early years of data fusion. This framework was created to help the improvements in military applications (Llinas *et al.,* 2004).

The model classified the data fusion process into five processing levels, a related database, and an information bus that interfaces the five components. The five levels is however grouped into two groupings, low-level fusion and high-level fusion, which include the accompanying components (Castanedo, 2013):

i. Sources: the sources are responsible for giving the input data. Different types of sources can be utilized, for example, sensors, from the earlier data, databases, and human input.

ii. ii. Human-computer interface (HCI: HCI is an interface that enables contributions to the framework from the system and produces output to the administrators. HCI includes inquiries, charges, and data on the got results and alerts.

iii. Database management system: this is a management system that stores fed information and fused the results. The system is important because it houses the important information for processing.

In contrast, the five levels abstraction of data processing are defined as follows (Llinas *et al.,* 2004):

Level 0 - this level is called the source preprocessing level: source preprocessing is the lowest level of data fusion process, it includes fusing data at signal and pixel levels. The level handles information extraction process and reduces data by maintaining them for the high-level processes (Llinas *et al.,* 2004).

Level 1 - known as object refinement level: this stage employs the processed data from level 0. Common approaches include spatio-temporal alignment, correlation, clustering, association, state estimation, feature combination that were at first extracted from images. The output of this level is classification and identification (object discrimination) and object tracking (object orientation and state). It, however, transforms the input data into a consistent data structure (Llinas *et al.,* 2004).

Level 2 - the level of situation assessment: situation assessment identifies the possible situation provides the observed event and obtained data there establishing relationships between the observed objects. Ultimately, this level is aimed at performing high-level inference and identifying important activities and happening specifically, patterns In general (Llinas *et al.,* 2004):.

Level 3 - the level of impact assessment: this level does the evaluation of impacts that might result from the detected activities in level 2 to obtain a proper perspective. The level evaluates the present and also predict future threats;

Level 4 - the process of process refinement: this level improves the outcome of level 1-to-level 3 and provide the required resources and sensor management.

The model operates on the Hadoop MapReduce framework to fuse and process Big Data. Several considered it as the most effective method of fusion because of its ability to do object refinement (Llinas *et al.,* 2004).

2.2.6 Fog Computing

Increasingly, the number of communicating devices connected to the internet is multiplying at an exponential rate with the introduction of the internet of thing paradigm which incorporates

everything that has the ability to communicate through a form of network module (Luan *et al.,* 2015). Theses device may range from desktop computer to body wearable used medically to monitor health record of a patient, the device or popularly known as machine in research domain exhibit a communication characteristic that exists between both human and machine likewise thereby birthing what is known as machine to machine communication, in this communication scheme, machine inter-communicate with each other via network channels such as mobile device with BTS, aircraft communicating with control tower, body-wearable with medical database (Luan *et al.,* 2015). Communication is a traditional method of conversing either between two persons or device it can only exist among two or more communicators and is a continuous process, the element of communication is an electrical signal called data. Data are of two types' real time (streaming) and static (in-memory data). Due to the amount of the data generated on continuous basis, according to (Dofe *et al.,* 2016; Pettit *et al.,* 2014) Cisco announced the emergence of big data paradigm which was proposed to manage, handle and analyzed data generated by the interconnected machines irrespective of the location and size (Castiglione *et al.,* 2015). Over time, managing the data became cumbersome because of the voluminous nature of the data, it comes in variety and at very high speed thereby necessitating cloud computing (Assunção *et al.,* 2015)

Cloud computing, is a computing technique that stores, analyze and process data of large size belonging to an individual, cooperate organization, governments and businesses (Hashem *et al.,* 2015; Ji *et al.,* 2012; Kim *et al.,* 2014). It consists of a federated network of individual computers and servers (Buyya *et al.,* 2010) connected to a single location. However, cloud computing characterized by poor throughput (Zhang *et al.,* 2011), and requires a large number of network resources such as latency and Bandwidth (Kulkarni & Agrawal, 2014b; Satyanarayanan *et al.,*

2009). this process is exemplified in the Hadoop MapReduce architecture developed by Google (Rao & Reddy, 2012).

Data center are other form or a subset of cloud that is broken into smaller units to manage data generated in a specified location yet operates on the cloud architecture that is centralized by the aggregation of racks containing modular of servers that are interconnected through TO-Of-Rack (Llinas *et al, 2004.*) arranged in rows. Regardless of the above disadvantages, cloud computing offers great potentials for doing business. With service, such Software as a service (SaaS), Platform as a service (PaaS) and Infrastructures as a service (IaaS) cloud computing has reduced the cost of hosting and managing data in the business environment (Firdhous *et al.,* 2014).

The emergence of Fog Computing was announced by Cisco by introducing its first software-only version (Bonomi *et al.,* 2012; Vaquero & Rodero-Merino, 2014). This method employs Virtualizing a software running on a single local server in the case of a single user.

However, fog computing or edge mobile computing, is defined as the extension of the cloud closer to the ground or the point where data is generated (Gupta *et al.,* 2017) to manage data generated in distributed manner to reduce cost of network resources that includes bandwidth, reduce latency of communication between resource nodes connected to the internet and the cloud (Zhu *et al.,* 2013). It empowers consistent utilizing of cloud and edge resources alongside its own infrastructures by pushes applications, services, data, computing power, and decision making far from the centralized architecture to the intelligent extremes of a network (Yi *et al.,* 2015). Fog computing altogether diminishes the data volume that must be moved between end device and cloud. Fog computing empowers data analytics and information age to happen at the data source. Besides, the thick geographic appropriation of fog accomplishes better-restricted precision for

some applications as thought about to the cloud. It encourages administration and programming of process, networking and storage benefits between data centres and end gadgets/devices. Fog computing basically includes segments of an application running both in the cloud and in devices amongst endpoints and the cloud, i.e. router and switches. Fog computing underpins mobility, resource and interface heterogeneity, interchange with the cloud, and circulated data examination to addresses prerequisites of uses that need low inactivity with a wide and thick geological conveyance. Fog computing takes focal points of both edge and cloud computing while it profits by edge gadgets' nearness to the endpoints, it likewise influences the on-request adaptability of cloud resources (Okafor *et al.*, 2017).

There are various advantages related to Fog Computing that guarantees its success. The primary advantage is the reduction of network traffic as the uncontrolled increment in network traffic may prompt congestion, and results in increased latency (Kulkarni & Agrawal, 2014a). Fog computing gives a platform for sifting and investigation of the data produced by sensors by using resources of edge devices. This definitely diminishes the traffic being sent to the cloud by permitting the placement of separating administrators near the source of data. An impressive lessening in spread latency is the following vital favourable position of using Fog Computing particularly for mission basic applications that require continuous data preparing (Kulkarni & Agrawal, 2014a). Other benefits of fog computing include computation at the edge and reducing wide-area traffic. Finally, cloud computing, even with its essentially unending resources, can turn into a bottleneck if all the raw data produced by an end device (sensors) is sent to a concentrated cloud. Fog Computing is fit for sifting and preparing impressive measure of approaching data on edge device, making the data handling engineering conveyed and in this way adaptable (Gupta *et al.*, 2017; Vikram, 2015).

2.2.7 Fog Computing System Architecture

Fog computing has a distributed architecture that targets services and applications with generally scattered organizations (Osanaiye *et al.,* 2017). Literature has reviewed that many architectures have been proposed. (Osanaiye *et al.,* 2017; X. Zhu *et al.,* 2015) portrayed the Cisco outline of fog computing architecture by displaying a three-layered approach comprising of distributed intelligence end-point computing, distributed intelligence fog computing, and centralized intelligence cloud computing.

In the architecture, IoT sensors are set at the base most layer of the architecture furthermore, distributed in various land areas, detecting and sensing the environment, and sending sensed values to upper layers by means of gateways for additionally processing and sifting (filtering). Additionally, IoT Actuators work at the base most layer of the architecture and are in charge of controlling an instrument or framework. Actuators are normally intended to react to changes in conditions that are caught by sensors. IoT Data Streams are made up of an arrangement of value radiated/emitted by sensors. In the architecture, any component in the network that is fit for facilitating application modules is called Fog Device. Fog devices that interface sensors to the Internet are by and large called gateways. Fog devices additionally incorporate cloud resource that is provisioned on-request from geographically distributed data centres (Okafor *et al.,* 2017)

2.2.8 Machine-to-Machine Communication

Internet of things at first was defined by radio frequency identification (RFID) connected devices (Alexander, 2016; Sneps-Sneppe & Namiot, 2012). The IoT network today is viewed as a large expanse of device that has communication ability such as sensors, actuators, GPS devices and mobile device exchanging information with each other cross remote site in an automatic way

without human intervention. IoT devices are devices embedded in physical objects connected via wireless and wired network often using the internet protocol to connect things to the internet. "Things" here refers to sensing objects (machines) delivering sensed data into a remote database that is managed through application software (Ahmad *et al.,* 2016).

Different terms have been used in literature to describe machine to machine (M2M) communication (Arash & Vincenzo 2014). M2M, the acronym for the machine to machine application is an emerging field which allows a large variety of machine to become nodes of the personal wireless network which provides the avenue for remote monitoring and controlling of application (Krishnan, 2010). In a wider range, M2M communication involves various innovative channel like intelligent remote control of different parts of machines to remote reading of sensing devices. Thus, M2M is not limited to creating passive data collection point but an intelligent inter machine coordination of community of sensors.

Arguably, an estimated number of the device will be connected to the internet by 2025 of which 50 billion mobile devices are estimated to be connected and a total of 500billion devices including mobile device has been referred to as guesswork by Ericson (OECD, 2012). This inter-connected device forms an eco-system that is defined by a hierarchy of broad layers of M2M enabled devices which are realized through the embedded chipset. Hence constituting a building block along with embedded wireless communication modules to shape up various customer device and application as presented in figure 2.6. The final enabler of this community or cluster of the device is the various M2M software application that collects and enable intelligent decision around M2M value chain which in turn forms the bases for operating M2M architecture.

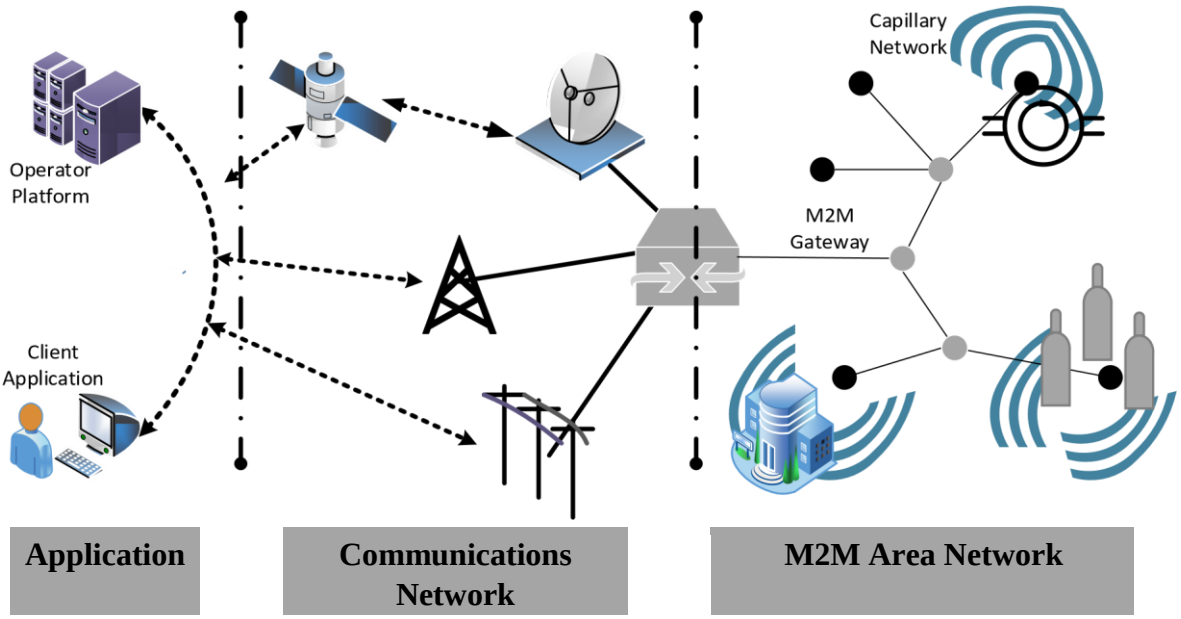


Figure 2.6: A simple M2M Architecture (Sneps-Sneppe & Namiot, 2012)

2.2.9 Ensemble Learning System

Ensemble system generally called multiple classifiers in decades have attracted the interest of researchers in the domain of machine learning and artificial intelligent (Polikar, 2012; Polikar *et al.*, 2012). This interest is because of its performance in classification and fusion when contrasted with single classifiers. Ensemble system contains various learners which are ordinarily called base learners. The generalization ability of an ensemble is typically much stronger than that of base learners. Actually, ensemble learning is appealing because it is able to boost *weak learners* which are slightly better than a random guess to *strong learners* which can make very accurate predictions. So, “base learners” are also referred as “weak learners. Ensemble learner can be said to been designed according to human's second nature which is to review different process preceding decision making in the various circle of life, for example, medicine, social problem, security. The capacity is essentially the blend of the weak learners to enhance the certainty of

decision making by weighing different alternatives and after that combining these opinions the process known as voting. A rich collection of algorithms have been developed using ensemble method with a global aim of improving the generalization performance of classification system (Ditzler *et al.,* 2017; Polikar, 2017). Ensemble algorithm work by running a base learning algorithm several times and forming voting out the resulting hypotheses.

The general idea behind ensemble system learning are as follows (Roli & Giacinto, 2002):

i. Train several learners called base learners L1, L2 ……. Lm to produce differently.
ii. Given a test input x, combine each of these hypotheses h1(x), h2(x)... hm(x) in some way.
iii. If the individual hypotheses are different from each other in some way, then probabilities exist that the overall result is more reliable than that given by any individual hypothesis.

2.2.9.1 Combining Classifiers.

Classifier combination is an approach in building ensemble learning in which either some common classifiers are combined in a linear format such as combining five SVM to form a learning or combining different classifiers to form an ensemble learning technique. This section considers rules for combining classifier.

2.2.9.2 Combining class labels

The combining rules are used if the class labels are output from classifiers C. the decision of t^{th} classifier is $d_{t,c} \in \{1,0\}$, t = 1, ……..,T and C =1, ……..,C. if the choice of a t^{th} classifiers is a class w_c, then the decision $d_{t,c}$ = 1 and otherwise 0. Where output 1 correspond positive and 0 is negative output. Equation (2.1) represents the decision rule of combining labels (Polikar, 2012).

$$d_{i,j} = \begin{cases} 1 & if\ D_1\ \ label\ in\ x\ w_j \\ 0 & otherwise \end{cases} \quad (2.1)$$

2.2.9.3 Voting based Fusion Techniques

This technique operates on labels in which classifiers k are to choose C or not, the ensemble then decides to choose class C that represents the largest total vote. The techniques include methods discussed.

2.2.9.4 Majority Voting

Majority Voting makes decision-based on the class with the largest number of voting among the ensemble classifiers. There is three version of majority voting. Unanimous voting, simple majority and plurality voting. Majority voting is mostly referred to as plurality voting. Equation (2.2) is the mathematical expression majority voting. Other form of voting techniques exist in literatures such as Weigh ted majority voting (Asmita & Shukla, 2014).

$$\sum_{j=1}^{T} W_i d_{i,a} = \max_c \sum_{j=1}^{T} w_i d_{i,a} \quad (2.2)$$

Where $d_{t,}$ is decision of classifier that decides on a class w_i.

1. Weighted Majority Voting

Some classifier are likely to be more accurate than others, weighing the choices of those classifiers can improve their performance if compared to plurality majority voting. The idea behind this method is the classifiers apportion a weight $_i$ to each classifier in proportion to its estimated performance. Decision are combined in WMV resulting into the choice of $_i$ (Asmita & Shukla, 2014).

$$\sum_{j=1}^{T} W_i d_{i,a} = \max_c \sum_{j=1}^{T} w_i d_{i,a} \quad \textbf{(2.3)}$$

The choice of each classifier is ensemble according to weighted majority voting then choose a class if the weighted voted received from all classifier by a class is higher than the total vote received by other class in a data.

2.2.10 Algorithms

This section discusses common algorithms and architecture used in implementing Ensemble Learning in literature.

2.2.10.1 Boosting

Boosting (also called Adaptive resampling and Combing) is a generally accepted method for enhancing the performance of any machine learning algorithm. The method works by running a weak learner more than once, (for example, order classification or decision trees), on different distributed training data i.e. increasingly adding one hypothesis per time to an ensemble (Dietterich, 2002). The classifiers delivered by the weak learners are then joined to form a single composite solid classifier with a specific end goal to accomplish a higher precision/accuracy than the weak learner's classifiers would have achieved (Asmita & Shukla, 2014).

It first appeared in literature in 1990 by Schapire and Freund and Schapire in 1995 introduced what is known as AdaBoost Algorithm (Dietterich, 2002). The essence of developing this algorithm is to assign weight to the example/instances in the training set. Equal weights are assigned in the beginning, however, in each round of training, the weights of all misclassified instances/variables are improved/increased while the weights of accurately classified instances are decreased. As a result, the weak learner is compelled to concentrate on the difficult instances of the training set.

Similar to Boosting, Adaboost has three tuning parameters which are listed below and the training process is shown in pseudocode 1.

1) The number of tree can over fit if C*h* is too large

2) The shrinkage parameter λ, is a small positive number that regulate the rate boosting learn the values is usually 0.01 and 0.001 the choice is problem dependent.

3) The number of spilt in each tree controls the complexity of boosting. The split can also the referred as interactive depth. Figure 2.7 and psudecode 1 represents the order and method in which Averaging and sequential training is done in boosting.

Figure 2.7 Showing the sequence and order of averaging Boosting result.

Set *Ch* = 0 and *Si* = *Zi* for all *i* in the training set

Set h = 1, 2, 3 ……. H, repeat (a) and (b)

Fit a tree C*h* with the training data using weight (X, *Si*)

Update C*h* by adding a shrinked version to the new classifier

Ch = C*h*(x) + λC*h*(x)

Update the residuals

Si = *Si* – λC*h*(x)

The final output of Boosting model is given as:

$Ch = \sum_{h=i}^{H} \lambda Ch_{(x)}$

pseudocode1: Boosting Ensemble learning.

The implementation process of boosting is explained in Algorithm 1

Where:

Ch = number of classifiers that constitutes a tree.

Si and *Zi* represent the residual and the response respectively

h= classifiers λ = shrinkage parameter

The algorithm accepts that the training set comprises of *m* instances, marked as - 1 or +1. Another case of classification is made by voting on all the classifiers $\{c_t\}$, each of having a weight of w_t. Numerically, it can be composed as (Rokach, 2005):

2.2.10.3 K- Means

K-Mean algorithm is an algorithm that group or cluster unlabeled dataset. It has two iterations the first is the cluster assignment step and the second is the move centroid step. The first step assigns data into a group with similarities or nearness of features and the second on the other hand classify the dataset based on statistical parameters such as mean and average.

To perform K-means clustering, first, the number of clusters must be defined then, the algorithm groups the dataset into the specified numbers of clusters. K-means can be defined to be a descriptive analysis method that uncover unknown structures in data into k- cluster. Equation (2.4) is the mathematical representation of k-mean for minimizing defined functions.

$$H = \sum_{j=1}^{k} \sum_{j=1}^{m} \left(f_i - f_j \right) \tag{2.4}$$

Having a dataset of m, applied equation (2.4) will partition the data into k number of clusters having a distance function d the mean feature and ith features that are jth related is represented as f_i *and* f_j respectively.

In equation (2.4), several method can be used for determining the parameters for clustering features. The method range from Euclidean distance which is the most used, manahattan distance and mahalanobin distance. Equation (2.5) is the equation for Euclidean distance between features(Barrett, 2006).

$$d = \sqrt{\sum_{i=1}^{v} \left(p_{1i} - p_{2i} \right)^2} \tag{2.5}$$

Where $\left(p_{1i} - p_{2i} \right)$ is the difference between point 1 and point 2, v Represents the number of observation where $i = 1$ to p.

2.2.10.4 Haar-like feature

The Haar-like feature is a machine learning approach developed by Viola and Jones in 2001 for visual object detection which has the capacity of processing image very fast and achieving high detection rate (Mustafa *et al.,* 2014). The principle is such that all human faces possess a common feature that is similar in nature. And these regularities are used for matching. Below are some of the features that are similar to all human face (Viola & Jones, 2001).

1. The eyes region is darker than the nose and upper cheek
2. The nose bridge is brighter than the eyes.

Trainable features used in matching faces:

1. The location and size of human eyes, mouth and nose bridge
2. Value – the intensity of pixels forming a location.

The algorithm relies on the above mention four features to detect and match the human face. Training the algorithm requires that each region is represented in the form of a black and white rectangle which is dark and bright respectively (Viola & Jones, 2001). The algorithm takes into account the sum difference between pixel value taken from the dark region and compared to the sum value of the pixel of bright region of a rectangle. The algorithm is an ensemble learning that

requires a combination of both weak and strong classifiers in other to produce a better result (Viola & Jones, 2001). These process of forming ensemble learning is shown using equation (4) above.

The algorithm is of four sequences (Viola & Jones, 2001) namely:

1. Haar feature selection
2. Creating an integral image

In order to reduce the processing time of Haar –like feature integral image can be adopted as the intermediate representation. The integral image at a position (x,y), is the sum of the pixel value on the top and left side of (x,y). This is expressed mathematically in equation (2.6)

$$_{u'}(x,y) = \sum_{j=1}^{(x,y)} \sum_{j=1}^{(x,y)} u'(x,y) \quad (2.6)$$

3. Adaboost training
4. Cascading of classifiers.

2.2.11 OpenCV

OpenCV is a cross-platform Java, C++ and python library that can be used to develop real-time computer vision application. It finds its application mainly in image processing, video capture and analysis including face detection (Kaehler & Bradski, 2016).

In this work, OpenCV will be used to implement object detection and matching on the Haar cascade algorithm.

2.2.12 Weka

Weka a machine learning environment developed for purpose of data mining (Frank *et al.*, 2009). Developed by the University of Waikato in New Zealand with a uniform interface to learn many algorithms with pre-and post-processing methods and evaluating result.

Weka as a tool for data analytics is built with the different working environments such as Workbench, Explorer, knowledge flow simple CLI and Experimenter. The most convenient means of using weak is through its graphical interface GUI. Notably, the algorithm in weka takes their input in a single ARRF file format (Frank *et al.,* 2009; Romero *et al.,* 2008).

Weka is a java written data analytic platform which supports several data mining algorithms for the execution of tasks such as data preprocessing, classification, clustering regression visualization and feature selection (Romero *et al.,* 2008).

2.2.13 Classification

Classification is an unsupervised learning method of extracting information from dataset.it is also referred to as categorization in machine learning. It involves the grouping of features or observation into categories of similarities or common features. Classification entails identification and detection of objects features based on trained rules which in most cases are mathematical function and these classifiers in similitude are called Algorithm (Kotsiantis *et al.,* 2007).

Several classification algorithms exist in literature examples includes Bagging, Boost, support vector machine SVM, Naïve Bayes and others (Voyant *et al.,* 2017). Each algorithm is application specific and differs in performance because their training input vector may differ in training (Kotsiantis *et al.,* 2007).

Image classification requires that a computer is able to analyze and interprets an image by identifying its pixels or features which must be numerical. Computer's ability to identify pixels is due to its ability to count pixels (Kotsiantis *et al.,* 2007).

Pixel-based classification is a data mining approach in which pixel values are extracted from an image specifically operational land imagery OLI through the use of Raster Library in R and ArcGIS and the extracted pixel values form the classification input vector on which the information can be categorized (Wang, 2012).

2.2.13.1 NDVI and NDWI

Normalized difference vegetation index and Normalized difference Water index are standard indexes which can generate an image relative biomass (Jackson *et al.*, 2004).

NDVI relies on the difference between the Red band and the high green reflectance in the nearinfrared (NIR) band of an image (Ji *et al.*, 2009). NDVI can be calculated as below (Konda *et al.*, 2018):

NDVI = (NIR-RED) / (NIR+RED). (2.7)

Here, band4 = RED and Band5 = NIR.

NDWI, notably, is an approach used for mapping water bodies in satellite images. It possesses strong usability and radiation in the range visible to infrared wavelength. It uses NIR and Green band to identify water body in OLI. Its wavelength value is larger (0.5) than that of vegetation (0.2) this help in differentiating water from vegetation in an image. NDWI can be calculated as below:

NDWI = (GREEN- NIR) / (GREEN+NIR) (2.8)

2.2.13.2 *Dataset*

The dataset to be used for this research work is a one terabyte size of data and are landsat8 and sentinel2A satellite data obtained from the Nigeria Geological Survey Agency, Abuja, Nigeria. And the U.S geological survey online database.

Landsat 8 image digital information is as shown in table 2.1

Table 2.1: Landsat8 Digital Information (USGS, 2018)

	Bands	**Wavelength**	**Resolution**
LandSat8 Operation and imagery (OLI) And thermal infrared Sensor(TIRS)	Band1 coster aero ol	0.45-0.51 0.45-0.51	30
	Band2 –Blue		30
	Band3 - Green	0.53-0.59	30
	Band4 - Red	0.64-0.67	30
	Band5 Near Infrared(NIR)	0.85-0.88	30
	Band6 – SWIRL1	1.57-1.65	30
	Band7 - SWIRL2	2.11-2.29	30
	Band8 Parachromatic	0.50-0.68	30
	Band9 – Cirrus	1.36-1.38	15
	Band10 – thermal infrared (TIRS)	10.60-11.19	100*30
	Band11- thermal infrared (TIRS)	11.50-12.51	100*30

2.2.13.3 System specification

Due to the nature of the dataset to be considered for this research work and the processing capacity that is required, Hadoop will be implemented in the cloud and a single installation on a corei5 Toshiba satellite L50 -B laptop for offline data processing. The process will run on virtualized Ubuntu Linux operating and Microsoft window 10.

2.3 Review of similar works

Anagnostopoulos & Skourlas, (2014) perform emotional classification with an ensemble majority voting classifier that combined three different classifies of low computation and complexity. The model was used to perform emotion classification form speech utterance by considering six acoustic parameters. Z-transform was performed on the parameters and feed into the classifiers and combined to form an ensemble system a tenfold cross-validation technique was used to measure the emotion accuracy. Their model outperforms two other models when compared in terms of class accuracy and prediction accuracy. However, the ensemble learning majority voting classifier was used under a small sample size of data. Thus, the performance of the classifiers is affected.

Yang *et al.*, *(*2015) developed an automatic classification of remote sensing image using multiple classifiers system. The aim of the work was to improve the correctness of identifying land used type sensing image; a satellite image of SPOT-5. Using different machine learning classifier such as support vector machine, back propagation neural network, fuzzy c-means and others to form an ensemble voting system of three multiple layer classifier system(MCSs). Two MCSs were implemented and WA-AHP was proposed to introduce analytics hierarch process into MCSs. To improve the accuracy of the classifiers, the data was preprocessed to remove noise contained in

the data. The experimental result shows that the weighted average that is based on AHP (WAAHP) is an efficient MCS while MCS, on the other hand, is superior to base classifiers in remote sensing image classification. However, the system was trained on a single satellite image for grouping and identifying features contained in the image which undermine the potentials of an ensemble learning method in decision making for effective and accurate data fusion.

Ahmad *et al.*, (2016) Proposed a system architecture designed for analyzing big data in M2M using JDL; a data fusion model with five abstraction that is applicable to both real-time and offline dataset. The proposed architecture first after data is collected, pre-process the data using unstated machine learning approach. Then, the dataset is split into a block of 1000 x1000 then, each block is fed into Hadoop MapReduce Queue in which data with similar features are aligned for the purpose of fusion. The architecture was tested on a dataset from European space agency images, health-related sensor data, and body temperature dataset. The result of the implementation on Hadoop ecosystem shows that the system architecture detected water and roads. However, the system architecture operates on MapReduce which represents a centralized data fusion architecture. Thus, it is an architecture that requires much network resources such as latency and bandwidth.

Okafor *et al.*, (2017) proposed a spine-leaf fog computing network topology (SL-FCN). The proposed model was developed to work side by side with the cloud computing data centre network. The model proposed an extended cloud model that optimized bandwidth thereby facilitating fast processing of Data at the edge. SL – FCN was compared against a legacy multi-layered architecture housing networking service and network on low latency and bandwidth in data centres consequently SL – FCN was found to outperform in latency reduction and network congestion in

highly distributed and multilayer virtualized data centre environment. However, the model introduces issues on the selection of cabling technology platform for link paths. Thus, making it difficult for deployment to a different network.

Oh & Kang, (2017) presented object detection and classification by decision level fusion for intelligent vehicle system. A data fusion technique for a multilayer LIDAR and a CCD sensor, classification output from independent unary classifiers using Convolutional Neural Network (CNN) of five layers, architecture which use more than two pre-trained convolutional layers to consider to local to global feature as data representation. To represent data in CNN, a region of interest pooling was applied to the output of each layer on the object candidate region. The proposed method was tested on the KITTI benchmark dataset to detect and classify three object classes such as cars, pedestrians and a cyclist. This method was reported to achieve high performance, identify about 500 objects and obtained a 77.72% classification performance with mean average precision. However, despite its high performance when compared to other techniques, it's not computationally cost-effective and the processing time was not reported.

Dautov & Distefano, (2017) developed distributed data fusion for the internet of things data fusion – fusing data close to it a point of generation help in minimizing the amount of noisy data being sent over throughput of limited wireless links. Which is a potential way of implementing sensor data fusion in distributed IOT system, aiming to leverage local processing capacities wherever possible or offload task to edge/cloud computing thereby necessary a multi-layer hierarchical data fusion which on the other hand will result into reduction in network latencies and the amount of transferable data? CEP is a fusion technique that has the capacity of fusing incoming streams that are implemented in IOT scenarios. The proposed approach was necessarily developed as an

improvement to other existing CEP technique which considered data fusion at the cloud level and neglecting the possibility of data fusion at the network level and only focuses on only data collection. However, a hierarchical multilevel architecture for data fusion in IOT was proposed in the data is first processed on board then pushed to the cloud. This is a three-tier conceptual level architecture which is local area data fusion (LADF), wide area date fusion (WADF) and Global area data fusion (GADF). Distributed data fusion was implemented using Drools fusion as an underlying CEP middleware, Raspberry Pi, represent the data collecting sensor and fused data collected locally and Amazon web service cloud was used in place of GACEP (GADF). The experiment was tested on university campus data. However, the research work was based on streaming data as such it never considers object and signals location and tracking.

Habib ur Rehman *et al.*, (2017) proposed Red Edge: A Novel Architecture for Big Data processing in mobile edge computing environment which essentially aim at the processing of data reduction on the edge. Big data dimensional reduction has been a machine learning solved problem only in the cloud despite the many challenges faced by cloud in handling such size of data to mitigate this, the proposed red edge model employs the mobile edge device as a data reduction platform where the data Mining algorithm process and uncover knowledge pattern and store the resultant data using local storage in mobile edge device and synchronizing with cloud data store. To achieve their objectives, a three-layer architecture was proposed that utilized computational power form the MECC system. The first layer handle data reduction at the edge device, the second performance data reduction by forming ad hoc network by enabling collaborative data processing among device while the third layer uses cloud resources to perform data reduction. The performance evaluation was based on memory consumption, latency and reduced bandwidth

utilization when compared with others architecture, Red Edge was found to outperform them with 92. 86% big data streaming reduction without compromising energy and memory consumption on mobile edge; reduced data to about 7.14% of overall data. However, the proposed techniques suffer battery consumption and memory overhead. Thus, making it inappropriate for utilization in distributed big data fusion setting as edge fog device will be located remotely.

Elkhatib *et al.*, (2017) investigated the ability to run different customer server in Docker container as against classical benchmark suits by the use of raspberry pi (RPI) to represent a fog deployment. The choice of raspberry pi over hypervisor was an experimental setup to explain that hypervisor is not suitable for low – power processor without hardware virtualization support also, the choice of Docker container represents a realistic way of achieving isolation in micro – cloud. The experiment performed on latency shows that fog has the potential to deliver service to the end user putting into consideration the programming model and hardware. Thus, the experiment was performed on small data size thus, resulting in an improved result. Consequently, no research of this regard has defined the actual data size required for the implementation of distributed fog data fusion architecture.

Din *et al.*, (2017) proposed a Data fusion technique for big data in wireless multi-sensor system using a hybridized algorithm for clustering and clustering member selection in a wireless sensor network system. The device was grouped based on geographical region and a device is assigned the head which is responsible for both inter and inter-cluster communicate to effectively achieve this objective, cluster architecture based network was adopted to minimize transmission power, for effective load balancing, reduce collision by reducing extensive hop try-hop communication. The data from member sensor were fused using the JDL DF model which was implemented in java

iteration and Hadoop architecture and tested on healthcare datasets. The proposed technique which is a centralized Data fusion technique competed favourably with other methods but has the challenges associated with centralized data fusion which includes latency and bandwidth.

Alturki *et al.*, (2017) present a hybrid approach for data analytic for IoT. The proposed approach is aimed at moving some of the processing off the cloud to investigate the cost of carrying out data analytics in cloud and off- cloud. In the proposed approach, data were fused and filtered closer to source of generation and thence sent data in which information have been extracted from to the cloud rather than sending the data in its raw form to the cloud for analyses. The approach was achieved through the use of two different architectures cloud and fog both represented in WEKA and raspberry pi 3 experimental set. Data training was implemented in the cloud while fog was responsible for fusion and aggregating sensor and transmitting dataset to a dedicated channel to the cloud. The data was analyzed in Weka using five classification Algorithm and the result shows improvement in accuracy which transcend to low latency consumption and low bandwidth. The approach was reported to perform better if the size of dataset used is reduced.

From the reviewed literature, several architectures have been successfully implemented in the area of data fusion for both centralized and distributed data fusion respectively. With an emphasis on Hadoop MapReduce as the most used for centralized architecture for big data and raspberry pi, Docker and ensemble learning techniques for distributed architecture, data size, latency, communication bandwidth remains a common limitation to the implementation of a distributed architecture for big data fusion. The need for implementing an architecture that is capable of fusing multi-source data in a relatively short period of time consume less network bandwidth at low latency geospatially remains an open research area.

Addressing the underlying research shortcomings such as large bandwidth and latency in a centralized architecture, thus, this work developed an architecture capable of processing data generated from a different location in a distributed manner to reduce the computational cost effect of fusing data centrally.

CHAPTER THREE

MATERIALS AND METHODOLOGY

3.1 Introduction

This chapter presents the various methods, procedures and materials adopted by this research work to implement a centralized big data fusion architecture and the distributed big data fusion architecture. The adopted materials and steps of the methodology are discussed below.

3.2 Materials

This section discus the materials used in this research work. The materials includes hardware and software and discussed as follows:

3.2.1 **Software**

The software adopted for the implementation of this research work includes

i. Hadoop

Hadoop single Node cluster was installed on Ubuntu Virtual machine VM for the implementation of MapReduce centralized architecture. The steps for installing Hadoop single node are shown in Appendix A.

ii. RLanguage

RStudio version 3.4.3 was installed for processing spatial images using Raster package.

iii. Weka Data Mining Environment

WEKA v3.8 was installed on windows 10 using the explorer for the implementation of distributed data fusion architecture using ensemble learning. Figure 3.1 shows the snapshot of WEKA GUI chooser including the all the modules and Explorer module.

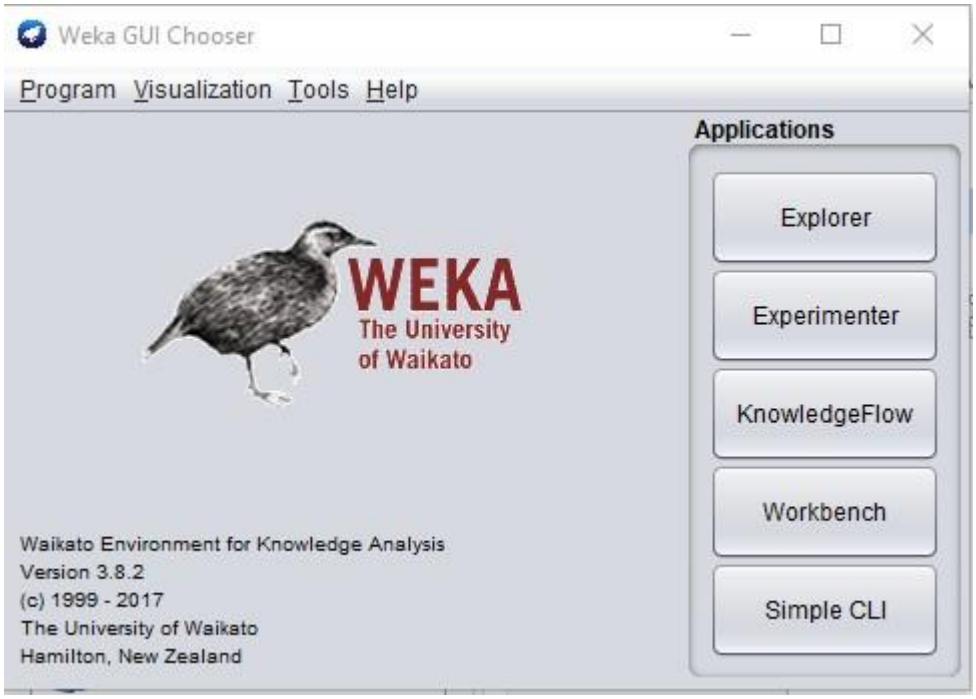


Figure 3.1: Snapshot of WEKA interface

3.2.2 Hardware

This research work carried out using virtualized HP 3125 AMD 2.5 GHz 4GB memory capacity and TOSHIBA Satellite L50-B for implementation Hadoop of the single cluster that will synchronize with the cloud platform.

3.3 Methodology

The methodology adopted in this research is presented in figure 3.1a.

1.) Implementation of centralized big data fusion architecture

i. Data acquisition from USGS database.

ii. Data pre-processing using Rstudio.

iii. Implementation of JDL Data fusion model

iv. Development of object detection and identification model

v. Object detection and identification using Adaboost (Haar- like method)

(a) Create an Xml file and train with Adaboost classifier
(b) Test and Cross validate

2.) Implementation of Distributed Big Data Fusion Architecture using Ensemble learning.

Single phase

i. Repeat (I and II) in 1.

ii. Feature engineering.

(a) Extract (RGB, ncell, nlayers and nrow)

(b) Extract coordinate and layers into x,y table

(c) Extract Region of interest (ROI) by cropping.

(d) Calculate ndvi and ndwi

Multi-phase

i. Convert data file format to ARFF file

ii. Ensemble the learners (classifiers)

a) Initialize a base learner in Weka

b) Increase the numbers of learners from one to ten.

c) For each step in (b) Perform data analytics - Save result.

d) Save result

3. To compare the two architecture in term of latency**.**

The methods adopted to achieve this research set objectives are presented in Figure 3.1.

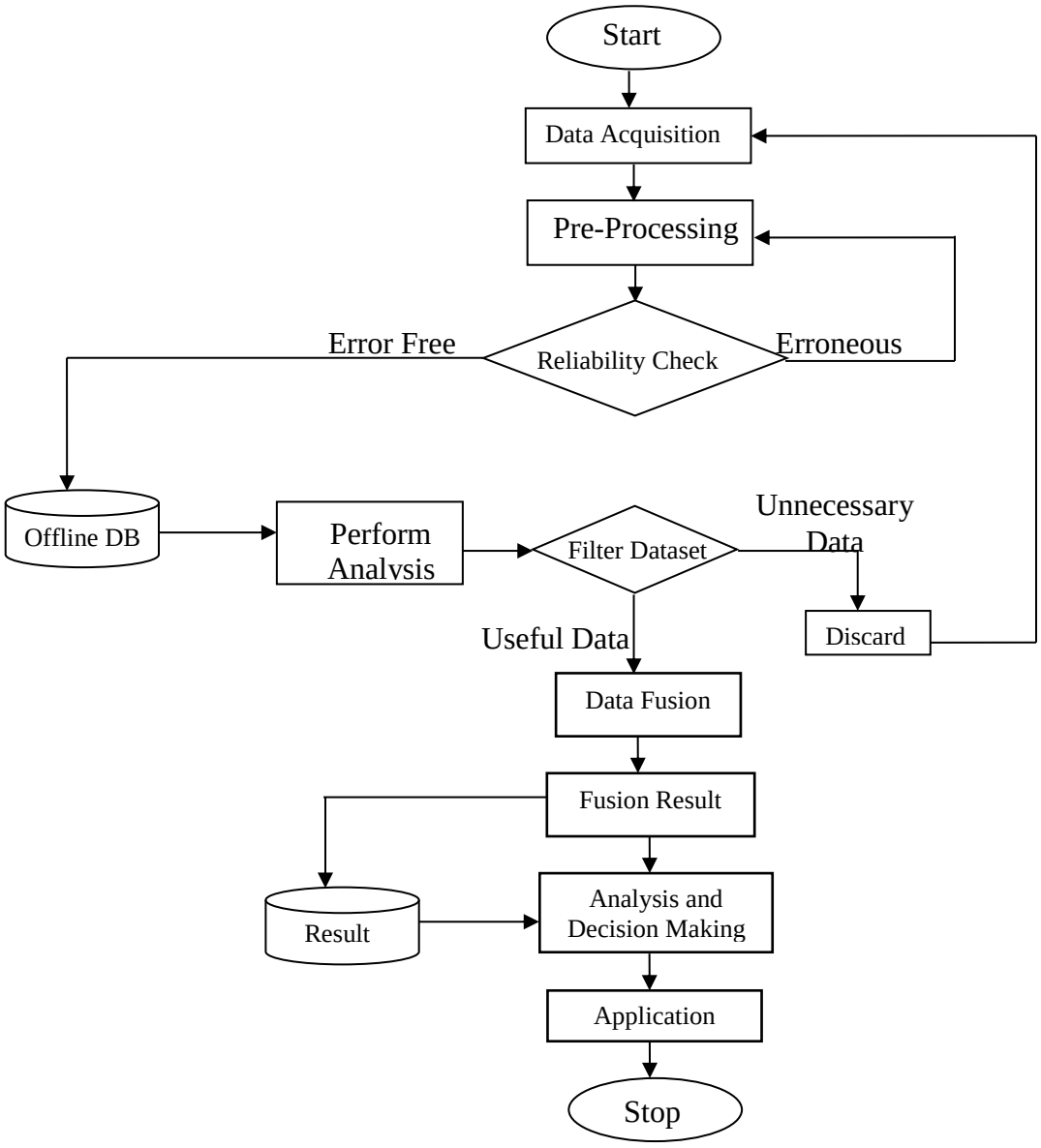


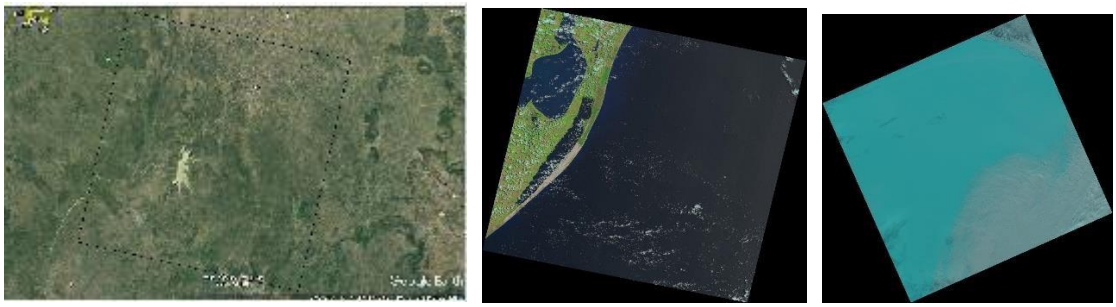
Figure 3.1a: Flow Chat of Offline Data Fusion Architecture

3.3.1 Development of Centralized Big Data Fusion Architecture

Centralized data fusion architecture is a data fusion architecture that fuse or aggregates data from different origin at a central location. It is implemented in this research work following the steps discussed herewith.

3.3.1.1 Data Acquisition

In this research work, the input images are Landsat satellite images obtainable from European Space Agency, Addition satellite imagery was sourced from the Nigerian Geological Survey, Abuja and the U.S. Geological Survey website through this link https://deltas.usgs.gov/fm/data/data_ndwi.aspx. Data collected are Landsat 7 and 8, and Sentinel 2A data covering different location in Nigeria from October 2017- February 2018. Figure 3.2 shows samples acquired data.



(a) Landsat8 Image Covering Settlement (b) Landsat8image (c) Cloudy Lansat8 Image

Figure 3.2: Landsat8 satellite images (USGS, 2018**)**

3.3.1.2 Pre-Processing

Filtration algorithm was applied on the data set. The filtering method adopted for de-noising dataset is the Gaussian median. The choice is due to its ability to preserve the edge of an image and removal of salt and pepper type of noise. Equation (3.1) is for mean distribution while equation

(3.2) is for Gaussians median filter. Given z= gray level, σ is mean, ith and jth is the center of Nij, median filter can be calculated using equation 3.2.

$$P(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{(z-\mu)^2}{2\sigma^2}} \quad (3.1)$$

$$y_{ij} = (\{x_{uv} \mid x_{uv} \in N_{IJ}\}) \quad (3.2)$$

3.3.1.3 Data reduction

Data reduction is also a stage in pre-processing which seek to reduce the size of high dimensional data to a manageable size that best represent the entire dataset. In this research work, PCA was used to carry out dimensional reduction. The process involves centralizing the data in a matrix X, where the tuples and row are features, then, $X^T X = n^V$, where V is the covariance of the data. Given p-dimension of feature where the vectors is given as $\vec{w}$ and the given projected size of data vector $\overrightarrow{x_i}$ the mathematical model for implementing PCA is given in equation (3.3) (Novembre & Stephens, 2008).

$$\frac{1}{n}\sum_{i=1}^{n}\left(\overrightarrow{x_i} - \overrightarrow{w}\right)\vec{w} = \left(\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right).\vec{w}\right)\vec{w} \quad (3.3)$$

3.3.2 Feature Engineering

In this research work, feature engineering includes extraction of features (water body) from satellite image and the process adopted is shown in Figure 3.3

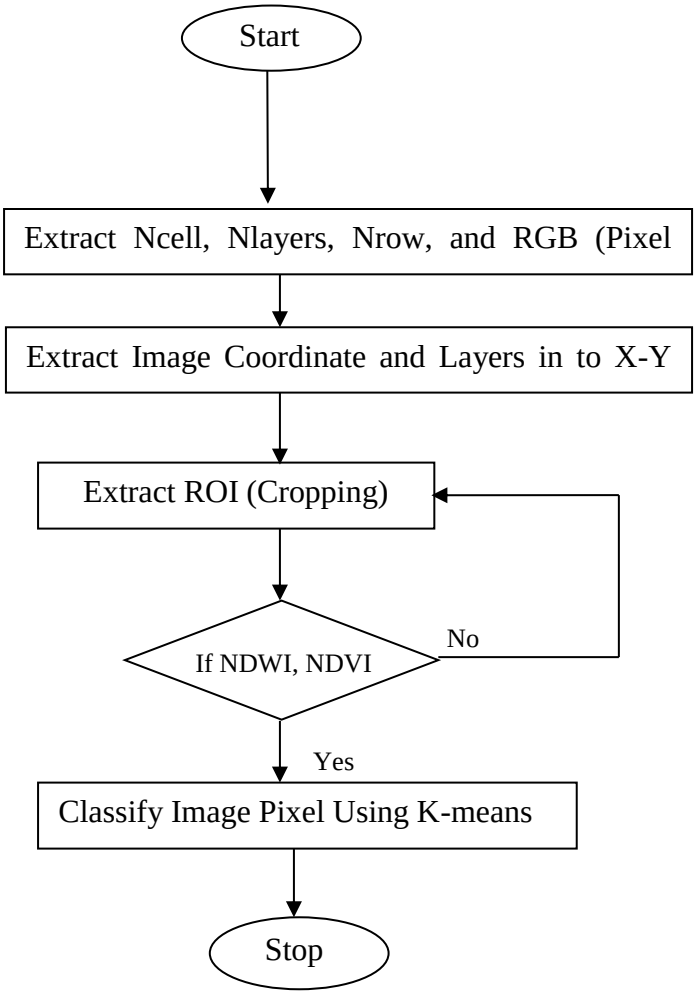


Figure 3.3: Feature Engineering Flow Chat

Figure 3.3 above represent the procedure followed to extract features from a satellite image.

3.3.2.1 Extract Pixel values.

Pixel value extraction using raster library to obtain the pixel values in ASCII code. The code for pixel extraction in RSTUDIO is shown as below:

```
>nlayers(R)

>ncell(R)

>R<- raster(nrow=20,ncol=200)
```

>S<- stack(sapply(1:5,function(i)setvalues(r,rnorm(ncell(r),I,3)))

>Q<- extract(s,1:ncell(s)

Head(Q)

3.3.2.3 Extract image coordinate and layers in X, Y table. .

This section extract image coordinates and store it in a table format. The following program listing shows snippets of the portion of that extracts the image coordinates into a table format. >Head(rasterTopoint(s))

3.3.2.4 Crop and extract pixel values of Region of Interest (RoI)

Spatial subset or cropping can be used to limit analysis of an image to geographical subset of the image it can be created using extent or crop function in R. for a given dataset, $\phi(g)$ defines the feature vector $g \in l$ where g is element of l feature on $\left[g_1, y_1\right], \left[g_2, y_2\right] \ldots\ldots, \left[g_l, y_l\right]$. ϕ Represent a function for feature extraction which shows how correlated the pixels. Hence, $\phi\left(l\left(\vec{g}\right)\right)$ is the feature vector of g . Cropping in this instance is same as searching for feature of interest vector $\phi\left(l\left(\vec{g}\right)\right)$ if the difference existing between two feature vectors $\phi\left(l(g)\right)$ and $\phi\left(l\left(\vec{g}\right)\right)$ is small then it would be close to an expected feature. The difference between two features can be established by calculating the Euclidean distance in equation ()

$$f_{(x)} = \| \phi\left(l(g)\right) - \phi\left(l\left(\vec{g}\right)\right) \|_2^2 \tag{3.4}$$

3.3.2.5 Extract NDVI and NDWI

To investigate the presence of both vegetation index and water bodies in the image, NDVI and NDWI was calculated for the original image and the cropped section using the mathematical expression shown in equation (3.5) and (3.6) respectively

NDVI= (NIR - RED) / (NIR+RED) (3.5)

NDWI = (GREEN- NIR) / (GREEN+NIR) (3.6)

Where: NIR= band4

RED=band1

3.3.3 Implementation of U.S. Joint Director of Laboratory

This section implements Data Fusion using the U.S JDL model. JDL data fusion is a five abstraction model developed by U.S Military to track an event history through fusion of related data. The abstractions are discussed in section 2.1.6. This work adopted object Alignment, object refinement and object detection and identification abstraction of the model.

3.3.3.1 Object Alignment

This process involves the conversion of images of different orientation to a unified extent. Bitmap image file (.BMP) extension was used for training and the image was resized to 960X637. The conversion was carried out using Microsoft office paint.

3.3.3.2 Object Refinement

This stage involves procedures such as converting data into consistent structure and calculation of statistical parameters. These procedures are discussed in detail below:

Converting the dataset into the consistent structure: the dataset used for training our model was converted from the .TIF format to .BMP format as required by the algorithm. Statistical parameter was calculated by first determining the standard deviation between the maximum and minimum pixel value distribution. The process also include determining the correlation between the variables and ascertaining the mean value of the dataset. Equation (3.7) perform statistical parameter such as mean value and standard deviation.

$$SD = \sqrt{\frac{\sum |x-\mu|^2}{N}} \tag{3.7}$$

Where: X= value of data set, μ= mean of the data set and N= number of data point in the population

3.3.3.3 Object identification and Detection using Haar-Like Feature (Adaboost) Algorithm.

Object detection and identification is a machine learning process of instructing the machine to detect and identify objects or features on the ground of the training instructions. To achieve object detection and identification, the dataset was divided into two separate data, one for training and the other for testing. The following steps were adopted

1. *Train:* model training is the mathematical and computation process of instructing the models on what action to perform using the selected features from the training dataset. The line of code to perform this operation can be found in Appendix A.

   The training dataset contains features with which the algorithm is trained with by specifying a region of interest (ROI) so that the algorithm can identify such region of interest when running on the testing dataset. In Haar-like feature, water region represents the dark region while land and other features represent the light region.

2. *Test:* testing evaluate the performance of the algorithm that was trained on dataset for feature identification, classification and detection. These datasets contains about 30% of the entire dataset.

3.4 Development of Distributed Big Data Fusion Architecture using Ensemble Learning.

This stage involves the implementation of ensemble learning using Adaboost in WEKA. The operation is first performed on a single based learner then on ensemble learners so as to establish a bases for comparison. Figure 3.4 is the flow chat for implementing a distributed data fusion architecture using Adaboost.

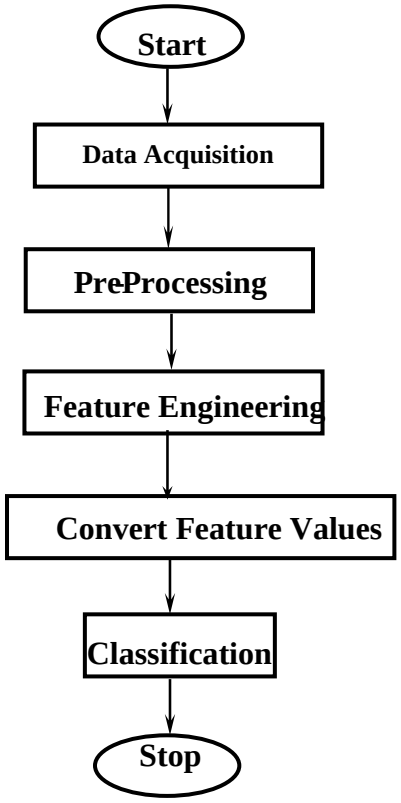


Figure 3.4: Flow Chart for Developing a Distributed Big Data Fusion Architecture.

3.4.1 Data Description:

The Dataset used in the course of this research work is a one terabytes size of satellite earth observatory images specifically for Nigeria. The features of interest are water body, vegetation, land and rocks. The images are non-geographical referenced this owns to the fact that the majority of the dataset were outsourced from different source. The images were observed to have seven bands and of varying extent and number of cells.

3.4.2 Data Pre-Processing:

WEKA pre-process tab is a function for carrying data preprocessing. Input data for WEKA must be in .csv and .arff file format. For the purpose of this research, earth observatory data were converted into .arff file format and the dataset was rid of missing values, then normalized and PCA was implemented. Also, statistical parameters were calculated using the pre-process tab of WEKA. Figure 3.5 is the snapshot of WEKA preprocessing GUI interface.

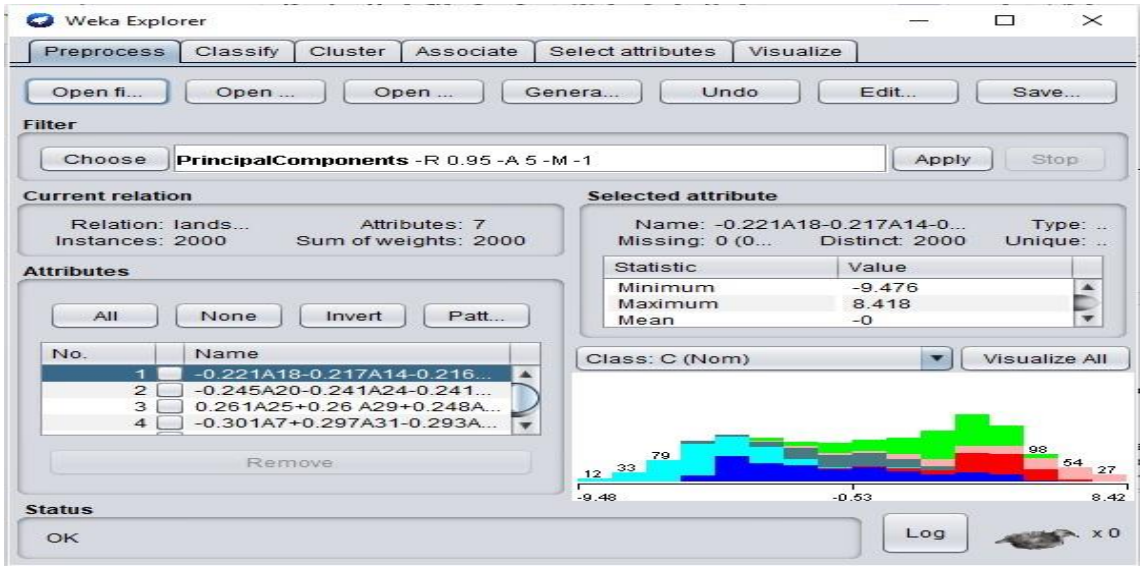


Figure 3.5: WEKA Interface for Data Pre-Processing

3.5 Classification

This stage employed the use of Adaboost Ensemble learning algorithms to implement objective two of this research work. The algorithm were first applied as a based learner to the dataset and then the number of the learners are gradually increased. The combination of their operation is what results in ensemble learning. Below is the implementation.

3.5.1 WEKA Implementation of Ensemble Learning (Adaboost)

This section introduce the process of ensemble Adaboost to implement distributed big data fusion architecture. The process involves a step by step approach through WEKA interface to build a model that give the most desired output. The process is highlighted below and figure 3.6 is a snapshot of WEKA interface for the implementation.

The implementation of ensemble Adaboost using WEKA is shown as listed below:

Stage one.

1. Click classify tab
2. Click Choose and select **classifiers, Mata and AdaBoostM1.**
3. Click on **AdaBoostM1** then change the setting and configuration
4. Click Choose to change the default algorithm from DecisionStump to **Random Forest** by selecting the method **classifiers**, **rules and the JRip.**
5. **S**et the enumerations to 1 for the single learner.

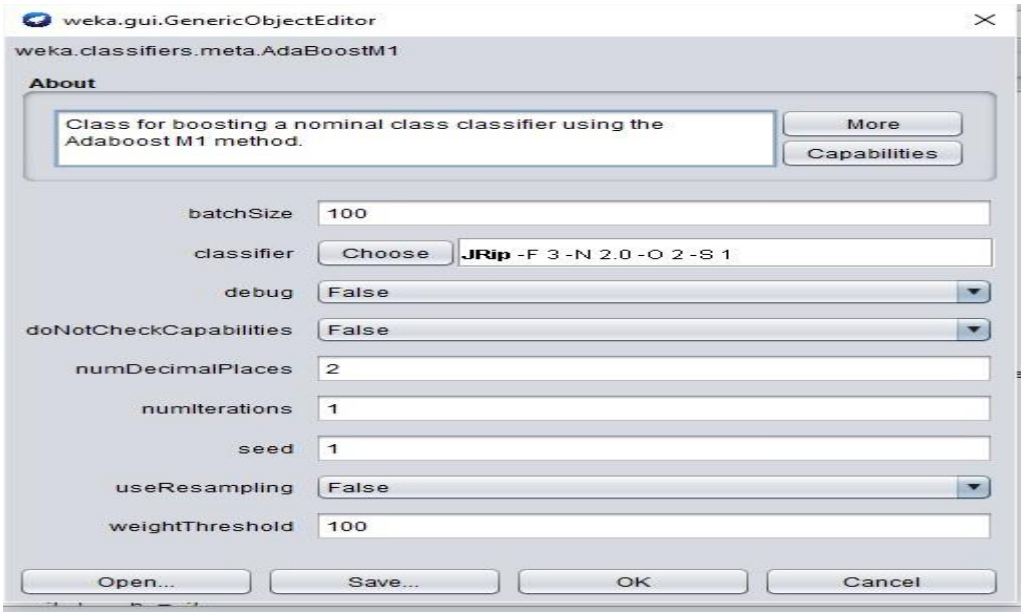


Figure 3.6: Snapshot of the implementation of Boosting Single and Ensemble Learning

Ensemble learner implementation using WEKA Stage two:

This stage implements the ensemble method of learning by changing a number of enumerations to 3, 5, 8 and 10 respectively.

From the implementation above, the following parameters were considered using WEKA.

The number of seed was set to 1, the weight Threshold which is the threshold for weight pruning was set to 100. The result was output in 2 decimal value s and the batch size is 100%. numIteration is the number of trees to be built in iteration.

3.6 Performance Evaluation

This section evaluates the performance of the proposed Distributed Big Data Fusion Architecture Using Ensemble Learning on two factors namely Latency and Bandwidth consumption. The two parameters in the context of this research are expressed execution time and Accuracy for each

classification result obtained. Herewith, a confusion matrix was employed as a base for drawing a comparison between the two architecture.

3.6.1 Bandwidth

Several literatures have express bandwidth as the aggregate of data that is transferred from the point of sending to the point of receiving. Prediction bandwidth reduction is achieved at the individual base learner in machine learning if the time taken for a learner to data from one to another point of fusion/ aggregation is lesser.

3.6.2 Latency

This is an interval of time between request and response. It is an expression of distance and time. In machine learning, prediction latency is computed as the measure of the elapsed time required to make a prediction. Latency is a factor that determines the choice of a machine learning classification algorithm. Latency in machine learning can be achieved through an approach called doing- in bulk. A prediction approach for predicting many instances in bulk.

3.6.3 Confusion Matrix

A classification table that shows the output of a processing in machine learning classification. The table contains information such as Accuracy, Misclassification, RMSE, MAE, True Positive (TP), False Negative (FN), and True Negative (TN), False Positive (FP), Recall, Precision, Sensitivity and Specificity.

Accuracy is the proportion of accurate to total to total prediction made

$$\text{Accuracy} = TP + TN / TP + TN + FN + FP. \quad (3.6)$$

$$\text{Precision} = TP / TP + FP. \quad (3.7)$$

Sensitivity = TP/ TP +FN. (3.8)

Specificity= TN/ FN+FP (3.9)

Where:

TP is the number images correctly classified. i.e image contain training features and was classified positive,

FN = image is positive but classified negative,

TN= image is negative and classified negative

FP= image is negative but classified as positive.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Introduction

This chapter presents the results of centralized big data fusion architecture and distributed big data fusion architecture which was implemented using different ensemble methods employing the use of Adaboost machine learning algorithms then establishing comparison between the two architecture. Implementing the Architectures involves data pre-processing, data visualization and implementation of Adaboost for data fusion

4.2 Centralized Data Fusion Architecture

Centralized architecture was developed in phases as presented in figure 3.1a. the phases are data acquisition, data pre-processing and reliability check, feature extraction and analysis. Phase 2 and was implemented using Rstudio while OpenCV on the Hadoop platform was employed in feature identification as applied in Haarcascade. The implementation involves the following steps Data pre-processing for preparing the dataset for training, data visualization. This stage is followed after the dataset have been prepared for training the essence is to have a visual view of the content or information (features and observation) contained in the dataset. Also, NDVI, NDWI and classification using K-means was implemented to investigate the region of interest. The result from these process inform our decision on variable to build our model.

4.2.1 Data **Pre-processing**

The result of data cleaning and filtration is present in Figure 4.1.

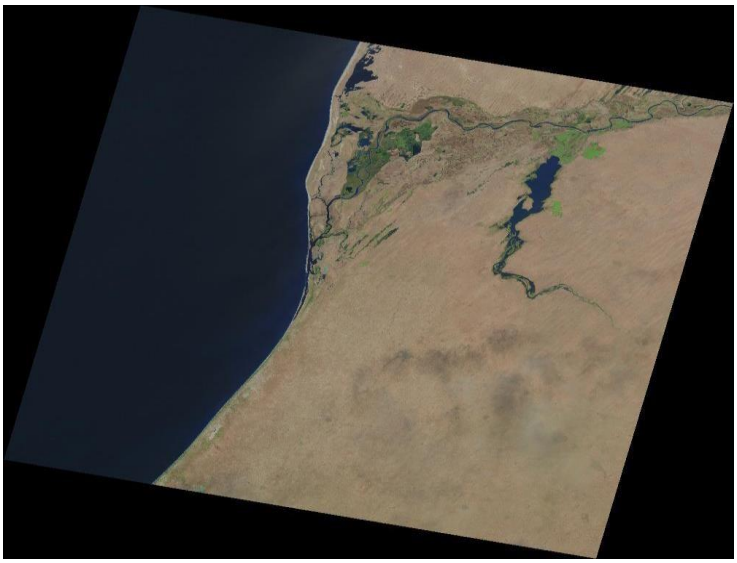


Figure 4.1: De-noised Satellite Image using Median Gaussian Filter Algorithm.

Figure 4.1. is the implementation of median Gaussian filter to de-noising image. The filter enhanced the image quality by removing inherent noise in the image. The results show a smooth and bright image.

4.2.1.1 *Data Visualization*

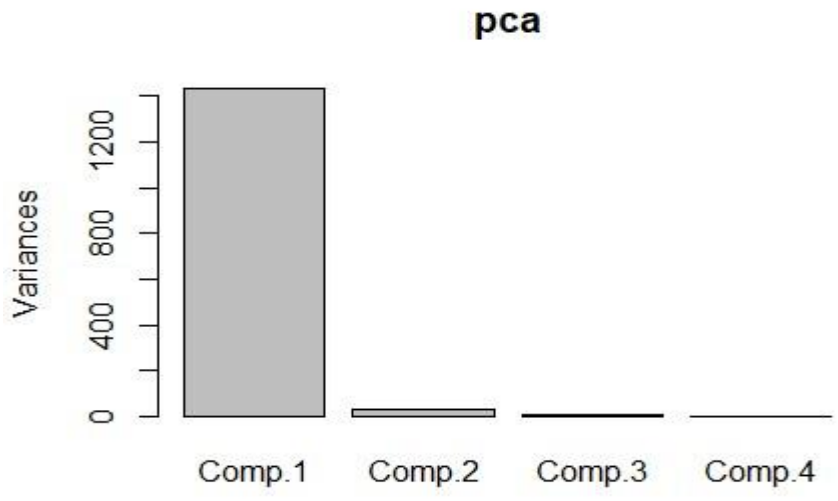


Figure 4.2: Plot of Principal Component Analysis

Figure 4.2 represents the plot of Principal Component Analysis (PCA) for landsat8 satellite image in Figure 3.2(a). The plot of the number of variances against number of component Analysis where Comp1 is the first principal component and has the highest variable distribution while comp2 to comp4 are second, third and fourth principal components. Each component represent variance of predictors contained in the dataset but decrease in value of information content from one to four.

4.2.1.3 Satellite image bands extraction

Satellite image band extraction is a method in spatial image processing used to visualize the colour component of an image. The method employed in R use the plotRGB() function to display the Red, Green and Blue band of an image. Figure 4.3 is the extracted bands of original image in Figure 3.2(a) containing RED, GREEN, BLUE and INFRARED band of a .TIFF satellite image.

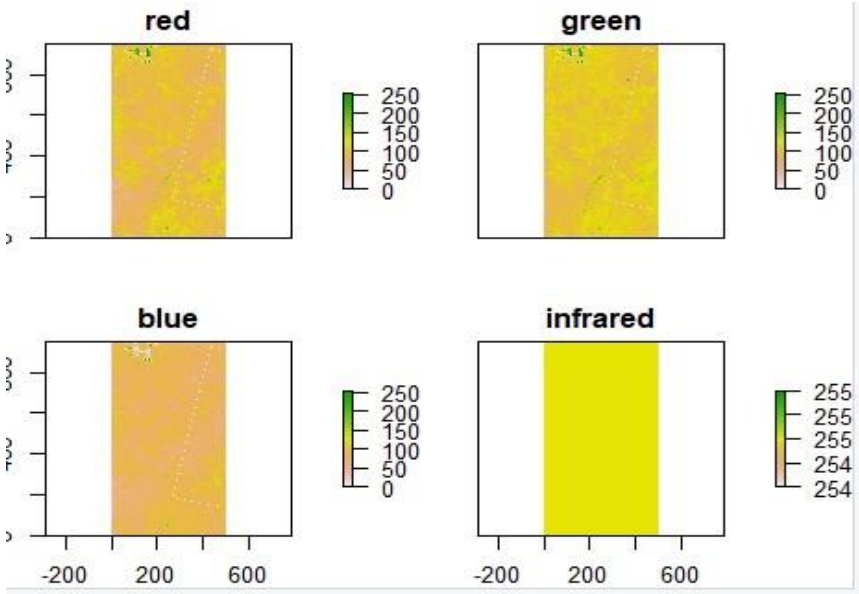
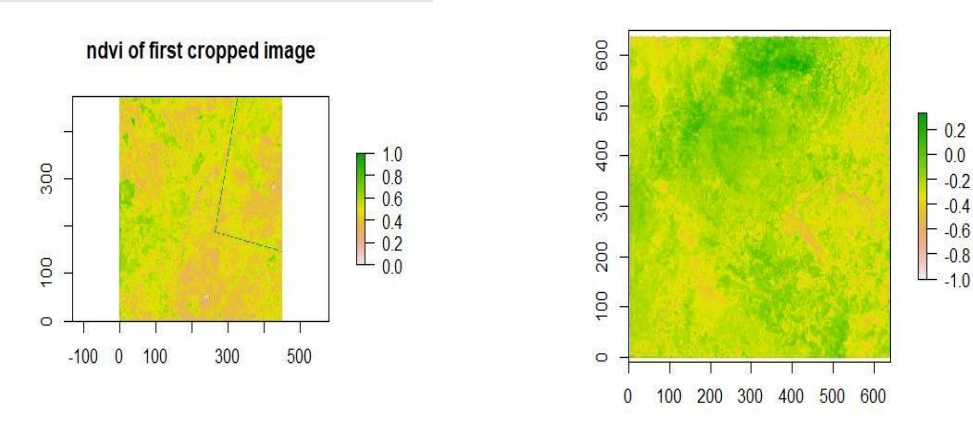


Figure 4.3: Plot of Image Visualization in Rstudio

Figure 4.3 shows the plot of image Band Visualization of figure 4.1 using Rstudio. The plot shows the image Bands (RGB & infrared), extent and pixel value distribution for each Band.

4.2.1.4 Implementation of NDVI and NDWI

Normalized difference of vegetation index and Normalized difference of water index are means of calculating and identifying the presence of vegetation and water features in an earth observatory image in spatial image analysis. The technique can be applied as a means of visualizing the component of trainable features for the purpose of model training. Figure 4.4 is the result of NDVI and NDWI implementation in Rstudio**.**



(a) The plot of NDVI (b) plot of NWDI

Figure 4.4: Plot of NDVI and NDWI feature

Figure 4.4 shows the implementation of NDVI and NDWI using Rstudio. The plot shows the distribution of water body and vegetation index which ranges from 0.5 - 0.59 and 0.6- 0.7 of the pixel value for vegetation and water respectively. Both X-axis and Y-axis represent the extent value representation of the image.

4.2.2 Classification/ clustering

Clustering is a machine learning technique that seek to classify or group features or information in a dataset by similarities. It is an unsupervised approach that is usually applied on a set of data to

visualized information distribution. Figure 4.5 is an implementation of classification technique applied an extracted region of interest of Figure 3.2(a) using K-means. The purpose is to visualize trainable feature contained in the region of interest by clustering them into groups.

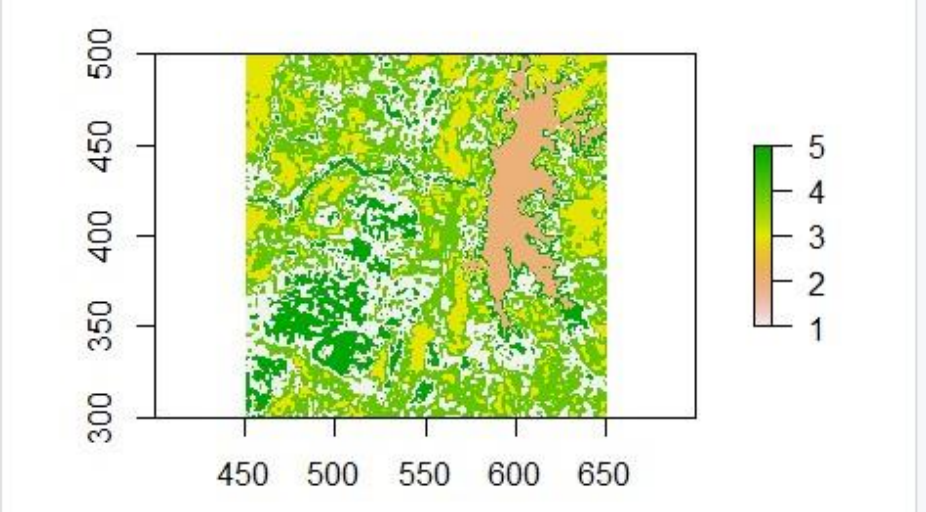


Figure 4.5: Feature Clustering using K-means

Figure 4.5 is the implementation of k-means. The Figure is the visual representation of features such water body which exist in colour band of 1 to 2.7, land has colour band of 2.7 to 3.2 and vegetation has colour3.3 to 5 as shown on the legend displayed on the right-hand side of the plot. The algorithm clustered each feature based on the similarity of their pixel values. Knowledge of the features is useful to the implementation of feature engineering.

4.3 Analysis of Centralized Data Fusion Implementation

This section presents the result of trained Adaboost (HAAR-Like feature) model. The result is in the form object-based feature detection and the analysis of the result thereof.

4.3.1 Adaboost (Haar)

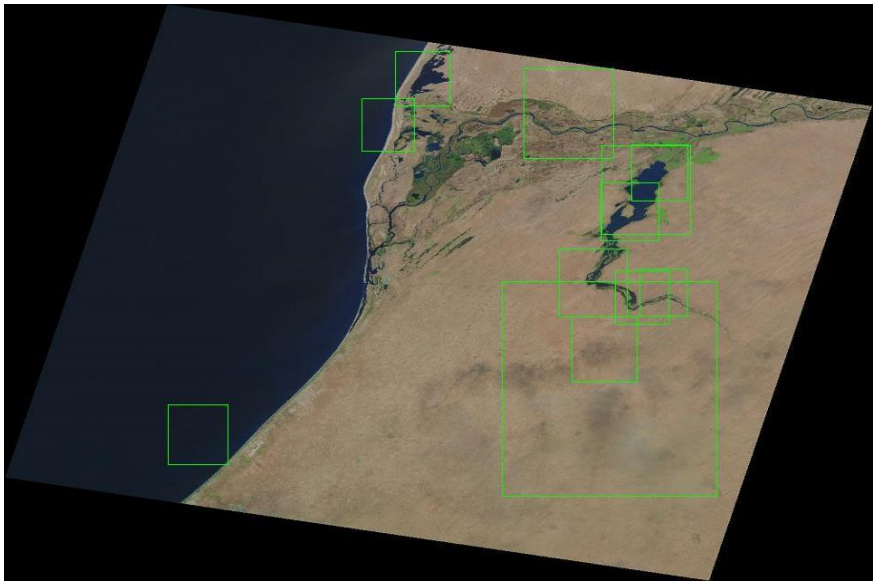


Figure 4.6: Adaboost Correctly Identified Water Body.

Figure 4.6 is an image showing the outcome of Adaboost Algorithm (Haar method) trained on training satellite image dataset to detect and identify water body in images. The algorithm was successful in detecting and identifying this feature in images but failed to differentiate the difference between shadows, dark portions and the training features in images. Figure 4.6 shows the water body correctly identified image in green triangles and Figure 4.7 shows incorrectly identified. Figure 4.8 showing MapReduce running Feature detection on Hadoop.

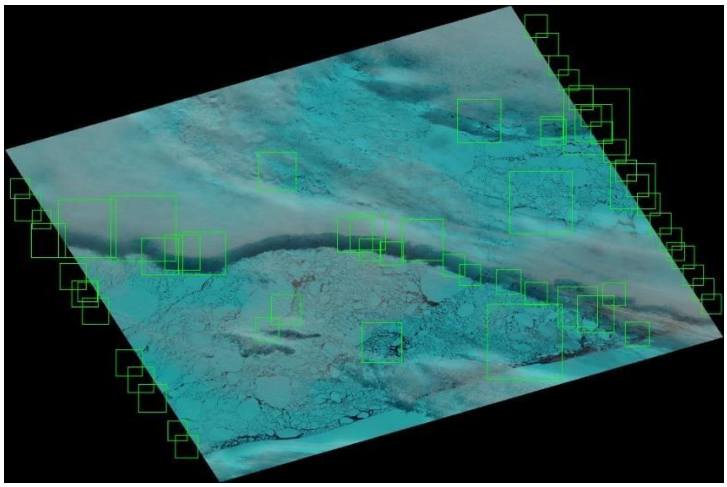


Figure 4.7: Features Incorrectly Identified.

Figure 4.7 is the pictorial representation of an incorrectly identified feature but shows the degree of sensitivity of the algorithm in interpreting the training rules which is based on Haar – like feature principle defining the difference between dark and light pixel. Herein, the algorithm interpreted the line as water because of the variance in the pixel values.

```
abiodun@abiodun-SATELLITE-L50-B: ~/opencv/sample
000033.bmp
Writing 000033.bmp
File: /home/abiodun/Pictures/Satelite/000021.bmp

Running DetectFaceDemo
/home/abiodun/opencv/lbpcascade_frontalface.xml
Detected 0 faces
000021.bmp
Writing 000021.bmp
File: /home/abiodun/Pictures/Satelite/000026.bmp

Running DetectFaceDemo
/home/abiodun/opencv/lbpcascade_frontalface.xml
Detected 0 faces
000026.bmp
Writing 000026.bmp
File: /home/abiodun/Pictures/Satelite/000031.bmp

Running DetectFaceDemo
/home/abiodun/opencv/lbpcascade_frontalface.xml
Detected 3 faces
000031.bmp
Writing 000031.bmp
```

Figure 4.8: Hadoop Implementation of Feature Detection and Identification

Figure 4.8 is the pictorial representation of object detection of the algorithm implemented in openCv library on Hadoop MapRedue centralized architecture. Haar interpret object as face; in the result shown in Figure 4.8, each face is a correspondence of square drawn on the ROI. The result shows that the algorithm detected the trained feature as a face which is the method by which Haar is operated on.

4.3.2 Confusion matrix

Confusion matrix is a table that shows statistical classification result of a machine learning. The table consist of True positive (TP), True Negative (TN), False Positive (FP), False Negative (FN).

Table 4.1: confusion matric table for centralized Architecture

Data size (%)	**TP**	**FP**	**TN**	**FN**
100	149	4	35	39
80	120	8	38	46
60	52	17	35	31
40	60	17	35	13
20	60	38	18	13

Table 4.1a shows the table of the confusion matrix. The table displays the experimental results of True Positive (TP), False Negative (FN), True Negative (TN), and False Negative (FN)

Table 4.1b: accuracy, sensitivity and specificity table.

Data size (%)	Accuracy	Sensitivity	Specificity
100	86.2	89.7	82.2
80	82.55	82.6	72.2
60	65.1	75.3	63.7
40	63.81	77.9	82.2
20	63.76	77.9	82.2

From table 4.1b, the algorithm attained peak performance for sensitivity when the dataset was at 100% having 89.7 and 82.2 for specificity when the set is partition and 40 and 20% was used for training the algorithm. High performance at 20 and 40% training dataset is an indication that the algorithm was able to detect the presence of negative image introduced into the test dataset. Bandwidth consumption and latency were observed to be very high using this algorithm because it requires a longer time to perform an operation. Though the Accuracy, sensitivity and specificity are high but requires longer time.

4.4 Results from Distributed Big Data Fusion Architecture Using Ensemble Learning. This section presents the results obtained for the process of developing a distributed architecture using an ensemble method of learning. These procedures starts with data acquisition,

Pre-processing, data visualization, attribute selection, feature engineering, analysis and results.

4.4.1 Data pre-processing

In this section, the dataset used is assumed to have been pre-processed in section 4.2 except for PCA that was applied for dimensional reduction in WEKA. Pixel values were extracted from the images. To implement PCA in WEKA, first initialize WEKA then apply PCA on dataset.

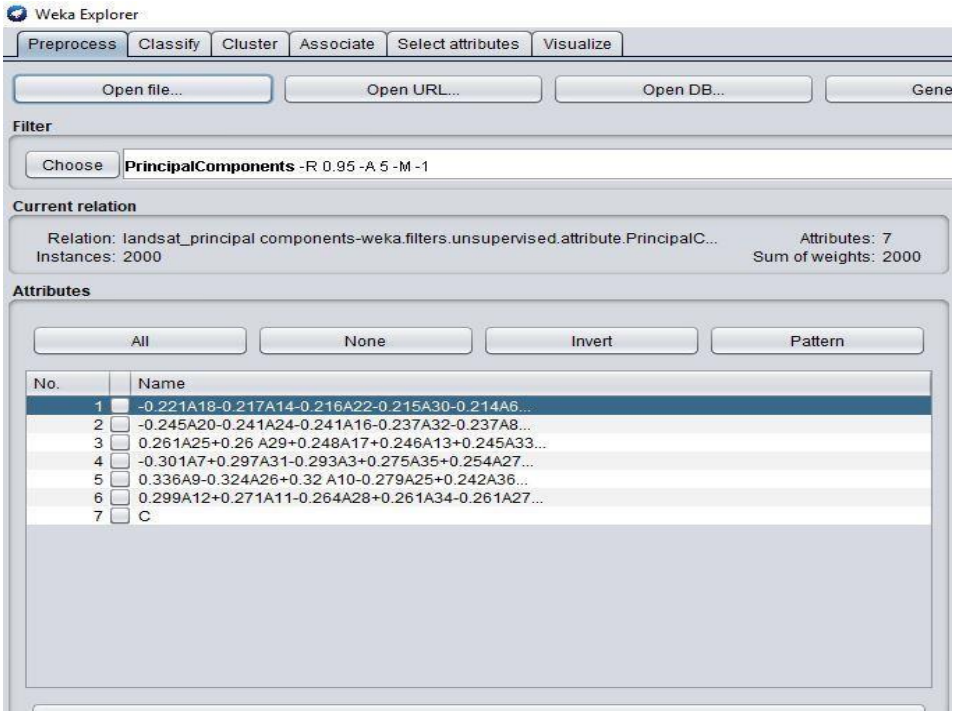


Figure 4.9: WEKA Implementation of Principal Component Analysis

Figure 4.10 shows the implementation of PCA using WEKA. The result shows that the variable was reduced from 10,000 to 2000 variables. This owes to the presence of missing values and incomplete information contained in the dataset. Hence, 2000 variable represents the first component which represent a good distribution of information.The interface shows a total of seven attributes which is comprised of six image Bands and the seventh is the class.

4.4.1.1 Statistical parameter

Statistical parameters here denotes the mathematical representation of variable distribution with respect to the minimum, maximum, mean and standard deviation of the dataset. Figure 4.11 is the snapshot of statistical parameter of training set after performing PCA.

Statistic	Value
Minimum	-9.476
Maximum	8.418
Mean	-0
StdDev	4.002

Figure 4.10: Statistical parameters

Figure 4.10 is the statistical representation of the data after performing dimensional reduction. The result reads: the minimum variable contained in the dataset is -9.476 which in most case, the negative values represents clouds as a feature. Maximum variable in the dataset is 8.418. From table 2, 8.4, the value is an infrared band and the mean value settles at -0 while the standard distribution of the variable is at 4.002, which also represent the centre of distribution so as to prevent biases in the training data.

4.4.1.2. Data Visualization using WEKA.

Based on information in table 2, features are identified through pixels values that represents each feature within the dataset. Figure 4.12 is the snapshot of data visualization in WEKA.

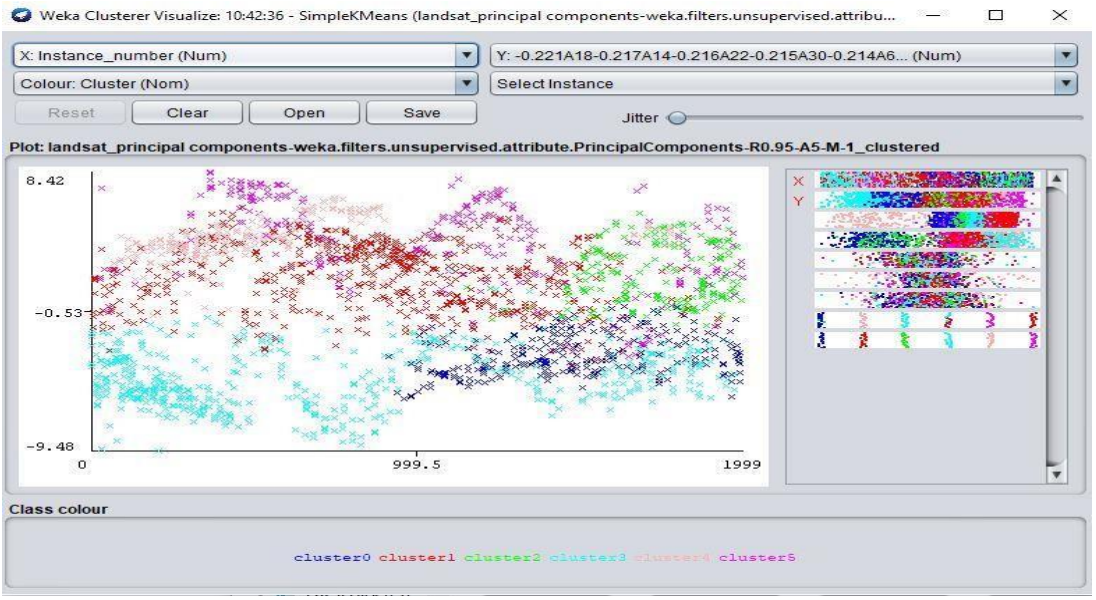


Figure 4.11: Data Visualization

Figure 4.11 represents pixels distribution in the dataset. The process shows the variables distributed. To achieve this, simple means algorithm which uses Euclidian Distance as its distance function and kmeans++ for the initialization method and has a maximum iteration number of 500. The result shows seven attributes labelled as cluster 0 to 6. Each cluster is represented in different colours. The XY axis shows the number of instances and number of variable distribution from the minimum to the maximum.

4.4.2 Implementation of ensemble learning Adaboost

Ensemble learning is an approach in machine learning that combines either same or different algorithm to perform a task then agree on the best result to output. Each classifier is referred to as a base learner. Learner can either be weak or strong learner. This research work employed the use of 10 base learners. Appendix C is the confusion matrix of the implementation including number of learners from one to ten. And figure 4.12 is the plot of number of evaluation matrix against accuracy for all learner

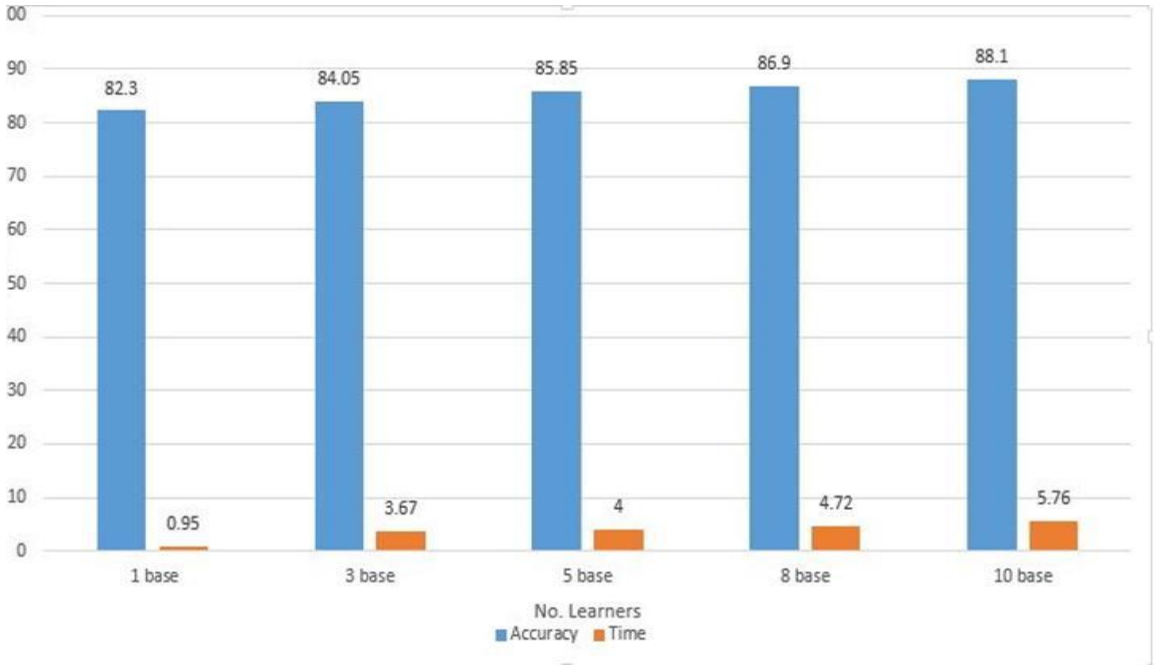


Figure 4.12: Boosting; Time of Execution

Figure 4.13 represents the plot of ensemble implementation of Boosting method using WEKA. The result shows the execution time difference between a base learner and ensemble 10 classifiers to be 0.95 min (950 ms) and 5.76 min (300,760 ms). From the result obtained, it is established that the more the classifiers, the longer it takes to achieve a better performance as it involves the sum of individual classifier (base learner) execution time and time it takes to aggregate the result to be output.

4.4.3 Comparing centralized and distributed architecture

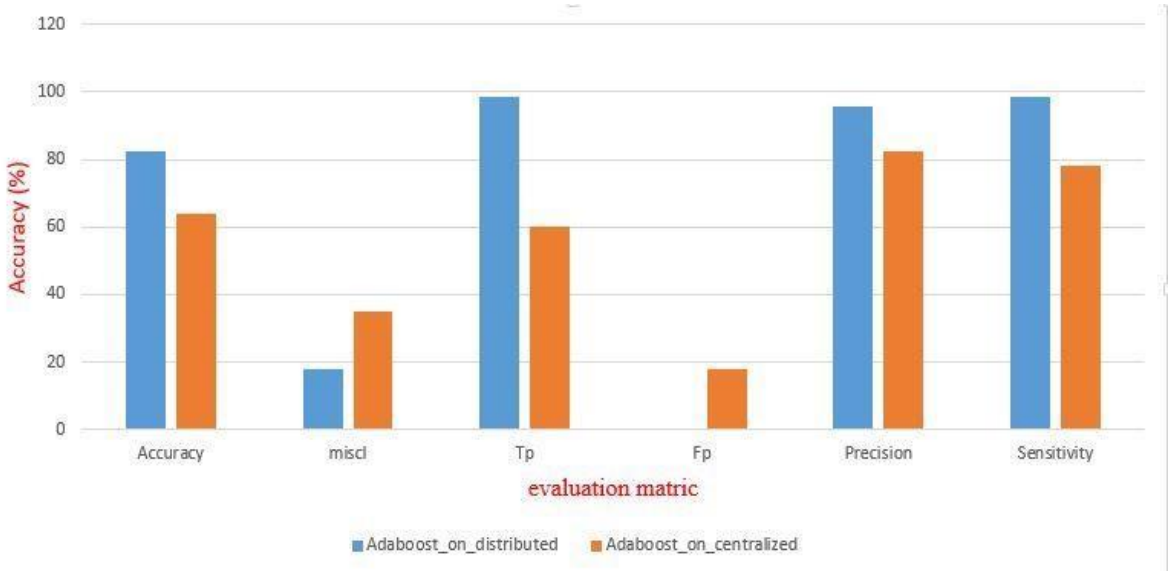


Figure 4.13: Comparison of Adaboost Implementation on Centralized and Distributed Architecture.

Figure 4.13 above show the result in plot of Adaboost implementation in centralized and distributed Architecture. Having latency and bandwidth the evaluation parameters, the result shows distributed architecture having 81.2 % accuracy against centralized architecture 67.8% accuracy. Precision and sensitivity for distributed was reported to be 0.97.3 and 0.98.3respectively and centralized architecture achieved 0.82.3 and 0.79.6 respectively. Hence, distributed architecture has lesser processing which translates into been computational cost effective in terms of latency and bandwidth consumption. Notably, the result shows that:

1. MapReduce processing Architecture is responsible for high computational cost such as Large Bandwidth and Latency Consumption in Data center as the result shows a difference of 31.44 min (1860, 440ms) increased Bandwidth consumption by Centralized Architecture.

2. Centralized Architecture is observed to be slow in processing as this is due to the inter-node communication that is avoidable if processing is distributed across geo- spatial and every node at the edge is given the responsibility of processing its captured data.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

This research work developed a Distributed Big Data Fusion Architecture for machine to machine communication using Ensemble Learning, which is aimed at mitigating the challenge of Bandwidth consumption and high Latency experienced in centralized data fusion domain.

5.2 Significant Contributions

The significant contribution of this research is as follows:

i. The developed distributed big data fusion architecture using ensemble learning to mitigate high Bandwidth consumption, network latency associated with centralized Data fusion architecture achieved a reduction of 5.44 mins (latency) as compared to 35.83 mins of centralized (MapReduce) architecture.

ii. The developed architecture has 88.1% accuracy improvement against 86.02 % accuracy in centralized architecture.

iii. This research work developed Fog-based distributed big data fusion architecture using WEKA GUI interface.

5.3 Recommendations for further work

i. The research conducted did not consider more than ten base learners as such its recommended that higher numbers be considered to represent more nodes.

REFERENCES

Abhishek Chandra, J. W., and Benjamin Heintz. (2012). Decentralized Edge Clouds. *IEEE Internet Computing,(5),70-73*

Ahmad, A., Paul, A., Rathore, M., & Chang, H. (2016). An efficient multidimensional big data fusion approach in machine-to-machine communication. *ACM Transactions on Embedded Computing Systems (TECS), 15*(2), 39.

Akita, R. M. (2002). User based data fusion approaches. In *Information Fusion, 2002. Proceedings of the Fifth International Conference on* IEEE. (Vol. 2, pp. 1457-1462).

Alsheikh, M. A., Lin, S., Niyato, D., & Tan, H.-P. (2014). Machine learning in wireless sensor networks: Algorithms, strategies, and applications. *IEEE Communications Surveys & Tutorials, 16*(4), pp.1996-2018.

Alturki, B., Reiff-Marganiec, S., & Perera, C. (2017). A hybrid approach for data analytics for internet of things. In *Proceedings of the Seventh International Conference on the Internet of Things* ACM (p. 7).

Anagnostopoulos, T., & Skourlas, C. (2014). Ensemble majority voting classifier for speech emotion recognition and prediction. *Journal of Systems and Information Technology, 16*(3), 222-232.

Asadi, A., Wang, Q., & Mancuso, V. (2014). A survey on device-to-device communication in cellular networks. *IEEE Communications Surveys & Tutorials, 16*(4), 1801-1819.

Asmita, S., & Shukla, K. K. (2014). Review on the Architecture, Algorithm and Fusion Strategies in Ensemble Learning. *International Journal of Computer Applications, 108*(8). 21-45

Assunção, M. D., Calheiros, R. N., Bianchi, S., Netto, M. A., & Buyya, R. (2015). Big Data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed PComputing, 79*, 3-15.

Azzedin, F. (2013). Towards a scalable HDFS architecture. In *Collaboration Technologies and Systems (CTS), 2013 International Conference on* IEEE. (pp. 155-161).

Barrett, P. (2006). Euclidean distance: Raw, normalised, and double-scaled coefficients. *Unpublished paper retrieved from http://www. pbmetrix. com/techpapers/Euclidean_Distance. pdf. Pp.45-60.*

Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing* ACM. (pp. 13-16).

Buyya, R., Ranjan, R., & Calheiros, R. N. (2010). Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In *International Conference on Algorithms and Architectures for Parallel Processing* Springer, Berlin, Heidelberg. (pp. 13-31).

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience, 41*(1), 23-50.

Castanedo, F. (2013). A review of data fusion techniques. *The Scientific World Journal, 2013*, (1-9).

Castiglione, A., Pizzolante, R., De Santis, A., Carpentieri, B., Castiglione, A., & Palmieri, F. (2015). Cloudbased adaptive compression and secure management services for 3D healthcare data. *Future Generation Computer Systems, 43*, 120-134.

Chen, M., Mao, S., & Liu, Y. (2014). Big data: A survey. *Mobile Networks and Applications , 19*(2), 171-209.

Chen, X.-W., & Lin, X. (2014). Big data deep learning: challenges and perspectives. *IEEE access, 2*, 514-525.

Chiang, M., & Zhang, T. (2016). Fog and IoT: An overview of research opportunities. *IEEE Internet of* Things *Journal, 3*(6), 854-864.

Dautov, R., & Distefano, S. (2017). Distributed Data Fusion for the Internet of Things.

In *International Conference on Parallel Computing Technologies* Springer, Cham.

(pp. 427-432).

Din, S., Ahmad, A., Paul, A., Rathore, M. M. U., & Jeon, G. (2017). A Cluster-Based Data Fusion

Technique to Analyze Big Data in Wireless Multi-Sensor System. *IEEE Access*, *5*, 5069-5083.

Dautov, R., & Distefano, S. (2017). Distributed Data Fusion for the Internet of Things.

In *International Conference on Parallel Computing Technologies* Springer, Cham.

(pp. 427-432).

Dofe, J., Frey, J., & Yu, Q. (2016). Hardware security assurance in emerging IoT applications.

In *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on* IEEE. (pp. 2050-2053).

Elkhatib, Y., Porter, B., Ribeiro, H. B., Zhani, M. F., Qadir, J., & Rivière, E. (2017). On using micro-clouds to deliver the fog. *IEEE Internet Computing*, *21*(2), 8-15.

Emani, C. K., Cullot, N., & Nicolle, C. (2015). Understandable big data: a survey. *Computer science review*, *17*, 70-81.

Esteban, J., Starr, A., Willetts, R., Hannah, P., & Bryanston-Cross, P. (2005). A review of data fusion models and architectures: towards engineering guidelines. *Neural Computing & Applications*, *14*(4), 273-281.

Evans, D. (2011). Evan D., "The internet of things: How the next evolution of the internet is changing everything,. *CISCO white paper*, *vol. 1*, pp.1–11.

Fasano, G., Accardo, D., Tirri, A. E., Moccia, A., & De Lellis, E. (2015). Radar/electro-optical data fusion for non-cooperative UAS sense and avoid. *Aerospace Science and Technology*, *46*, 436-450.

Firdhous, M., Ghazali, O., & Hassan, S. (2014). Fog computing: Will it be the future of

cloud computing. The Third International Conference on Informatics & Applications (ICIA2014).pp. 33-56

Frank, E., Hall, M., Holmes, G., Kirkby, R., Pfahringer, B., Witten, I. H., & Trigg, L. (2009)

.Weka-a machine learning workbench for data mining. In *Data mining and knowledge discovery handbook* Springer, Boston, MA (pp. 1269-1277).

Gao, X. (2014). Investigation and Comparison of Distributed NoSQL Database Systems: Indiana

University. Pp. 92-102

Krempl, G., Žliobaite, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, V. & Stefanowski,

J. (2014). Open challenges for data stream mining research. *ACM SIGKDD explorations newsletter*, *16*(1), 1-10.

Gollapudi, S. (2016). *Practical Machine Learning*: Packt Publishing Ltd. Pp. 23-40

Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K., & Buyya, R. (2017). iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Software: Practice and Experience, 47*(9), 1275-1296.

Habib ur Rehman, M., Jayaraman, P. P., & Medhat Gaber, M. (2017). RedEdge: A

Novel Architecture for Big Data Processing in Mobile Edge Computing Environments. *Journal of Sensor and Actuator Networks, 6*(3), 17.

Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., & Khan, S. U. (2015). The rise of "big data" on cloud computing: Review and open research issues. *Information Systems, 47*, 98-115.

Jackson, T. J., Chen, D., Cosh, M., Li, F., Anderson, M., Walthall, C., Hunt, E. R. (2004). Vegetation water content mapping using Landsat data derived normalized difference water index for corn and soybeans. *Remote Sensing of Environment, 92*(4), 475-482.

Ji, C., Li, Y., Qiu, W., Awada, U., & Li, K. (2012). Big data processing in cloud computing environments. In *Pervasive Systems, Algorithms and Networks (ISPAN), 2012 12th International Symposium on* IEEE. (pp. 17-23).

Ji, L., Zhang, L., & Wylie, B. (2009). Analysis of dynamic thresholds for the normalized difference water index. *Photogrammetric Engineering & Remote Sensing*, *75*(11), 1307-1317.

Kaehler, A., & Bradski, G. (2016). *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*: " O'Reilly Media, Inc.". pp. 21-32

Kim, G.-H., Trimi, S., & Chung, J.-H. (2014). Big-data applications in the government sector. *Communications of the ACM, 57*(3), 78-85.

Konda, V. G. R. K., Chejarla, V. R., Mandla, V. R., Voleti, V., & Chokkavarapu, N. (2018). Vegetation damage assessment due to Hudhud cyclone based on NDVI using Landsat-8 satellite imagery.*Arabian Journal of Geosciences*, *11*(2), 35.

Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering, 160*, 3-24.

Krishnan, V., & Sanyal, B. (2010). M2M technology: challenges and opportunities. *Tech Mahindra*, 18-21.

Kulkarni, S., & Agrawal, P. (2014a). *Analysis of TCP performance in data center networks*: Springer. Springer. (pp. 1-15):

Kulkarni, S., & Agrawal, P. (2014b). Introduction *Analysis of TCP Performance in Data Center Networks:* Springer. (pp. 1-15)

Llinas, J., Bowman, C., Rogova, G., Steinberg, A., Waltz, E., & White, F. (2004). Revisiting the JDL data fusion model II: space and naval warfare systems command San Diego ca. pp. 25-45

Luan, T. H., Gao, L., Li, Z., Xiang, Y., Wei, G., & Sun, L. (2015). Fog computing: Focusing on mobile users at the edge. *arXiv preprint arXiv:1502.01815*.

Mustafa, R., Min, Y., & Zhu, D. (2014). Obscenity detection using Haar-like features and gentle adaboost classifier. *The Scientific World Journal, 2014*. Pp. 22-65

Natale, M. (2015). *Computational tools for the interactive exploration of proteomics data and automatic bio-networks reconstruction.* Politecnico di Torino. Pp. 67-78

Novembre, J., & Stephens, M. (2008). Interpreting principal component analyses of spatial population genetic variation. *Nature genetics, 40*(5), 646-651.

OECD. (2012). "Machine-to-Machine Communications:Connecting Billions of Devices", : OECD Digital Economy Papers, No. 192, OECD Publishing. Pp. 76-80

Oh, S.-I., & Kang, H.-B. (2017). Object detection and classification by decision-level fusion for intelligent vehicle systems. *Sensors, 17*(1), 207.

Okafor, K., Achumba, I. E., Chukwudebe, G. A., & Ononiwu, G. C. (2017). Leveraging Fog Computing for Scalable IoT Datacenter Using Spine-Leaf Network Topology. *Journal of Electrical and Computer Engineering, 2017*. 57-65.

Osanaiye, O., Chen, S., Yan, Z., Lu, R., Choo, K. K. R., & Dlodlo, M. (2017). From cloud to fog computing: A review and a conceptual live VM migration framework. *IEEE Access*, *5*, 8284-8300.

Pettit, C., Stimson, R., Nino-Ruiz, M., Morandini, L., Widjaja, I., Delaney, P, Kvan, T. (2014). *Supporting urban informatics through a big data analytics online workbench.* Paper presented at the NSF Workshop on Big Data and Urban Informatics. Pp. 21-25

Polikar, R. (2012). Ensemble learning *Ensemble machine learning*. Springer. (pp. 1-34):

Lewitt, M., & Polikar, R. (2003). An ensemble approach for data fusion with Learn++.

In *International Workshop on Multiple Classifier Systems* Springer, Berlin, Heidelberg. (pp. 176-185).

Zhang, C., & Ma, Y. (Eds.). (2012). *Ensemble machine learning: methods and applications*. Springer Science & Business Media. Pp.32-56.

Rao, B. T., & Reddy, L. (2012). Survey on improved scheduling in Hadoop MapReduce in cloud environments. *arXiv preprint arXiv:1207.0780*. Pp.21-23.

Rodrigo N. Calheiros, R. R., Anton Beloglazov, C´esar A. F. De Rose, and Rajkumar Buyya. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software – practice and experience*, 41:23–50.

Rokach, L. (2005). Ensemble methods for classifiers *Data mining and knowledge discovery handbook* Springer. (pp. 957-980):

Roli, F., & Giacinto, G. (2002). Design of multiple classifier systems. *Series in Machine Perception and Artificial Intelligence, 47*, 199-226.

Romero, C., Ventura, S., & García, E. (2008). Data mining in course management systems: Moodle case study and tutorial. *Computers & Education, 51*(1), 368-384.

Sinanc, D., & Sagiroglu, S. (2013, November). A review on cloud security. In *Proceedings of the 6th International Conference on Security of Information and Networks* ACM. (pp. 321-325).

Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing, 8*(4). Pp.86-89.

Sneps-Sneppe, M., & Namiot, D. (2012). About M2M standards and their possible extensions. In *Future Internet Communications (BCFIC), 2012 2nd Baltic Congress on* IEEE. (pp. 187-193).

Ting, S., Tse, Y., Ho, G., Chung, S., & Pang, G. (2014). Mining logistics data to assure the quality in a sustainable food supply chain: A case in the red wine industry. *International Journal of Production Economics, 152*, 200-209.

Uzunkaya, C., Ensari, T., & Kavurucu, Y. (2015). Hadoop ecosystem and its analysis on tweets. *Procedia-Social and Behavioral Sciences, 195*, 1890-1897.

Vaquero, L. M., & Rodero-Merino, L. (2014). Finding your way in the fog: Towards a comprehensive definition of fog computing. *ACM SIGCOMM Computer Communication Review*, *44*(5), 27-32.

Vikram, S. (2015). Green computing. In *Green Computing and Internet of Things (ICGCIoT), 2015 International Conference on* IEEE. (pp. 767-772).

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* IEEE. (Vol. 1, pp. I-I).

Voyant, C., Notton, G., Kalogirou, S., Nivet, M.-L., Paoli, C., Motte, F., & Fouilloy, A. (2017). Machine learning methods for solar radiation forecasting: A review. *Renewable Energy*, *105*, 569-582.

Wald, L. (1998). A European proposal for terms of reference in data fusion. In *Commission VII Symposium" Resource and Environmental Monitoring"* (Vol. 32, No. 7, pp. 651-654).

Walunj Swapnil K, Y. A. H., andSonu Gupta. (2016). *Big Data: Characteristics, Challenges and Data Mining.* Paper presented at the International Journal of Computer Applications (0975 – 8887).

Wang, L., & Alexander, C. A. (2016). Big Data Analytics and Cloud Computing in Internet of Things. *American Journal of Information Science and Computer Engineering*, *2*(6), 70-78.

Wang, X. (2012). *Spatial analysis of travel behavior and response to traveler information*: Old Dominion University. Pp. 50-54.

Wheeler, A. (2014). Deep Collaborative Data Fusion in a Cloud Framework. In *Proceedings on the International Conference on Artificial Intelligence (ICAI)* (p. 1).

Wu, X., Zhu, X., Wu, G. Q., & Ding, W. (2014). Data mining with big data. *IEEE transactions on*

knowledge and data engineering, *26*(1), 97-107.

Wu, X., Zhu, X., Wu, G. Q., & Ding, W. (2014). Data mining with big data. *IEEE transactions on knowledge and data engineering*, *26*(1), 97-107.

Yang, B., Cao, C., Xing, Y., & Li, X. (2015). Automatic classification of remote sensing images using multiple classifier systems. *Mathematical Problems in Engineering*. Pp. 24-75

Yi, S., Li, C., & Li, Q. (2015). A survey of fog computing: concepts, applications and issues. In *Proceedings of the 2015 workshop on mobile big data* ACM. (pp. 37-42).

Zhang, X., Kunjithapatham, A., Jeong, S., & Gibbs, S. (2011). Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing. *Mobile Networks and Applications*, *16*(3), 270-284.

Zhu, J., Chan, D. S., Prabhu, M. S., Natarajan, P., Hu, H., & Bonomi, F. (2013). Improving web sites performance using edge servers in fog computing architecture. In *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering* IEEE. (pp. 320-323).

Zhu, X., Chan, D. S., Hu, H., Prabhu, M. S., Ganesan, E., & Bonomi, F. (2015). Improving Video Performance with Edge Servers In The Fog Computing Architecture. *Intel Technology Journal*, *19*(1). 22-34

APPENDICES

Appendix A

Centralized Architecture Source code

INSTALLATION

Part- A Setting up Ubuntu Server Machine for Hadoop

Step 1: Login with Root

#su - password:

Step 2: Update the System

#apt-get update

Step 3: Installation of OpenSSH Server

#apt-get install openssh-server

Step 4: After the SSH server is installed, the configuration can be done by editing sshd_config that resides in the /etc/ssh directory. It is important to note sshd_config is for the SSH server while ssh_config is for the SSH client. Create a backup copy of sshd_config that can be used to restore your configuration by running the command below.

#cp /etc/ssh/sshd_config /etc/ssh/sshd_config.factory-defaults

Step 5: Open sshd_config in a text editor by running the command below for Ubuntu versions

#gedit /etc/ssh/sshd_config

Disable password authentication by changing this line in the configuration file

#PasswordAuthentication yes to PasswordAuthentication no.

Add the following Lines to the End:

AllowUsers anand

PermitRootLogin no

PubkeyAuthentication yes

Change LogLevel INFO to LogLevel VERBOSE.

Save and Restart SSH

#systemctl restart ssh

Step 6: After SSH is configured, an SSH key for the user eduonix is generated by running the commands below. They create an RSA key pair without a password. A password would be required every time Hadoop interacts with its nodes so we can save ourselves the bother of being prompted for a password every time.

#su - anand

#ssh-keygen -t rsa -P ""

After the key has been created, we use it to enable SSH access to the local machine by running the command below which adds it to the list of known keys.

#cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys

Testing SSH

#ssh localhost

Part-B Installing Apache Hadoop

Step 1: Adding Jave Repository and Installing Oracle Java 8

#add-apt-repository ppa:webupd8team/java

#apt-get update

#apt-get install oracle-java8-installer

#java -version

Step 2: Downloading and Installing Hadoop

Login with user "anand"

#cd Desktop

#sudo tar xzvf hadoop-2.7.4.tar.gz

#sudo mkdir /usr/local/hadoop

#sudo mv hadoop-2.7.0 /usr/local/hadoop

#sudo chown -R anand /usr/local/hadoop

Step 3: You need to edit the .bashrc file for the user anand, so open it in a text editor by running gedit ~/.bashrc from a terminal.

Check Where the Java is Installed

#readlink -f /usr/bin/java

#gedit ~/.bashrc Type

the following:

export JAVA_HOME=/usr/lib/jvm/java-8-oracle export

HADOOP_INSTALL=/usr/local/hadoop/hadoop-2.7.4 export

PATH=$PATH:$HADOOP_INSTALL/bin export

PATH=$PATH:$HADOOP_INSTALL/sbin export

HADOOP_MAPRED_HOME=$HADOOP_INSTALL export

HADOOP_COMMON_HOME=$HADOOP_INSTALL export

HADOOP_HDFS_HOME=$HADOOP_INSTALL export

YARN_HOME=$HADOOP_INSTALL export

HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native export

HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib/native" export

YARN_LOG_DIR=/hadoop/yarn/log export HADOOP_LOG_DIR=/hadoop/yarn/log

export HADOOP_MAPRED_LOG_DIR=/hadoop/yarn/log

Save the file and exit

Save the Changes to Bashrc File

#source ~/.bashrc

Step 4: Exporting JAVA_HOME Path

The directory /usr/local/Hadoop/Hadoop-2.7.4/etc/Hadoop contains configuration files. Open Hadoop-env.sh in a text editor and set JAVA_HOME variable by adding the line below. This specifies the Java installation that will be used by Hadoop.

#cd /usr/local/hadoop/hadoop-2.7.4/etc/hadoop

#nano hadoop-env.sh export

JAVA_HOME=/usr/lib/jvm/java-8-oracle save

and exit

Step 5: Create a directory that will act as the base for other temporary directories and assign it to the user anand by running the commands below.

#sudo mkdir -p /app/Hadoop/tmp

#sudo shown anand /app/Hadoop/tmp

Part-C Editing the XML Based Hadoop Configuration Files

All the Hadoop configuration files reside under usr/local/hadoop/hadoop-2.7.4/etc/hadoop

#cd /usr/local/hadoop/hadoop-2.7.4/etc/hadoop gedit

core-site.xml

A. Edit Core-site.xml

<property>

<name>fs.default.name</name>

<value>hdfs://localhost:9000</value>

</property>

B. Edit mapred-site.xml

#cp /usr/local/hadoop/hadoop-2.7.4/etc/hadoop/mapred-site.xml.template

/usr/local/hadoop/hadoop-2.7.4/etc/hadoop/mapred-site.xml

#nano mapred-site.xml

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

//may leave this out

<property>

<name>mapred.job.tracker</name>

<value>localhost:9001</value>

</property>

C. Edit yarn-site.xml

#nano yarn-site.xml

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

Part-D The hdfs-site.xml is used to specify the name node and data node directories. Before modifying this file, we create the name node and data node directories.

#sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode

#sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode

#sudo chown -R anand /usr/local/hadoop_store

Modify hdfs-site.xml

#nano hdfs-site.xml

<property>

<name>dfs.replication</name>

<value>4</value>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value> file:/usr/local/hadoop_store/hdfs/namenode </value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value> file:/usr/local/hadoop_store/hdfs/datanode </value>

</property>

Save and Exit

Part-E Final Steps

Format the file system by running hdfs namenode -format to initialize the file system

#hdfs namenode -format

Start the single node cluster

#start-dfs.sh

#start-yarn.sh

#jps

HIPI installation:

HiPi Commands https://stackoverflow.com/questions/2643502/git-permission-denied-publickey git clone git@github.com:uvagfx/hipi.git sudo cp hipi.pub /home/abiodun/hipi.pub cat hipi.pub | xclip sudo apt-get install xclip && cat hipi.pub | xclip cat hipi.pub | xclip cat hipi.pub | pbcopy ssh-keygen

ls

sudo nano hipi

cd ~/.ssh cd ./

sudo nano hipi.pub

Enter file in which to save the key (/home/salefu/.ssh/id_rsa):

$ xclip -sel clip < ~/.ssh/id_rsa.pub

Copies the contents of the id_rsa.pub file to your clipboard

$> cd hipi

$> gradle

:core:processResources UP-TO-DATE

:core:classes

Copying from local to HIB: L50-B:~/hipi$ tools/hibImport.sh ~/Desktop/TestImage

TestImage.hib

Input image directory: /home/salefu/Desktop/TestImage

Input FS: local FS

Output HIB: TestImage.hib

Overwrite HIB if it exists: false

** added: aerial photo.jpg

** added: aerial photo2.jpg

** added: aerial photo3.jpg

Created: TestImage.hib and TestImage.hib.dat

To verify "TestImage.hib" use the command: -B:~/hipi$ hadoop fs -ls

To inspect the content of the created file: $> tools/hibInfo.sh TestImage.hib --show-meta

To create a source directory hierarchy for programs:

$> mkdir -p examples/helloWorld/src/main/java/org/hipi/examples

$> cd examples/helloWorld/

add a Gradle build task for our new program by creating the file examples/HelloWorld/build.gradle with the following contents:

$> sudo gedit build. gradle jar { manifest {

attributes("Main-Class": "org.hipi.examples.HelloWorld")

}

}

$> cd ../../

$> sudo gedit settings.gradle

We also need to update the settings.gradle file in the root directory to tell Gradle about this new build target:

include ':core', ':tools:hibImport', ... ':examples:covar', ':examples:helloWorld'

$> cd examples/helloWorld/src/main/java/org/hipi/examples/

examples/helloWorld/src/main/java/org/hipi/examples$> sudo gedit HelloWorld.java Next, create a new Java source file at examples/helloWorld/src/main/java/org/hipi/examples/HelloWorld.java that contains the following code:

```
package org.hipi.examples; import
org.apache.hadoop.conf.Configured; import
org.apache.hadoop.util.Tool; import
org.apache.hadoop.util.ToolRunner; public
class HelloWorld extends Configured
implements Tool { public int run(String[]
args) throws Exception {
System.out.println("Hello HIPI Abbey!");
    return 0;
  }
public static void main(String[] args) throws Exception {
    ToolRunner.run(new HelloWorld(), args);
    System.exit(0);
  }
```

}

Can change directory using: cd ../../

$> cd examples/helloWorld

$> gradle jar

:examples:HelloWorld: jar

BUILD SUCCESSFUL

If the build is successful, it will produce the JAR file

examples/helloWorld/build/libs/helloWorld.jar directory. Run this program using the following

command from within the examples/HelloWorld directory:

$ cd examples mkdir -p

helloImage/src/main/java/org/hipi/examples

$> cd helloImage/ $

sudo gedit build.gradle

jar {

manifest { attributes("Main-Class":

"org.hipi.examples.HelloImage")

} }

cd ../../

hipi$ sudo gedit settings.gradle

':examples:helloImage'

$> cd examples/helloImage/src/main/java/org/hipi/examples/ sudo

gedit HelloImage.java package org.hipi.examples; import

org.hipi.image.FloatImage; import

```
org.hipi.image.HipiImageHeader; import
org.hipi.imagebundle.mapreduce.HibInputFormat; import
org.apache.hadoop.conf.Configured; import
org.apache.hadoop.util.Tool; import
org.apache.hadoop.util.ToolRunner; import
org.apache.hadoop.fs.Path; import
org.apache.hadoop.io.IntWritable; import
org.apache.hadoop.io.Text; import
org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; import
org.apache.hadoop.mapreduce.Job; import
org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer; import
org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; import
java.io.IOException; public class HelloImage extends Configured implements Tool {
public static class HelloWorldMapper extends Mapper<HipiImageHeader, FloatImage,
IntWritable, FloatImage> {     public void map(HipiImageHeader key, FloatImage
value, Context context)       throws IOException, InterruptedException {
            // Verify that image was properly decoded, is of sufficient size, and has three color
channels (RGB)          if (value != null && value.getWidth() > 1 && value.getHeight() >
1 && value.getNumBands() == 3) {
```

```
            // Get dimensions of image
int w = value.getWidth();              int h =
value.getHeight();              // Get pointer
to image data              float[] valData =
value.getData();
            // Initialize 3 element array to hold RGB pixel average
float[] avgData = {0,0,0};
            // Traverse image pixel data in raster-scan order and update running average
for (int j = 0; j < h; j++) {                  for (int i = 0; i < w; i++) {
                  avgData[0] += valData[(j*w+i)*3+0]; // R
avgData[1] += valData[(j*w+i)*3+1]; // G
avgData[2] += valData[(j*w+i)*3+2]; // B
                }
            }
            // Create a FloatImage to store the average value
            FloatImage avg = new FloatImage(1, 1, 3, avgData);
            // Divide by number of pixels in image
avg.scale(1.0f/(float)(w*h));              // Emit
record to reducer              context.write(new
IntWritable(1), avg);
         } // If (value != null...
      } // map()
   }
```

```
 public static class HelloWorldReducer extends Reducer<IntWritable, FloatImage, IntWritable,
Text> {     public void reduce(IntWritable key, Iterable<FloatImage> values, Context context)
throws IOException, InterruptedException {
        // Create FloatImage object to hold final result
        FloatImage avg = new FloatImage(1, 1, 3);
        // Initialize a counter and iterate over IntWritable/FloatImage records from mapper
        int total = 0;          for
(FloatImage val : values) {
avg.add(val);               total++;
        }       if (total
> 0) {
            //    Normalize    sum    to    obtain    average
avg.scale(1.0f / total);
            //  Assemble  final  output  as  string
float[] avgData = avg.getData();
            String result = String.format("Average pixel value: %f %f %f", avgData[0], avgData[1],
avgData[2]);
            //  Emit  output  of  job  which  will  be  written  to  HDFS
context.write(key, new Text(result));
        }
     } // reduce()
  }
 public int run(String[] args) throws Exception {
```

```
   // Check input arguments
if (args.length != 2) {
     System.out.println("Usage: HelloImage <input HIB> <output directory>");
     System.exit(0);
   }
   // Initialize and configure MapReduce job
   Job job = Job.getInstance();
   // Set input format class which parses the input HIB and spawns map tasks
job.setInputFormatClass(HibInputFormat.class);
   // Set the driver, mapper, and reducer classes which express the computation
job.setJarByClass(HelloImage.class);     job.setMapperClass(HelloWorldMapper.class);
job.setReducerClass(HelloWorldReducer.class);
   // Set the types for the key/value pairs passed to/from map and reduce layers
job.setMapOutputKeyClass(IntWritable.class);
job.setMapOutputValueClass(FloatImage.class);
job.setOutputKeyClass(IntWritable.class);     job.setOutputValueClass(Text.class);
   // Set the input and output paths on the HDFS
   FileInputFormat.setInputPaths(job, new Path(args[0]));
   FileOutputFormat.setOutputPath(job, new Path(args[1]));
// Execute the MapReduce job and block until it complets
boolean success = job.waitForCompletion(true);
   // Return success or failure
return success ? 0 : 1;
```

```
  }
  public static void main(String[] args) throws Exception {
    ToolRunner.run(new HelloImage(), args);
    System.exit(0);
  }
}
```

```
cd ../../
~/hipi/examples/helloImage$
gradle jar
```

If the build is successful, it will produce the JAR file examples/helloWorld/build/libs/helloWorld.jar directory. Run this program using the following command from within the examples/HelloWorld directory

If build however FAILS; Solution.....

Add dependencies in build.gradle:

```
$> cd helloImage/ $ sudo gedit build.gradle jar{
manifest{ attributes("Main-Class":
"org.hipi.examples.HelloImage")
} dependencies{
compile project(':core')
} from { configurations.compile.collect { it.isDirectory() ? it :
zipTree(it) }
}
}
```

$> hadoop jar build/libs/helloImage.jar TestImage.hib sampleimages_average

///Error was resolved by recompiling grade and rerun as below

~/hipi/examples/helloImage$ hadoop jar build/libs/helloImage.jar TestImage.hib sampleimages_average

17/12/04 15:21:16 INFO mapreduce.Job: map 0% reduce 0%

17/12/04 15:21:41 INFO mapreduce.Job: map 100% reduce 0%

17/12/04 15:22:33 INFO mapreduce.Job: map 100% reduce 100%

17/12/04 15:22:37 INFO mapreduce.Job: Job job_1512346699264_0001 completed successfully

WRONG_REDUCE=0

File Input Format Counters

Bytes Read=747349

File Output Format Counters

Bytes Written=50

$> hadoop fs -ls sampleimages_average

To check result: helloImage$ hadoop dfs -cat sampleimages_average/part-r-00000

.....D HelloImage2 run

~/hipi/examples/helloImage2$ hadoop jar build/libs/helloImage2.jar TestImage.hib sampleimages_average2

To check result: helloImage2$ hadoop dfs -cat sampleimages_average2/part-r-00000

OpenCV:

$ mkdir opencv && cd opencv

$ unzip /home/abiodun/DonPelsHadoop/opencv-3.3.1.zip mv

-i ~/MyFile ~/OtherFolder/MyFile

opencv]$ tar -xvzf /home/salefu/DonPelsHadoop/cmake-3.10.0.tar.gz B:~$

sudo apt-get install python-numpy

opencv-3.3.1$ cmake -D CMAKE_BUILD_TYPE=RELEASE -D

BUILD_NEW_PYTHON_SUPPORT=ON -D CMAKE_INSTALL_PREFIX=/usr/local ./

opencv-3.3.1$ cmake -D BUILD_SHARED_LIBS=OFF -D

CMAKE_BUILD_TYPE=RELEASE -D BUILD_NEW_PYTHON_SUPPORT=ON -D

CMAKE_INSTALL_PREFIX=/usr/local ./

$ sudo apt install cmake mkdir my_build_dir

cd my_build_dir cmake .. -

DCMAKE_BUILD_TYPE=Release

To remake:Previous build has created CMakeCache.txt and a directory CMakeFiles. Remove this file and and directory as:

../cmake-3.10.0/my_build_dir/build/cmake -DBUILD_SHARED_LIBS=OFF

/opencv-3.3.1/my_build_dir$ cmake .. -D BUILD_SHARED_LIBS=OFF -D

CMAKE_BUILD_TYPE=RELEASE -D BUILD_NEW_PYTHON_SUPPORT=ON -D

CMAKE_INSTALL_PREFIX=/usr/local ./ opencv-3.3.1/my_build_dir$ cmake -D

BUILD_SHARED_LIBS=OFF opencv-2.4.11]$ ../cmake-3.3.1-Linux-x86_64/bin/cmake -

DBUILD_SHARED_LIBS=OFF ~/opencv/opencv-3.3.1$ ../cmake-3.10.0/bin/cmake -

DBUILD_SHARED_LIBS=OFF cmake . -DBUILD_SHARED_LIBS=ON && make

cmake -DBUILD_SHARED_LIBS=OFF -DBUILD_STATIC_LIBS=ON

abiodun@abiodun-SATELLITE-L50-B:~/opencv/opencv-3.3.1$

-ptx/dnMDrIGhyVQ https://github.com/google/brotli/issues/542

Usage: /home/abiodun/opencv/cmake-3.10.0/bootstrap [<options>...] [-- <cmake-options>...]

works inside cmake-3.10.0: cmake -D BUILD_SHARED_LIBS=OFF -D

CMAKE_BUILD_TYPE=RELEASE -D BUILD_NEW_PYTHON_SUPPORT=ON -D

CMAKE_INSTALL_PREFIX=/usr/local ./ works too: ~/opencv/opencv-3.3.1/abbey$

cmake -DBUILD_SHARED_LIBS=OFF -

DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=/usr/local ..

cmake source:/home/salefu/opencv/cmake-3.10.0/Source cmake .. -

DBUILD_SHARED_LIBS=OFF -DCMAKE_BUILD_TYPE=Release

DCMAKE_INSTALL_PREFIX=/home/salefu/opencv/cmake-3.10.0/Source works

too: ~/opencv/opencv-3.3.1/abbey$ cmake -DBUILD_SHARED_LIBS=OFF -

DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=/usr/local ..

Scanning dependencies of target gen-pkgconfig

[0%] Generate opencv.pc

[0%] Built target gen-pkgconfig

Scanning dependencies of target zlib

[1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/adler32.c.o

3.3.1/abbey$ make [100%] Building CXX object

apps/version/CMakeFiles/opencv_version.dir/opencv_version.cpp.

o

[100%] Linking CXX executable ../../bin/opencv_version

[100%] Built target opencv_version

This will create a jar containing the Java interface (bin/opencv-2411.jar) and a native dynamic library containing Java bindings and all the OpenCV stuff (lib/libopencv_java2411.so). We'll use these files to build and run an OpenCV program. Opencv/opencv-3.3.1/abbey$ ls lib | grep .so libopencv_java331.so

~/opencv/opencv-3.3.1/abbey$ ls bin | grep .jar

opencv-331.jar opencv-331.jar.dephelper

Running OpenCV Face Detection

Program>>

/opencv$ mkdir sample && cd sample sample]$

vi build.xml

Note: build.xml content

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Main" basedir="." default="rebuild-run">
  <property name="src.dir"     value="src"/>
  <property name="lib.dir"     value="${ocvJarDir}"/>
  <path id="classpath">
     <fileset dir="${lib.dir}" includes="**/*.jar"/>
  </path>
  <property name="src.dir" location="src" />
  <property name="build.dir" location="bin" />
  <property name="build.dir"   value="build"/>
  <property name="classes.dir" value="${build.dir}/classes"/>
  <property name="jar.dir"     value="${build.dir}/jar"/>
```

```
  <property name="main-class"  value="${ant.project.name}"/>
  <target name="clean">
    <delete dir="${build.dir}"/>
  </target>
  <target name="compile">
    <mkdir dir="${classes.dir}"/>
    <javac includeantruntime="false" srcdir="${src.dir}" destdir="${classes.dir}"
classpathref="classpath"/>
  </target>
  <target name="jar" depends="compile">
    <mkdir dir="${jar.dir}"/>
    <jar destfile="${jar.dir}/${ant.project.name}.jar" basedir="${classes.dir}">
      <manifest>
        <attribute name="Main-Class" value="${main-class}"/>
      </manifest>
    </jar>
  </target>
  <target name="run" depends="jar">
    <java fork="true" classname="${main-class}">
      <sysproperty key="java.library.path" path="${ocvLibDir}"/>
      <classpath>
        <path refid="classpath"/>
        <path location="${jar.dir}/${ant.project.name}.jar"/>
```

```
        </classpath>
      </java>
  </target>
  <target name="rebuild" depends="clean,jar"/>
  <target name="rebuild-run" depends="clean,run"/>
</project>
```

Note: to save the content, use SHIFT ZZ

$> mkdir -p examples/helloWorld/src/main/java/org/hipi/examples

$> cd examples/helloWorld/ add a Gradle build task for our

new program by creating the file

examples/HelloWorld/build.gradle with the following contents:

$> sudo gedit build. gradle jar { manifest {

attributes("Main-Class": "org.hipi.examples.HelloWorld")

 }

}

$> cd ../../

$> sudo gedit settings.gradle

We also need to update the settings.gradle file in the root directory to tell Gradle about this new build target:

include ':core', ':tools:hibImport', ... ':examples:covar', ':examples:helloWorld'

$> cd examples/helloWorld/src/main/java/org/hipi/examples/

examples/helloWorld/src/main/java/org/hipi/examples$> sudo gedit HelloWorld.java Next,

create a new Java source file at

examples/helloWorld/src/main/java/org/hipi/examples/HelloWorld.java that contains the following code:

```
package org.hipi.examples; import
org.apache.hadoop.conf.Configured; import
org.apache.hadoop.util.Tool; import
org.apache.hadoop.util.ToolRunner; public class HelloWorld
extends Configured implements Tool {  public int run(String[]
args) throws Exception {     System.out.println("Hello HIPI
Abbey!");     return 0;
  }
  public static void main(String[] args) throws Exception {
    ToolRunner.run(new HelloWorld(), args);
    System.exit(0);
  }
}
```

Can change directory using: cd ../../

```
$> cd examples/helloWorld
$> gradle jar
:examples:HelloWorld: jar
BUILD SUCCESSFUL
```

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

```
 DetectFaces.java import
org.opencv.core.Core; import
org.opencv.core.Mat; import
org.opencv.core.Scalar; import
org.opencv.highgui.*; import
org.opencv.core.MatOfRect; import
org.opencv.core.Point; import
org.opencv.core.Rect; import
org.opencv.objdetect.CascadeClassifier; import
java.io.File;
/**
* Created by dmalav on 4/30/15.
*/ public class DetectFaces {
public void run(String imageFile) {
      System.out.println("\nRunning DetectFaceDemo");
    // Create a face detector from the cascade file in the resources
      // directory.
      String xmlPath = "/home/cloudera/project/opencv-examples/lbpcascade_frontalface.xml";
      System.out.println(xmlPath);
      CascadeClassifier faceDetector = new CascadeClassifier(xmlPath);
      Mat image = Highgui.imread(imageFile);
      // Detect faces in the image.
```

```
        // MatOfRect is a special container class for Rect.
MatOfRect faceDetections = new MatOfRect();
faceDetector.detectMultiScale(image, faceDetections);
        System.out.println(String.format("Detected %s faces", faceDetections.toArray().length));
        // Draw a bounding box around each face.
for (Rect rect : faceDetections.toArray()) {
            Core.rectangle(image, new Point(rect.x, rect.y), new Point(rect.x + rect.width, rect.y +
rect.height), new Scalar(0, 255, 0));
        }
        File f = new File(imageFile);
System.out.println(f.getName());        //
Save the visualized detection.
        String filename = f.getName();
        System.out.println(String.format("Writing %s", filename));
        Highgui.imwrite(filename, image);

    }
}
Main.java import org.opencv.core.Core;
import java.io.File; public class Main {
public static void main(String... args) {
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
if (args.length == 0) {
```

```
  System.err.println("Usage Main /path/to/images");
  System.exit(1);
}
    File[] files = new File(args[0]).listFiles();
showFiles(files);
  }  public static void showFiles(File[]
files) {      DetectFaces faces = new
DetectFaces();
    for (File file : files) {
if (file.isDirectory()) {
          System.out.println("Directory: " + file.getName());
showFiles(file.listFiles()); // Calls same method again.
        } else {
          System.out.println("File: " + file.getAbsolutePath());
          faces.run(file.getAbsolutePath());}}}}
```

Appendix B

Table X: result of ensemble method

TABLE X1: BOOST

No.BL	Accuracy	misc	RMSE	TP	FP	PRECISION
1	80.5	19.5	0.255	0.911	0.019	0.935
3	83.95	16.05	0.0571	0.946	0.0228	0.928
5	86.85	13.15	0.196	0.964	0.014	0.952
8	87.05	12.95	0.1932	0.972	0.014	0.953
10	88.1	11.9	0.1885	0.972	0.013	0.957

TABLE X2: RANDOM FOREST

No.BL	ACCURACY	MISCL	RMSE	TP	FP	PRECISION
1	83.4	16.6	0.2252	0.957	0.012	0.959
3	85.7	14.3	0.187	0.983	0.010	0.968
5	87.45	12.55	0.1773	0.983	0.008	0.972
8	87.45	12.55	0.1715	0.987	0.010	0.968
10	87.45	12.55	0.1713	0.987	0.010	0.966

TABLE X3: BAGGING.

No.BL	ACCURACY	MIS	RMSE	TP	FP	PRECISION
1	82.55	17.45	0.2267	0.959	0.028	0.911
3	85.15	14.85	0.192	0.974	0.012	0.961
5	86.25	13.75	0.1851	0.983	0.014	0.954
8	86.85	13.15	0.18	0.983	0.014	0.956
10	86.95	13.05	0.1778	0.978	0.013	0.958

Table X4: IMPLEMENTATION OF SINGLE BASE LEARNERS

ALGORITHMS	ACCURACY	MISCL	RMSE	TP	FP	PRECISION
Bagging	82.55	17.45	0.2267	0.959	0.028	0.911
Boosting	80.5	19.5	0.255	0.911	0.019	0.935
SVM	82.3	17.7	0.3187	0.957	0.030	0.985
NaiveBayes	82.4	17.6	0.2048	0.987	0.016	0.950
Random Forest	83.4	16.6	0.2252	0.957	0.012	0.959