Cloud-based Machine Learning Architecture for Big Data Analysis

Rezvan Pakdel
MSC
113220306

Thesis submitted for the degree of Doctor of Philosophy

NATIONAL UNIVERSITY OF IRELAND, CORK

FACULTY OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

November 2019

Head of Department: Prof Cormac J. Sreenan

Supervisor: Dr John Herbert

Research supported by IRISH RESEARCH COUNCIL (IRCSET)

Contents

List of Figures

This is to certify that the work I am submitting is my own and has not been submitted for another degree, either at University College Cork or elsewhere. All external references and sources are clearly acknowledged and identified within the contents. I have read and understood the regulations of University College Cork concerning plagiarism.

Rezvan Pakdel

I dedicate this work to my family, friends, more especially to my son, Sherwin Hashemi.

Abstract

The use of machine-learning that leverages large amounts of data (big data) is increasingly important in many areas of business and research. To help cope with the demanding resources required by these applications, solutions including hardware platforms (e.g. graphics cards), more efficient algorithms (e.g. deep learning algorithms), and special software environments (e.g. tensor flow) have been developed. In addition, for specific applications, special optimisations are often developed based on the requirements of the particular application. This thesis also addresses the challenge of efficiency of machine learning over big data but does so in a way that is complementary to specialised hardware and algorithms, and in a way that is also independent of application and data type. The thesis has developed several types of general optimisations and implemented these on top of an underlying generic machine learning architecture. The generic machine learning architecture includes stages for segmentation, feature extraction, model building and classification. The optimisation components enhance this architecture in a general way that works with any datatype and any dataset, and where the optimisation responds to the needs of the particular application, and is self-adjusting for the particular dataset being processed. The optimisations developed are: model optimisation; feature optimisation; resources optimisation; cloud platform cost-benefit optimisation. Model optimisation involves evaluating multiple models in parallel, and using feedback on model performance to choose the best ones based on the dataset being processed. Feature optimisation involves evaluating various features and combinations of features, and then choosing those features that are most effective for classification. Resources optimisation involves dynamically adjusting compute instances to respond to the demands of an application. Cloud platform cost-benefit optimisation involves evaluating the cost of available public cloud compute instances, and determining appropriate cost-efficient instances depending on the needs of an application. General techniques of sampling, evaluation and feedback are used in several optimisation components. The underlying framework and optimisations have been implemented and deployed in a private cloud environment. Evaluation on various datasets (image and text datasets) has shown these optimisation components to be effective, and provide useful generic components that can work in conjunction with other optimisations to address the challenging demands of machine learning over big data.

Acknowledgements

I would like to thank my supervisor, Dr John Herbert, for the patient guidance, encouragement and advice he has provided throughout my work. He has been a tremendous mentor for me. I would like to thank Dr Babak Shirazi, who provided support for the pathology-based experiment. I would like to thank my collogues, Dr Dapeng Dong and Dr Jason J. Quinlan, for their thoughtful suggestions. I have enjoyed many interesting discussions with the members of the Mobile & Internet Systems Laboratory (MISL) during my research. I owe many thanks to the MISL administrative manager, Mary Noonan, for her support, and IT staff of the Computer Science department who provided me with equipment for part of my work. My work would not have been possible without the financial support provided by the Irish Research Council(IRCSET). Finally, I must express my gratitude to my family and my husband, Mohammad Hashemi, who have always supported me in pursuing my interests. Without their love, encouragement, patience, and understanding, this thesis would not have been possible.

Chapter 1

Introduction

The modern world is generating increasing amounts of digital data from many different sources, including social media, sensors, and smart devices. This data is commonly referred to as "big data" since it exceeds the capabilities of traditional computing due to its volume, its velocity (the speed at which it is generated) and its variety [Dau18, Yu14]. The variety of data includes structured (e.g. relational data), semi-structured (e.g. XML data), and unstructured data (e.g. text and image) [Mis18]. One of the main ways of exploiting big data, especially unstructured data, is through Machine Learning (ML), and that makes this area, and its need for optimisation, a very important topic for research.

1.1 Background

With the rapid development of the Internet, the size of datasets has increased from KB level to TB, even PB level [GEN13]. Daily generated data exceeds 2.5 quintillion bytes. Over 90% of this data was generated over the last two years. Considering the acceleration in the growth of the Internet of Things (IoT), there is going to be a massive amount of data generated in the future [BIG18].

Big data, especially unstructured data [SKIW17], is seen as a new valuable resource which can be exploited to benefit society and corporations. Scientists and companies are using high-performance parallel architectures and frameworks (e.g. Hadoop [Gho11, Wan10] and Spark [Gou18, Hak18]) to quickly, securely, and cost-effectively process data. These frameworks support distributed data-intensive applications and allow users to use scalable hardware resources to process their big data [VC15].

Machine learning algorithms examine large quantities of data to identify patterns. They use the extracted features from data, learn the patterns in the data, and create models for data recognition and classification. In supervised learning, algorithms use a pre-labelled dataset to create a model that enables forecasting. The objective in supervised learning is to predict the label of a new unseen data item, and its success depends on the representation of features within the data. Furthermore, the ultimate challenge is how a mixture of software-based and hardware-based technologies can create a state-of-the-art technical solution that can support data processing in terms of providing accurate results along with the efficiency in cost and performance.

1.2 State of the art

Big data processing requires a system with ample computing resources to get the results quickly and without exceeding the budget. Designing such a system and providing its required computing power are challenging using traditional computing paradigms. Cloud computing can solve this problem by using a network of remote servers hosted on the Internet to manage, store, and process data. Machine learning is a trending area of research, especially when used with "Big Data". The dynamic provision of virtualized resources provided by cloud computing infrastructures enables cloud-based applications to improve and reduce the number of resources used automatically [CCD+12]. Processing big data in a distributed manner has also been widely studied [PIP16]. Useful applications of using big data include self-driving vehicles [FBG+17], healthcare [SRB19, GPP18, RAG14], social media [BRW+15], and many other areas such as finance, marketing and sales. There is a wide range of applications to which machine learning can be applied, such as document classification, emotion analysis [LCT18], natural language processing (NPL) [ZH18], speech recogniton [NSA+19], and face detection [FRMT18, AV17]. For instance, in the area of social media, Facebook's DeepFace is able to train its algorithms using large datasets to identify faces with an accuracy of up to 97.35%. Several distributed machine learning frameworks (e.g. Jubatus, Apache Mahout, and H2O) have been developed [PIP16]. Most of them offer distributed and in-memory computations to facilitate developers to run machine learning applications efficiently.

1.3 Challenges

Existing distributed machine learning frameworks leverage large-scale parallelism (e.g. Hadoop Map Reduce) for implementing distributed machine learning algorithms. However, their focus is to support ML expert developers and practitioners who have sufficient knowledge of machine learning and data processing. Ordinary users with not enough skills of machine learning, who want to process their large datasets using ML, might not desire to get involved in the processing details.

Using machine learning over big data introduces several challenges, including choosing an appropriate paradigm suitable for a problem, identifying a useful set of discriminatory features practical for the machine learning paradigm, writing a piece of code to solve that problem, deploying the solution, and executing it.

Performing the feature extraction and machine learning operations require sufficient computing resources and can be time-consuming and costly. Currently, distributed machine learning platforms (e.g. Spark, TensorFlow) include different machine learning stages. For example, a stage for training a model by processing a pre-labelled dataset and a stage for classifying an unlabeled dataset. However, they usually exclude the essential basic stage for data preprocessing, such as data feature extraction.

The dynamic nature of the cloud environment creates a challenge for cloud service providers in the efficient management of resources. Since users pay for every resource they consume per hour, it is essential that users use them efficiently. It is a challenge for cloud users to choose the best cloud resources to run their application while keeping their overall costs low. There are many competitive pricing models, and public cloud platforms (such as Amazon and Azure) offer an array of price options to suit users. These cost models have many parameters, including CPUs, memory and disk size, and operating system. The challenge here is that users need to understand pricing models and application requirements to choose the most appropriate cost model.

A major issue is the lack of an efficient machine learning framework for big data to work with various data types (e.g. text and image) [SS17] and application domains (e.g. skin cancer and insurance policies). It is also unclear which specific ML algorithm should be used to obtain good results for a particular dataset. This is because no learning paradigm can consistently perform better than all other paradigms for all problem situations [RSG$^+$14]. The challenge is how to choose an algorithm which achieves excellent results in the classification of data.

The extraction and representation of data features (attributes) are the basis for machine learning paradigms, and is, therefore, a critical step, especially for unstructured data. Various features can be extracted from data based on the data type. When a dataset has too many features, it is not ideal to include all of them in a machine learning model. Too many features raise significant computational difficulties and may lead to poor prediction accuracy as well as poor performance. Therefore, it is a complicated process to find the best subset of features from a wide range of features to achieve optimal learning performance and data accuracy.

1.4 Our approach

A comprehensive, generic cloud-based framework for supporting machine learning over big data is proposed. The underlying framework provides generic stages for segmentation, feature extraction, model building and classification. The architecture is technology invariant in that various technologies (e.g. Cloud software platforms) can be used to implement it in different ways. In each of the stages of the framework, several different software solutions can be integrated to enhance the framework's functionality such as using advanced machine learning algorithms, employing various feature extraction techniques, and leveraging distributed big data processing frameworks. In the proposed architecture, several optimisation components are implemented on top of the underlying framework to support optimising the performance, classification accuracy, use of resources, and cost. These optimisations are independent of the data type (e.g. text and image) and the application (e.g. skin disease diagnosis and face detection).

This thesis will address the following research questions:

- **Research Question 1:** Can a generic machine learning framework be developed, covering all the stages of data pre-processing, model building and classification, which can be used in various application domains, and which can support different types of unstructured data, and different machine learning algorithms?

  - **Our solution to Research Question 1** is to design a cloud-based framework that incorporates standard stages of machine learning (such as segmentation, feature extraction, and model building), that supports training and testing modes, and that can be used for different data types (e.g. text and image) and different machine learning algorithms (e.g. Naive Bayes and K-nearest neighbour). The framework is implemented using an asynchronous architecture incorporating queues, based on standard message formats for tasks and results. The generic architecture supports different types of data (e.g. text and images) and can accommodate parallel processing (segmentation, feature extraction, and machine learning) for a dataset of images and a dataset of text.

- **Research Question 2:** Can the proposed machine learning framework incorporate generic optimisations that optimise feature selection and ML algorithm selection, so that they adjust for the application (e.g. pathology or phytology), and the specific dataset (e.g. a particular set of blood cell images or leaf images)?

  - **Our solution to Research Question 2** is to develop various optimisation components built around techniques such as parallel evaluation, and a sample testing, evaluation, and feedback loop. The components provide optimisations to select the most useful features for classification and to select the best algorithm under user-specified criteria (e.g. most accurate, best performance, some combination of accuracy and performance). Training a model can be very demanding on resources, especially when there are a large number of features. The feature optimisation identifies the most useful features, and then only these features need to be extracted and used in building the ML model. Different algorithms provide different levels of accuracy and performance, and algorithm optimisation selects the algorithm (for a particular application and dataset) which best satisfies the user requirements with regard to accuracy and performance. The generic mechanisms used in these components, based on the sampling of the dataset, provide information without processing large amounts of data and result in optimisations that are data-driven and auto-adjust for the particular dataset being evaluated.

- **Research Question 3:** Can the proposed cloud-based framework optimise its use of resources in the cloud, and can it also incorporate a cost model, allowing optimisation with regard to both efficiency and cost of cloud resources?
  - **Our solution to Research Question 3** is to incorporate a resource optimisation component that monitors the use of resources, and then uses scaling to leverage cloud computing resources more efficiently. We also incorporate, another component with a cost model of commercial cloud offerings which optimises with regard to cost as well as efficiency. We show that there can be big processing differences between different datasets and between different algorithms on the same dataset, and so these optimisations auto-adjust for the data and the algorithms. The monitoring can indicate bottlenecks and allow a balanced, more efficient architecture to be achieved through appropriate scaling of resources. The cost-model of cloud resource offerings supports optimisation within an individual service provider (e.g. Amazon) or across service providers.

1.5 Publications

Important aspects of the thesis have been presented at distinguished peer-reviewed international conferences and published in the associated proceedings.

The basic framework and proof of concept feature-category optimisation are reported in:

- Rezvan Pakdel and John Herbert, *A cloud-based data analysis framework for object recognition,* in Proceeding of the 5th International Conference on Cloud Computing and Services Science (CLOSER 2015) [PH15].

Detailed feature optimisation and ML algorithm optimisation are reported in:

- Rezvan Pakdel and John Herbert, *Efficient Cloud-Based Framework for Big Data Classification,* the 2nd IEEE International Conference on Big Data Computing Service and Applications (BigDataService 2016) [PH16a].

Optimisation in the use of cloud computing resources, and demonstration of the same framework supporting text and image are described in:

- Rezvan Pakdel and John Herbert, *Scalable Cloud-Based Analysis Framework for Medical Big-Data*, the IEEE 40th International Computer Software and Applications Conference 2016 (Compsac 2016) [PH16b].

An optimisation component that incorporates a cost and resources model of cloud computing instances, and allows a cost-efficient solution to be implemented for a particular application and particular dataset is described in:

- Rezvan Pakdel and John Herbert, *Adaptive Cost Efficient Framework for Cloud-Based Machine Learning*, the IEEE 41st International Computer Software and Applications Conference 2017 (Compsac 2017) [PH17].

1.6 Thesis structure

In this chapter, we have discussed the importance of big data applications and the challenges and current trends in big data analysis. We have big data features, choice of appropriate features (the necessary information characterising the problem), choice of the learning algorithm, and use of computation resources. The remaining chapters are organised as follows.

In Chapter 2, we investigate several essential challenges of big data analysis. We explore a variety of data analysis techniques for big data processing and the taxonomy of the existing big data computing tools, frameworks, and platforms. We also present an overview of existing cloud platforms, different types of data learning and cloud services that data scientists can use for machine learning.

In Chapter 3, we propose a generic cloud-based machine learning framework for data classification, which can work with large volumes of heterogeneous data. The framework has an initial stage that extracts different features from a dataset and then creates models using a machine learning algorithm. We propose a feature-category optimisation component that evaluates models built with different category combinations. It then finds the best model for a specific dataset according to the model accuracy.

In Chapter 4, we update the framework and integrate two additional general optimisation components to select the best features and select the best classifiers for a specific dataset. The updated framework supports multiple data types (e.g. image and text). The optimisation components enhance the accuracy and performance of the classification task using a sampling, evaluation and feedback method.

In Chapter 5, we introduce an auto-scaling optimisation component into our cloud-based framework. Auto-scaling enables the framework to scale cloud computing resources automatically based on the dynamic processing load in order to improve the efficiency of big data processing.

In Chapter 6, we introduce a new optimisation component that uses sampling and feedback mechanisms to choose the appropriate instance type for a job in the application. Performance metrics are collected for a sample of a dataset and a matching algorithm maps these metrics for a job to the best choice of instances from the public cloud (in this case, AWS). Finally, the framework recommends appropriate instance types for a job in the application with respect to cost efficiency.

In Chapter 7, we present a summary of the thesis with discussion and conclusions.

Chapter 2

Background

The world's data is growing faster than ever before. Every human being on the planet is expected to produce 1.7 megabytes of data every second by 2020 [BIG18]. This data is a valuable resource, and we need intelligent data analysis techniques and scalable architectures to obtain useful information quickly. Cloud computing offers a suitable infrastructure of unlimited elastic storage on demand and cost-effective computation capacity. Data analysis involves the collection, storage, and processing of data. Processing big data requires ample and efficient storage facilities together with high-performance processors to get results in acceptable times. In this chapter, the existing work in the area of big data analysis and associated technologies (e.g. cloud computing) are reviewed.

2.1 Big Data

> "Big Data is the data that is too big, too fast or too hard to process" [Mad12].

The term 'too big' means that amount of data might be large scale (petabyte) and come from various sources; 'too fast' means that the data requires to be processed quickly; 'too hard' means that a processing tool processes the data with difficulty [Mad12].

Big data is data that size (Volume) is beyond the ability of traditional database software tools to capture, store, and manage; the rate of production (Velocity) exceeds the speed of current systems to process; complexity in mixed type and format (Variety) challenges existing algorithms to perform efficiently and effectively. These three dimensions of Volume, Velocity, and Variety together characterise big data. Figure 2.1 shows three characteristics of big data [JH15].

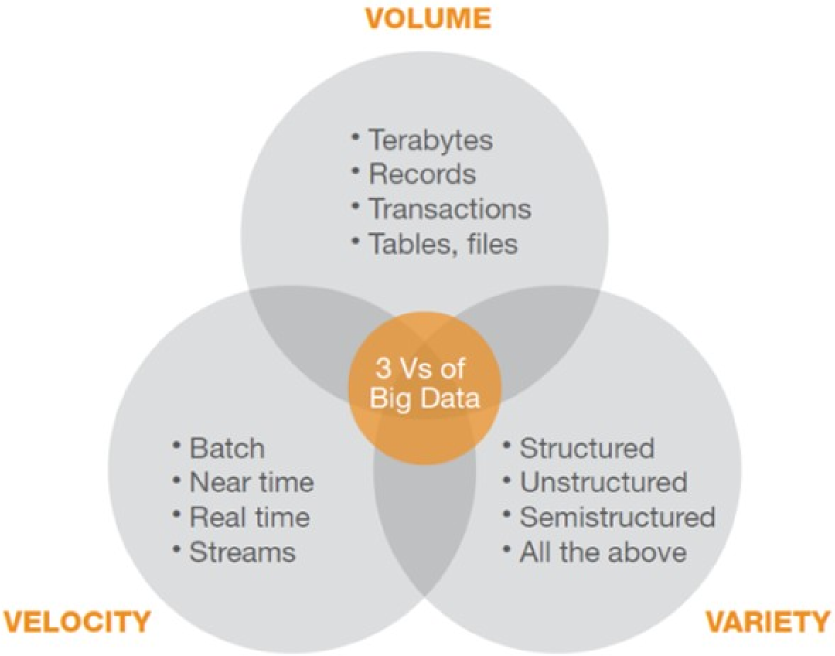
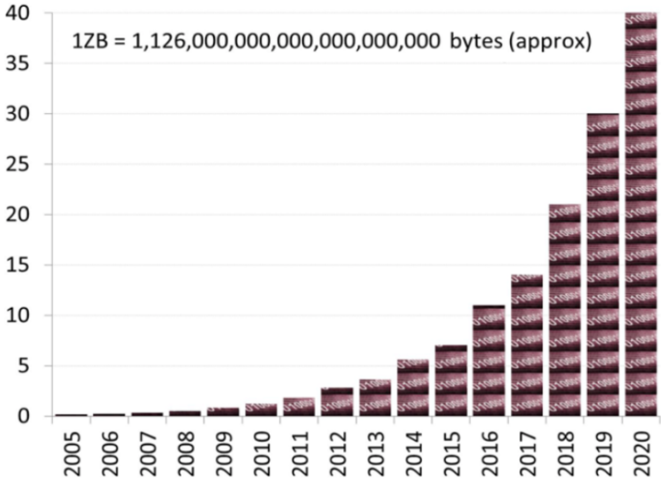


Figure 2.1: Big Data characteristics

- **Data Volume:** The volume describes the size of the data. According to an IDC (International Data Corporation) [Ale17] report, the volume of unstructured data will reach 40 zettabytes by 2020 (Figure 2.2).



Source: https://audiotech.com/trends-magazine/harnessing-big-data-opportunity/

Figure 2.2: Data volume growth by year in zettabytes

- **Data Velocity:** The velocity is the speed at which data comes from diverse sources such as sensors, business processes, and interaction between people and things like social media, websites, and mobile devices.

- **Data Variety:** Data can be in different forms, such as text, image, video, and audio. This data variety is claimed to be the key issue in big data analytics [SS17].
- **Data Veracity:** Veracity relates to inconsistencies and information uncertainty. Regulate quality and precision in these data, is hard.
- **Data Value:** Data in itself is of no use or significance, but in order to extract the information„ it requires to be transformed into something precious.

Data can be either structured, semi-structured, or unstructured [SKIW17]. Processing big data using traditional databases and traditional data processing applications is challenging. Scientists and companies need new techniques to store and access a large amount of data in a fast, secure and cost-effective way. Also, they need new efficient parallel techniques for fast processing and analysis of data. For example, the search service provider (Google) has to cache hundreds of millions of web pages and respond to search queries immediately, requiring it to develop novel techniques [BP98].

2.1.1 Data types

There are two main types of data: structured and unstructured - Figure 2.3 elaborates them, along with examples.

- **Structured data:** Structured data can be stored in traditional database systems such as Oracle and SQL server in the form of rows and columns. Structured data have a relational key and can be easily mapped into predesigned fields.
- **Unstructured data:** Unstructured data may have some internal structure but does not conform to a database schema. Unstructured data cannot be stored in the form of rows and columns. Generally, unstructured data varies in size and content and is difficult to analyse. The primary challenge of unstructured data is that they are difficult for non-technical business users to understand and analytically use.

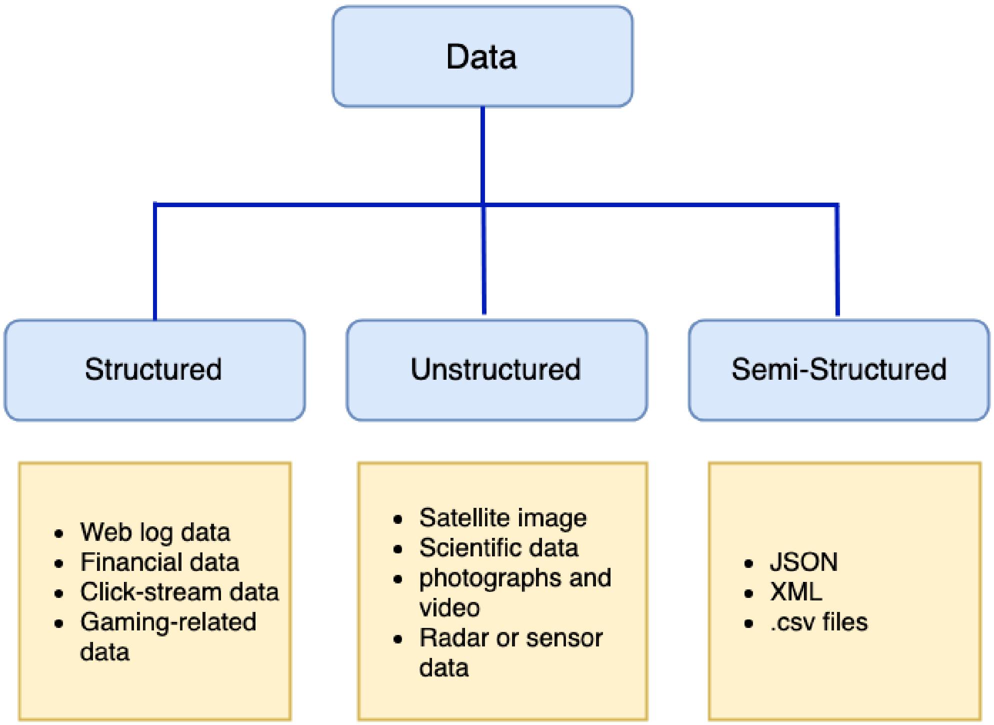


Figure 2.3: Big Data type

- **semi-structured data:** There is also a derived data type, semi-structured, that does not have the same organizational level and structured information predictability. This datatype (such as JSON, XML, and.csv files) may contain rational record-based information, but information may not be structured in a recognizable structure [NLC15].

2.1.2 Big Data applications

Big data benefits many areas, such as business, science [Yu14], goverment sector [KTC14], and smart farming [WGVB17]. In this section, the importance of big data and the challenges for data-intensive applications are reviewed.

2.1.2.1 Healthcare

Big data in healthcare involves health-related digital datasets. Processing and managing this data is complicated [GPP18]. Big data in healthcare offers numerous benefits such as disease detection and prediction [CHH^{+}17, EDM19], reduced medical errors, the right care at the right time, and better medical outcomes. Big data has great potential to improve medicine and guide clinicians in delivering value-based care.

- **Biotechnology and drug discovery:** Biotechnology companies and pharmaceutical companies use gene sequencing to develop more effective drugs for diseases. Gene sequencing involves an extensive collection of data, and biologists have to clean the sequence data and then translate the raw data into a suitable form [Lef15]. Biologists use sequence data for millions of humans and perform data mining to look for genome patterns that identify particular diseases. Processing this data with traditional technologies is impossible. Hence, new computational technologies are required to reduce processing time.

2.1.2.2 Government

A large variety of data can be created from different sources such as imagery, sensors, social media, video, audio, and cell phones. The government can use the data to improve national security and prevent crimes [KTC14]. Big data analytics can improve efficiency and effectiveness across a broad range of government services, by improving existing processes and operations and enabling entirely new ones.

2.1.2.3 Websites analytics

Millions of unique visitors add a broad range of content to websites per day. This can quickly amount to tens of hundreds of gigabytes per day (tens of terabytes per year) worldwide of accumulated user data. Companies are interested to understand this data and analysis the data [KS18, RMV$^+$18], improve their website loading and response time, and also offer more targeted ads [XLL13].

2.1.3 Big Data processing

Nowadays, big data is everywhere, and it is important to know how to handle the data to extract more useful information. Data analysis uses analytical methods to extract value. In many fields, data is analysed using specific analytical methods. For processing big data, a system with large computing resources is needed to get the results quickly and without exceeding the budget. This system should use subsystems such as decision making, data aggregation, storage, analysis, provisioning, and scheduling to efficiently process the data. One technique to reduce the computing cost of processing big data is to use a data segmentation technique [HRY16].

2.1.3.1 Data segmentation

Data segmentation is an essential technique in data preprocessing for ML [SM17]. Segmentation is beneficial for many applications (e.g. medical imaging and locating objects in satellite images). Different techniques like Threshold Method, Edge-Based Segmentation Method, Region-Based Segmentation Method, Clustering Based Segmentation Method, Watershed Based Methods, Partial Differential Equation Based Segmentation Method, Artificial Neural Network Based Segmentation Method [DY14] can be used for segmentation the data. For example, one of these techniques or a mixture of them can be used in medical image analysis to detect cancer [MASL18] or to detect roads and bridges in satellite images [Liz14].

Processing data can be mainly performed in two ways: Batch Processing and Real-Time Processing.

2.1.3.2 Batch processing

In batch processing, the data is collected, stored, and later processed. In this processing model, the aggregated data is processed at once [MGS12], and usually, during off-peak hours. The primary performance measure of batch processing is the amount of work done per unit of time and fast response time is not essential. Data processing in this model can take minutes, hours, or even days to execute.

2.1.3.3 Real-time processing

In real-time processing, data must be processed as it arrives, and the results should be provided quickly. The stream of input data produces an output stream of results. Good examples of real-time data processing systems are bank ATMs, traffic control systems, e-commerce order processing, online booking and reservations, and credit card real-time fraud detection. In this processing model, it is crucial that the deadline is met, and therefore, the data has to be processed within a specified time.

In general, the real-time processing model is suitable for applications that require the results quickly and batch processing is for general purpose applications. The differences between these two processing models are shown in Table 2.1.

Table 2.1: Batch processing vs. Real-time processing

	Real-time Processing	**Batch Processing**
Input	stream of new data or updates	data chunks
Data Size	infinite or unknown	known and finite
Storage	not store	store
Hardware	typical single limited amount of memory	multiple CPUs, memories
Processing	a single or few pass over data	processed in multiple rounds
Time	a few second	much longer
Applications	web mining, sensor network. traffic motoring	widely adopted in almost every domain

2.1.4 Existing big data analysis frameworks

Big data can be analysed using several techniques that perform complicated tasks in parallel to increase the processing speed and improve data analysis. Several distributed computing frameworks [PIP16] capable of processing big data in real-time or near-real-time are discussed.

2.1.4.1 Apache Hadoop

Apache Hadoop is a reliable, scalable, and distributed open source computing platform. The Hadoop framework can be used to process big data in parallel. Since its release in 2007, Hadoop has been adopted as a solution for scalable processing of large data in many applications including machine learning related applications [Gho11], graph processing, and behavioural simulations [Wan10]. Modern versions of Hadoop are composed of several components or layers that work together to process batch data. Hadoop Distributed File System (HDFS) is a distributed file system layer that coordinates storage and replication across the cluster nodes. HDFS ensures that data remains available despite inevitable host failures. It is used as the source of data, to store intermediate processing results, and to persist the final calculated results. Hadoop has a native batch processing engine named MapReduce. MapReduce is a programming model that is suitable for big data processing. MapReduce uses multiple machines in parallel in the cluster to perform large-scale data analysis.

2.1.4.2 Apache Spark

Apache Spark is a fast and general engine to process large scale data in parallel. It can handle both batch and real-time data processing workloads [?]. It includes Spark SQL, Spark Streaming, MLib (Machine Learning) and GraphX (parallel graph calculation). Spark can run on the Hadoop YARN cluster manager and can read any existing Hadoop data. Spark is used in various areas such as big spatial data processing [SYWJ17], text stream data analysis (from services such as twitter) [Hak18], and stream application logs [EVA18].

2.1.4.3 Apache Storm

Apache Storm is a stream processing system [Eva15]. The data can be processed quickly by Storm in real-time. Apache Storm is highly scalable, easy to use, and offers low latency with guaranteed data processing. It has a simple architecture to build applications called Topologies. It enables the developer to develop their logic virtually in any programming language and supports communication over a JSON-based (JavaScript Object Notation-based) protocol [BV16].

2.1.4.4 Apache S4

Apache S4 (Simple Scalable Streaming System) is a platform for processing continuous data streams. It is designed specifically for managing data streams. S4 applications are designed to combine streams and processing elements in real time.

2.1.5 Big data challenges

Several widely-used and useful technologies and data analysis techniques have been studied. Many existing data analysis frameworks and their strengths, weaknesses, and challenges have been reviewed. In this section, the most appropriate techniques, technologies, and frameworks that contribute to this study will be discussed.

Parallel processing is one of the cost-effective methods to solve computationally significant and data-intensive problems quickly [Cza14]. Efficient parallel processing frameworks are crucial for addressing large data performance and scalability requirements. As cloud computing offers multiple resources (e.g. virtual machines) and allows on-demand resource allocations [ZYF+18], it is suitable for big data applications. Cloud computing can be useful in hardware cost reduction and processing cost reduction for big data processing implementation. Using cloud computing can provide fast reactive provisioning for big data as provisioning servers in the cloud is secure. This fast provisioning is essential for big data as the amount of data can change rapidly over time. Besides, the resources can be scaled based on the processing requirements of the big data [CCD+12]. Cloud computing makes the environment more robust, automated and provides multi-tenancy (that allows customers to share computing resources). Data application resources can be better controlled, monitored and reported in the cloud computing environment. All of these advantages indicate that cloud computing can be considered as a suitable platform for deploying big data applications.

2.2 Machine Learning

Machine learning algorithms examine large quantities of data to identify patterns. In supervised learning, algorithms use a labelled dataset to create a model that enables forecasting. The objective in supervised learning is to predict the label of a new unseen data item, and its success depends on the choice of features for the data. The extraction and representation of features is, therefore, a crucial step for unstructured data processing. Machine learning requires some classifiers to use some data features and attributes, which is extracted from data to represent the data content, discover the patterns from data and learn them to classify new unseen data. ML invloves data mining, learning, and classification.

2.2.1 Data mining

Data mining is the process of finding related information from a collection of data. Data mining focuses on unknown properties being discovered in the data. Data mining methods can be used to improve learner accuracy. Big data mining is currently one of the most critical emerging areas of research [Yu14]. Traditional data mining techniques may not work well for big data due to the size, high dimensionality, heterogeneity and distribution of the data. The analysis of large data volumes requires efficient big data mining techniques and substantial computing resources.

Mining of particular information related to a concept is done based on the feature of the data. The obtaining of representational features, for data can be called feature extraction. Extracting a set of discriminatory features (attributes) from data (more importantly from unstructured data) provides the basis for machine learning which is used to discover useful hidden patterns [AOO14].

The discovered patterns support decision making and prediction. Different type of feature extraction methods introduced.

2.2.2 Unstructured data analysis

Unstructured data are those that have no predetermined form or structure. Unstructured data often contain text and multimedia content [Ing12]. For example, e-mail messages, videos, photos, word processing documents, audio files, web pages, and several other kinds of documents are categorised as unstructured data. In many fields, the amount of unstructured data grows significantly faster than structured data [SRMG18]. Analysing them are more complicated. Hence, unstructured data analysis is required to reform it into structured data, then proceed with the standard analysing frameworks for structured data. A common method of analysing unstructured data is first, to find a set of distinctive features that can help to distinguish between data; second, to build a machine learning model using a labelled training dataset; and finally, to use the model to classify new data. For instance, finding and identifying objects in an image is an essential task in computer vision. Humans can recognise objects irrespective of the point of view, size/scale, rotation or translation. Extracting some features from an image, then applying some machine learning classifier to the extracted features, is a method that is used in image processing [AOO14].

- **Image-based feature extraction:** Various approaches have been used to accurately detect objects in images, such as geometry-based, appearance-based and feature-based approaches. Feature extraction is the main element in most object recognition methods [NT16]. The image features can be divided into two groups, global and local. Local features are features extracted from the results of the subdivision of the image based on image segmentation or edge detection. On the other hand, global features are features extracted from the entire image or just a larger sub-area of an image [Gom15]. So, the global features describe the visual content of the entire image. Global features, such as shape and texture, are very useful. They create very compact representations of images where each image corresponds to a point in a high-dimensional space.

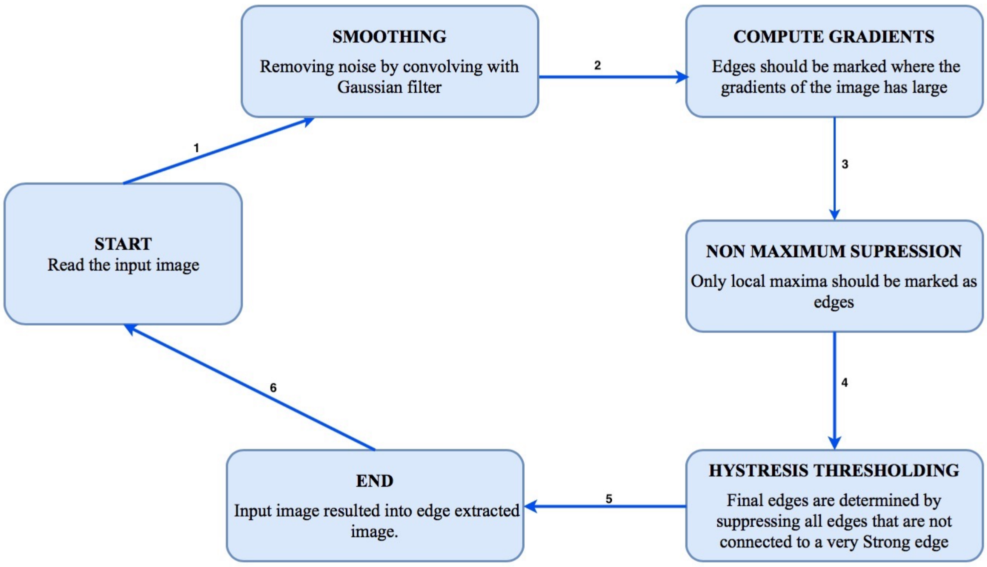

SMOOTHING
Removing noise by convolving with Gaussian filter
COMPUTE GRADIENTS
Edges should be marked where the gradients of the image has large
START
Read the input image
NON MAXIMUM SUPRESSION
Only local maxima should be marked as edges
END
Input image resulted into edge extracted image.
HYSTRESIS THRESHOLDING
Final edges are determined by suppressing all edges that are not connected to a very Strong edge
1
2
3
4
5
6

Figure 2.4: Canny edge detector algorithm [RKS13]

Several algorithms and methods have been proposed for extracting features from images. Harris corner detection is a method to detect and match point features like corners or edges [Mal11]. Canny edge detection, developed by John F. Canny in 1986, uses a multi-stage algorithm to detect a wide range of edges in images. The Canny edge detector uses a multi-stage algorithm to detect a wide range of edges in images (Figure 2.4). Region and contour detectors are also methods for object recognition. Detectors use image contours and region boundaries that should be less likely to be disrupted by clutter backgrounds near object boundaries. Region detectors are used for category recognition [KZB04] but are not practical for a large number of images representing different categories. Recognising an object can be done by extracting these features from an image. Using a combination of methods is claimed to be more useful in recognising objects in images [ASNG10].

- **Textual feature extraction:** There are many methods for extracting features from data in the form of text. The first step is to define what the features are. For example, for sentiment (polarity) classification

[Ans16, ZJ17], the features are mostly the terms or phrases that influence the sentiment of the text. Opinion adjectives, adverbs, nouns, verbs are useful features. Sometimes, the terms are not enough and bi-grams, tri-grams or, generally, n-grams should be considered. Some methods are described below for extracting features from the textual data.

- **Trimming vocabulary:**

  This method is used to remove “non-content” words, frequent “stop words” such as “the”, “and”, or rare words that occur <10 times in 100,000 words.

- **Stemming:**

  This method is used to reduce all variants of a word to a single term (see, saw, seen to “see”)

- **Define classes:**

  A user should specify how many classes he/she has. For example, the user may have different classes like Spam and Not Spam for email, or Sport, Science, and Politics for a newsfeed.

Much research has been conducted in the area of text mining in recent years. In [Jo10], various text mining techniques are used to discover textual patterns from online sources. Durga et al. [DG11], proposed a new model for text categorisation to capture the relations between words by using a WordNet ontology. This approach maps the words this representing the same concept into one dimension and provides better efficiency for text classification. The authors also identified best practices in the extracting for information based on semantic reasoning, and different advantages of intelligent information extraction were highlighted. The authors explained the suggested methods, such as query expansion and extraction of semantic-based document retrieval. Chakraborty et al. [Cha14] proposed a method to organise and analyse textual data for extracting insightful customer intelligence from an extensive collection of documents, and for using such information to improve business operations and performance. The authors describe the efficient classification of product review documents as a multi-label classification scenario using Structured Support Vector Machine [CJ15].

Chakraborty et al. [Cha14] proposed a method to organise and analyse textual data for extracting insightful customer intelligence from an extensive collection of documents, and for using such information to improve business operations and performance.

2.2.3 Learning

Machine Learning (ML) allows computers to learn without explicit programming [R10]. With the growth rate of big data, the challenge faced by the ML community is how to process and learn from big data efficiently. For efficient processing of big data, not only data processing but also implementations of ML algorithms should be considered. Learning from the extracted features can be used to describe the content of data for one of the followings: Classification, Ranking, Dimensionality reduction, and Clustering.

- **Classification:** The classification is used to assign a specific category to each item from a dataset. For example, given a document, assign it whichever domain (i.e. history, biology, and mathematics) it belongs.
- **Regression and time series analysis:** Regression is used to predict a real value for each item, such as future stock market values.
- **Ranking:** The ranking is used to return an ordered set of features based on some criteria defined by the user (e.g. web search).
- **Dimensionality reduction:** Dimensionality reduction (feature selection) is used to transform initial large feature spaces into a lower-dimensional representation to preserve the primary representation properties.
- **Clustering:** Clustering is used to group items based on a predefined distance measure. It is usually used on large datasets.

Machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve. They can be categorised into three groups: supervised, unsupervised, and semi-supervised learning.

- **Supervised learning:** The supervised learning method in ML tries to look for the mapping between inputs and outputs using already known results. A good instance is when trying to predict unknown answers based on labelled data that contains readily available answers. As Figure 2.5 shows, after the data has been pre-processed, the data is split into two random sets, training dataset and testing dataset. For example, 70% of the data is used for training a robust model, while 30% of the data is used to evaluate the model accuracy. The built model can be later used to recognise and classify new data [FF17].

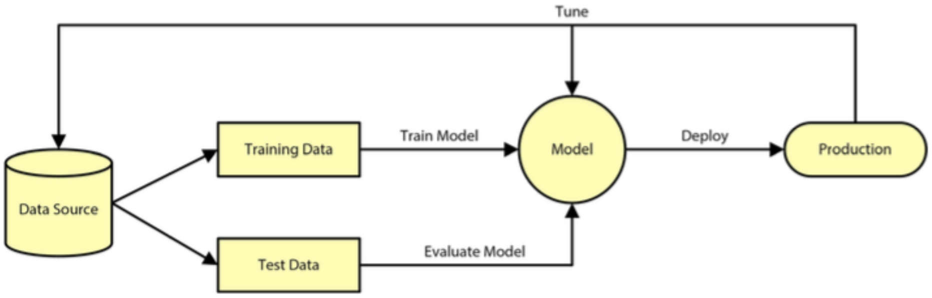


Fig. 1 Supervised machine learning workflow

Figure 2.5: Supervised machine learning flowchart

- **Unsupervised learning:** In unsupervised learning, developing ML solutions becomes much more difficult when compared to supervised learning because the ML algorithm is not provided with known input data and known outcomes. It has to analyse the data by itself and get common patterns and structures to develop a predictive model. This technique could be likened to density estimates in statistics, where there are attempts to find patterns in the input data. The distinct difference between these two learning approaches is that for supervised learning, marked up patterns are made available while for unsupervised learning marked up patterns are not made available [DZDW19].

- **Semi-supervised learning:** In this type, the input data is a mixture of labelled and unlabeled examples. Semi-supervised learning falls between unsupervised learning (without any labelled training data) and supervised learning (with completely labelled training data). Many machine learning researchers have found that unlabelled data can be used in conjunction with a small amount of labelled data to produce a considerably improved result with better accuracy [GFK$^+$13].

2.2.4 Classification

ML uses past information and experience to improve the performance or to make predictions [Moh12]. ML is applied in a wide range of applications such as text mining and document classification, keyword extraction, spam detection, emotion extraction, speech recognition, image segmentation, image recognition, fraud detection, medical diagnosis, and recommendation systems. In these applications, several types of learning problem can be identified. There is a variety of machine learning classification algorithms known as *classifiers*.

- **OneR:** OneR is a simple rule-based classifier that serves to extract a set of rules based on processing a single attribute. It is easy to produce consistent performance on various classification problems by probing only a single attribute.
- **Random Forest:** In Random Forest, there is a collection of decision trees (known as a "Forest"). For classifying a new object based on attributes, each tree gives a classification and the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).
- **C4.5 and J48:** C4.5 is an algorithm used to generate a decision tree. It constructs a massive tree by taking into account all attribute values and narrows down the decision rule with the help of pruning. J48 is an open source Java implementation of the C4.5 algorithm [KC14].
- **Naive Bayesian (NB):** Naive Bayesian (NB) classifier is a robust approach to supervised classification and has a high degree of accuracy. A probabilistic measure based on the probability of the class and the associated distribution of probability for each attribute of the training data is used to classify an unseen instance.
- **Support Vector Machine (SVM):** The support vector machine (SVM) is a type of generalized linear classifier that is widely applied for classification and regression problems. SVM is an algorithm that determines the best decision boundary between vectors that belong to a given group (or category) and vectors that do not belong to it.

2.2.5 Machine learning applications

Machine learning is useful in a diverse area of applications such as self-driving vehicles [FBG+17], health [SRB19], social media [KGL+18], food [LLW18], nature [MSR+18].

- **Healthcare:** Processing very large amounts of information collected about individual patients is not possible for a human. Hence, machine learning (ML) provides a way to find patterns from data and process them automatically. Machine learning can be applied to various types of healthcare data (structured and unstructured) to predict important diseases, including cancer, neurology and cardiology diseases [JJZ+17]. If such diseases are predicted well in advance, the information can provide essential insight to doctors who can then adapt their diagnosis and treatment on a per patient basis. For instance, in 2015, [AEAA15], an expert system for skin disease diagnosis was proposed which allows a user to identify diseases of the human skin and provides advice and medical treatments. In this work, the authors used image processing, feature extraction, and machine learning to recognise the disease. The authors, in [WGJ18], proposed a system to diagnose skin-related disease by employing a machine learning approach and using the skin colour feature and texture feature.

- **Food:** The aim of machine learning is to explore the search space to find the best solution to any problem. Machine learning models are currently being developed to address the complexity and variety of data in the food industry. Many companies and organisations (i.e. Tastry, Vinify, Edamam, Pingwell, Sure, and Instacart) already incorporate deep learning, machine learning, and AI into food and beverage product development and services. Gastrograph AI uses machine learning to understand consumer sensory perception of flavour and predict consumer preference for food and beverages. Food recognition is another interesting subject, mostly among scientists. For instance, Lu et al. [LLW18] use an SVM classifier to recognise the food category, resulting in a recognition rate of 94% for food replicas and 58% for real food items. Kong et al. [KT12] proposed the use of Scale Invariant Feature Transform (SIFT) features clustered in visual words and fed to a simple probabilistic Bayesian classifier, which matches food items in a food database containing images of junk food and different types of fruit and vegetables.

- **Nature:** Machine learning can be used for natural resources such as plants [SK18, MSR+18], water [ASHL18, WZD17], and air [XFZ+19, KGC+18]. For instance, Meena et al. [MSR+18] studied the discovery of an unhealthy region of plant leaves and the classification of plant leaf diseases. Statistical Gray-Level Co-Occurrence Matrix (GLCM) features are used to support the classification of leaves using a machine learning algorithm (support vector machine). Also, machine learning methods can be beneficial for the accurate automated binary classification of surface water in complex geographies. For example, Acharya et al. [ASHL18] used several machine learning methods to classify surface water from satellite images and, thus, to monitor and manage surface water in Nepal. In many industrial and urban areas, monitoring and preserving air quality has become one of the essential activities. The air quality is affected by various forms of pollution (e.g. transport, electricity, and fuel uses). Hence, accurate air quality monitoring models which process information about the concentration of air pollutants is vital. These models can be used to evaluate and predict the quality of air. For example, machine learning based techniques were used for air quality forecasting [KGC$^+$18]. This paper reviews the published research results relating to air quality evaluation using methods including decision trees, and deep learning.

2.2.6 Machine learning and data processing platforms

Several toolboxes and plugins have been implemented to utilise the current data processing platforms (that uses R, Python, SQL and SASprogram languages [KDn18]) with machine learning technologies. Since the introduction of the distributed data processing frameworks such as Hadoop, machine learning enabled platforms have also been developed. Petuum, Jubatus, MLlib (MLBase), Oryx, H2O, and Mahout are examples of them [PIP16]. Most of the frameworks rely on Hadoop's MapReduce paradigm. Their mechanisms are the same as the traditional distributed data processing, which is to split the data and process it in parallel. That requires the underlying computing platform to provide a cluster of computing nodes to enable processing using limited computing power. This, however, makes the system limited unless cloud computing technology is employed. Azure ML, Amazon ML, Google Prediction API, and EigenDog are examples of popular Machine Learning as a Service [PIP16].

2.3 Cloud computing

Cloud computing offers network access to a shared pool of on-demand configurable computing resources (networks, servers, storages, applications, and

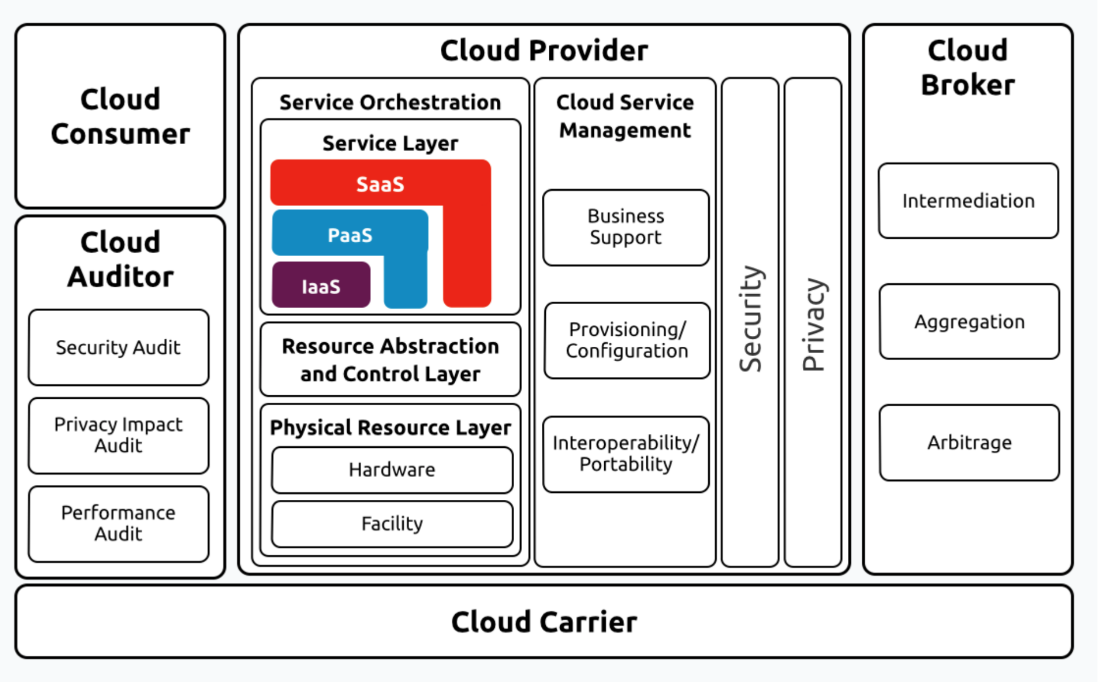


Figure 2.6: Cloud Architecture

services). The resources can be rapidly provisioned and released with minimal management effort or service provider interaction [Avr14]. The cloud is more than just a set of hardware, software, and services; it uses the combination and integration of comprehensive information technologies like distributed and parallel computing, network storage technologies, virtualisation, and load balancing to run an application [KYM$^+$18].

As shown in Figure 2.6[1], a cloud architecture can be viewed as five components (cloud provider, cloud consumer, cloud carrier, cloud auditor, and cloud broker) working together to provide on-demand services [HLS$^+$11].

- **Cloud provider:** The cloud provider is an organisation that provides cloud services, controls IT infrastructures, and manages technical failures.
- **Cloud consumer:** A cloud consumer (i.e. user) is a person or an organisation that uses cloud services through an agreement with a cloud provider or broker.
- **Cloud broker:** Cloud brokers are third-party companies that work closely with both cloud providers and cloud consumers. Generally, they provide consultation and trade various cloud solutions with their existing or new customers.

[1] `https://www.vyomtech.com`

- **Cloud carrier:** Cloud carriers are network and telecommunication companies that provide infrastructure so the cloud services are available to the cloud consumers.
- **Cloud auditor:** Cloud auditors are third-party specialists in an independent evaluation of cloud services provided by cloud providers. A cloud auditor can examine various areas such as security, privacy, performance, licensing, operation, and other areas in order to highlight differences between different operations and data privacy.

2.3.1 Cloud characteristics

Cloud computing has some essential characteristics which are identified by the U.S. National Institute of Standards and Technology (NIST) [PBA$^+$08], as follows:

- **On-demand capabilities:** On-demand capabilities mean that a user can change the usage of the services (i.e. storage networks and software) through an online control panel at any time. Typically, the user should pay monthly for what they used. The payment plans vary for each provider.
- **Elasticity:** Elasticity is the dynamic scaling of IT resources. It refers to the ability to react immediately to clients dropping or adding services in real time. This scaling must be done based on runtime conditions.
- **Broad network access:** Broad network access is the ability to use a device (smartphone, tablet, office computer, and laptop) to access business management solutions.
- **Resource pooling:** Resource pooling allows cloud providers to allocate and deallocate different physical and virtual resources dynamically according to consumer demand.
- **Measured service:** Measured service is that a user allows the cloud provider to measure storage levels, processing, and bandwidth of the services. The number of resources that may be used can be monitored and controlled from both the user side and the cloud provider side.

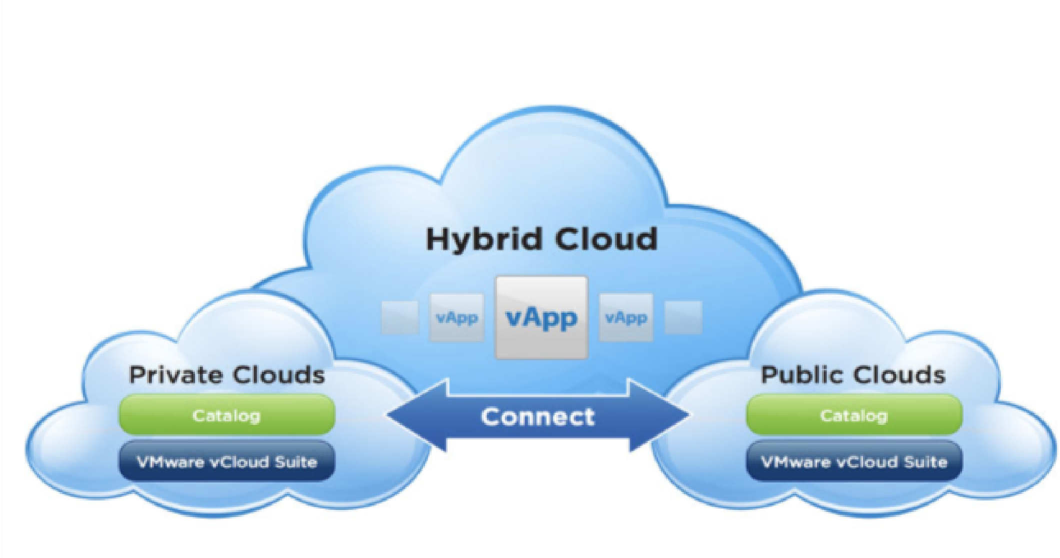


Figure 2.7: Three Types of Clouds

2.3.2 Cloud computing types

A key element of cloud computing is the deployment model. Deployment is the process of making software available to consumers. Three Cloud computing types (Public, Private, and Hybrid [Sim17]) shown in Figure 2.7 are reviewed in this section.

2.3.2.1 Public cloud

In the public cloud model, cloud services are available to all users over the Internet. A public cloud is usually free or is on a Pay-Per-Use model. It is perfect for small size and medium size enterprises.

2.3.2.2 Private cloud

A private cloud is hosted in the private data centre of an organisation and provides its services only to a selected group of users. A private cloud is more secure but expensive because the company is responsible for operating and maintaining the technologies. A vital benefit of the private cloud is the ability to customise the compute, storage, and networking components to better suit the requirements.

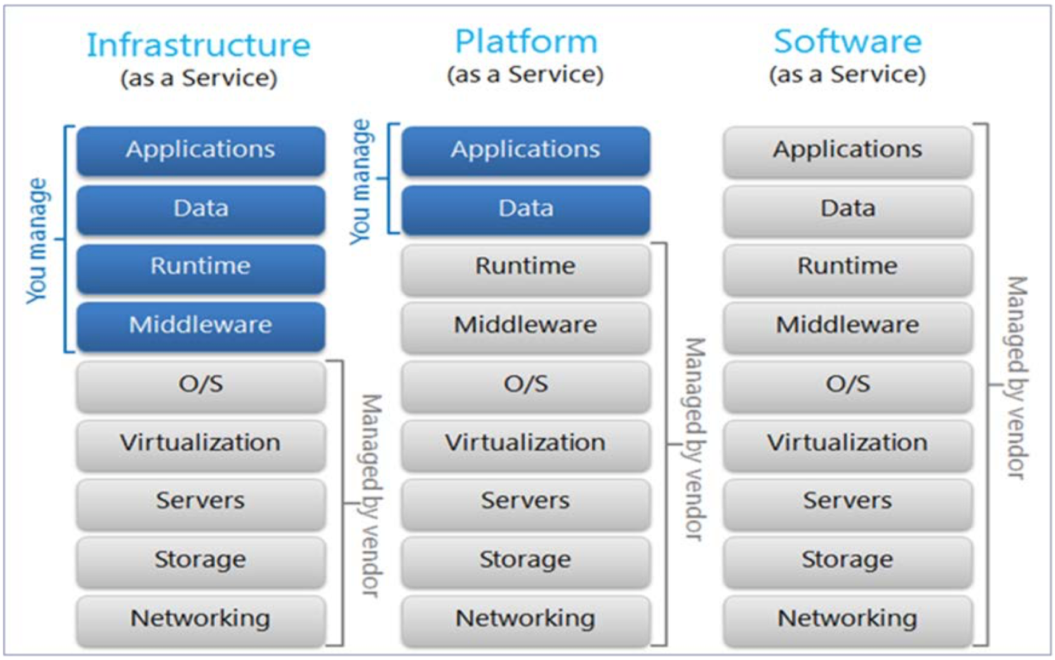


Figure 2.8: Three layers of cloud service models

2.3.2.3 Hybrid cloud

In the hybrid cloud model, an organisation manages some resources in-house and some out-house. For instance, a company has its human resource (HR) and customer relationship management (CRM) data in a public cloud (i.e. Salesforce) but have more sensitive data processing in a private cloud.

2.3.3 Cloud service models

There are various cloud services offered by several cloud providers (Figure 2.8). In general, they can be divided into three categories (Infrastructure as a service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS)) [Mal18].

2.3.3.1 Infrastructure as a Service (IaaS)

IaaS is the basis of cloud computing. IaaS provides virtual infrastructure as well as actual hardware for managing the storage, virtual machines, and virtual network over the Internet. A provider can create virtual servers with different configurations. Infrastructure resources can scale up/down based on demand growth. In the IaaS model, a client pays on a Pay-Per-Use basis for the use of equipment to support computing operations. An advantage of IaaS is the rapid increase or decrease of infrastructure on demand, lower risk in Return of Investment (ROI), reduction of human resource and hardware costs, and automated scaling of computing power. Examples of IaaS include Amazon EC2 [Ama18], RackSpace [Rac19], and Google Compute Engine [Goo19].

2.3.3.2 Platform as a Service (PaaS)

In PaaS, underlying cloud infrastructure such as operating systems, servers, and the network are hidden from customers. The cloud provider manages and delivers programming languages, frameworks, libraries, services, and tools for customers to create and deploy applications.

2.3.3.3 Software as a Service (SaaS)

SaaS is the most familiar form of cloud service for consumers. It provides a complete software solution on a pay-as-you-go basis. Salesforce, Google Apps, and storage solutions (Box Cisco WebEx, Microsoft Office 365, and Dropbox) are examples of SaaS.

2.3.4 Cloud providers

Amazon, Google, and Microsoft provide cloud platforms (e.g. Amazon's Elastic Compute Cloud, Google App Engine, and Microsoft's Windows Azure) for business applications [LKK+14]. They provide cost-effective, highly scalable, and flexible solutions for large-scale and data-intensive applications.

2.3.4.1 Amazon Web Services (AWS)

Amazon Web Services (AWS) is one of the most popular cloud computing platforms. It offers a wide range of storage and computing services. The key features of AWS are Comprehensive security capabilities, Rich controls and auditing, Hybrid IT architectures, Scalability, Web Applications, Big Data and High Performance Computing supports, Backup and storage, and Disaster recovery.

2.3.4.2 Microsoft Azure

Microsoft Azure (also known as Windows Azure) is a public cloud that provides the resources and tools to users for building, deploying, and managing cloud applications. It supports various programming languages and offers several other features such as Ease of use, Administrative tool, and PaaS.

2.3.4.3 Google Cloud Platform

Google compute engine is a scalable and high-performance virtual machine provider that supports large-scale workloads and offers several features such as Google infrastructure backup, Scalability, High-performance, Low cost, Fast and efficient networking, Environmental friendly global network, and Flexibility.

2.3.5 Cloud software platform

Various cloud platforms have been developed to manage cloud infrastructures. Aside from the comercialised platforms, there are several open-source platforms that have been widely used by researchers and scienticts. OpenStack, Eucalyptus, CloudStack and OpenNebula are well-known examples [VDWF12] described here.

2.3.5.1 OpenStack

OpenStack is a cloud computing software platform released in July 2010. OpenStack was initially designed by the National Aeronautics and Space Administration (NASA) and Rackspace [SAE12, Yad13]. A vast pool of computing, storage, and networking resources are managed through a web-based dashboard (i.e. Horizon) and via the OpenStack provided API. OpenStack is scalable and easy to use. OpenStack offers several components (shown in Figure 2.9).

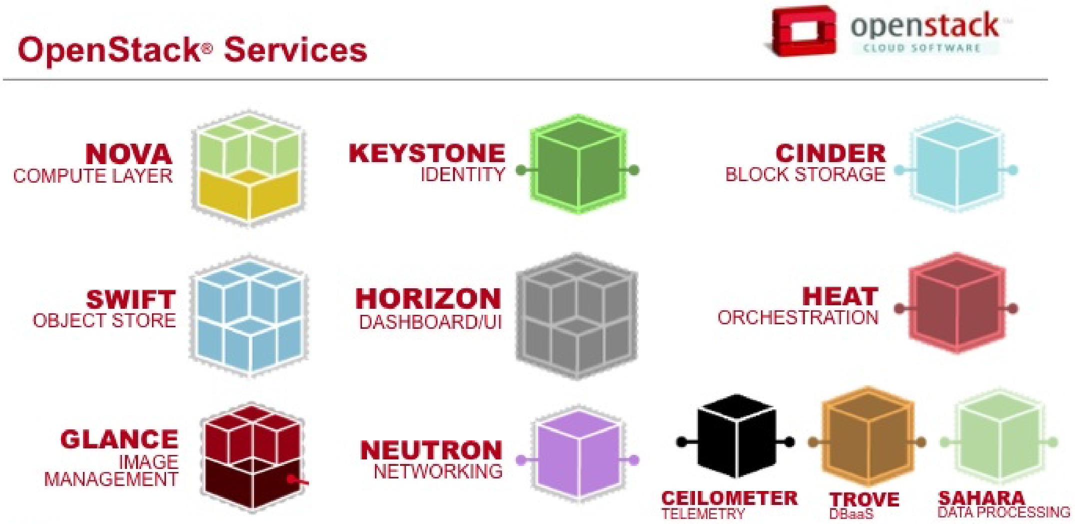


Figure 2.9: Components of OpenStack

- **Nova:** Nova provides essential components (shown in Figure 2.10) that support the management and delivery of computing resources. Nova controls the life-cycle of OpenStack instances.

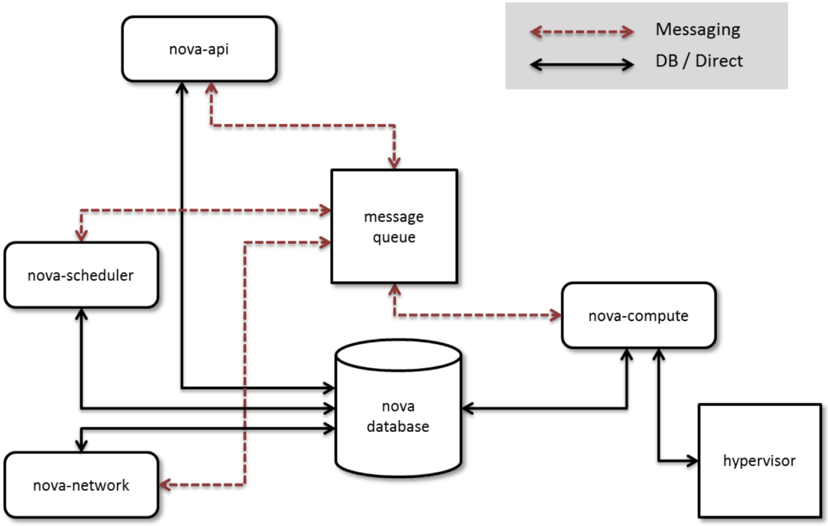


Figure 2.10: OpenStack Compute Architecture Overview

- **Swift service:** Swift is a storage system for objects and files. A file is given a unique identifier, and using it, other OpenStack components can access the file.

- **Glance:** Glance is a virtual machine image management service. It manages (i.e. creates, stores, and retrieves) virtual machine images by communication with Swift.

- **Neutron service:** Neutron is a network management service that maintains network connectivity between OpenStack components. It provides static and floating IP to support dynamic rerouting of the traffic during maintenance or in case of failure. It also manages firewalls and virtual private networks.
- **Keystone:** OpenStack has interactions among components and services and therefore, security matters. Keystone (identity service) is provided to manage the authentication and authorization of various services.
- **Database service:** The Database Service provides a reliable and scalable cloud Database-as-a-Service functionality for relational as well as for non-relational database engines.
- **Orchestration (Heat):** Orchestration is a template-based way to describe cloud application instances, networking information, volumes, security groups, and even users. Heat is an example of Orchestration that allows users to describe deployments of complex cloud applications in a template (text file). The template then is parsed and executed by the Heat engine.
- **Telemetry (Ceilometer):** Telemetry is the process of reliably collecting data on the utilisation of the physical and virtual resources in the OpenStack environment. It supports use cases such as metering, monitoring, and setting alarms. For instance, Ceilometer efficiently collects, normalises, and transforms data (e.g. resource usage) produced by OpenStack services. For example, this data can be used for generating an alarm when usage exceeds the threshold limit.

In summary, OpenStack provides an easy-to-use web-based dashboard where cloud services can be monitored, robust role-based access controls, and multiple forms of authentication (i.e. user name/password and token-based authentication).

2.3.5.2 Eucalyptus

EUCALYPTUS stands for Elastic Utility Computing Architecture for Linking a Program To Useful Systems. It is an open-source software framework that provides the platform for private cloud computing implementation on computer clusters [DGG15]. Eucalyptus components are shown in Figure 2.11.

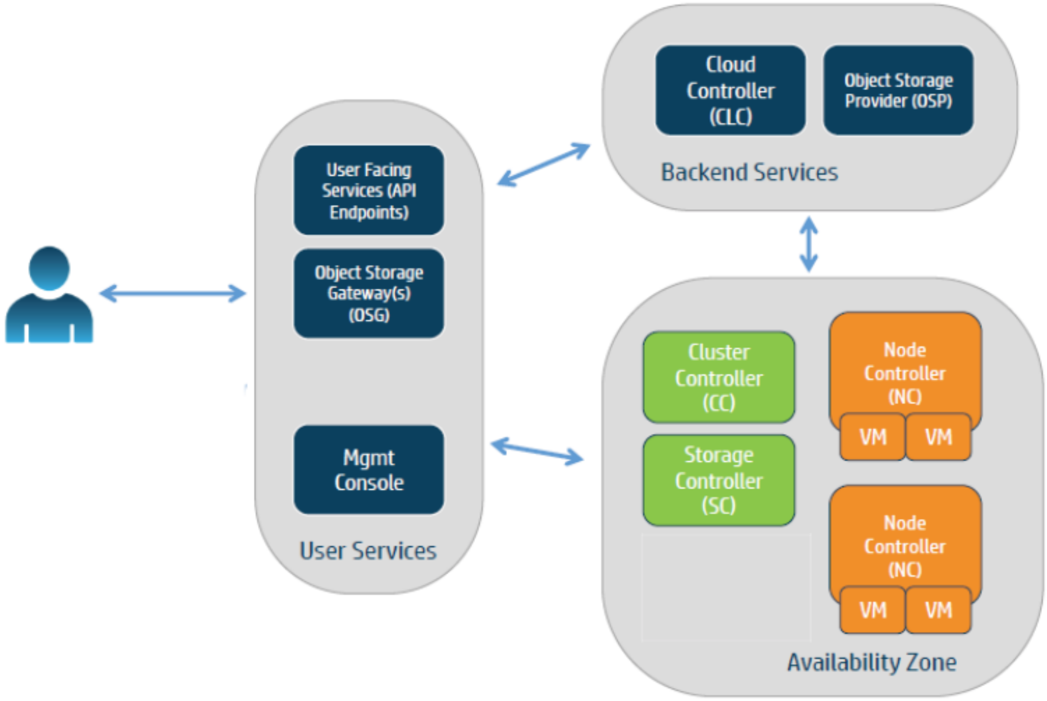


Figure 2.11: Components of Eucalyptus

- **Node Controller (NC):** The function of NC is the execution of the query to collect necessary information, controlling and providing the collected information to the Cluster Controller (CC).
- **Cluster Controller (CC):** It is a front-end component that uses the information which is provided by NC to manage execution.
- **Cloud Controller (CLC):** Cloud Controller is a significant component of the Eucalyptus. It is an interface between user and cloud, and a user can use it to access the cloud. Its main task is network resource management, scheduling, and authentication.
- **Storage Controller (SC):** Storage Controller is used for storing data. It is a data storage device that stores and accesses virtual machine images.

Eucalyptus mainly benefits the processes of service deployment and management, also supports administrators to easily install, maintain, and administer the cloud.

2.3.5.3 CloudStack

CloudStack is an IaaS cloud platform which is used to deploy and manage large networks of virtual machines [MS16]. It composes the management server and cloud infrastructure. CloudStack delivers services quickly and helps to reduce IT operations costs. A typical CloudStack infrastructure is organised, as shown in Figure 2.12. It consists of several parts (zone, pod, cluster, hosts, primary storage, and secondary storage).

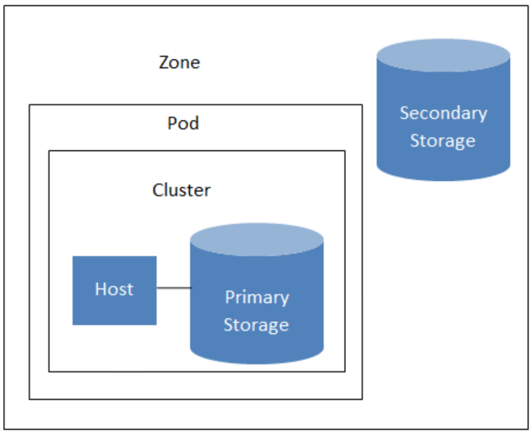


Figure 2.12: CloudStack infrastructure

- **Zone:** The zone contains a single data centre consisting of pods and secondary storage.
- **Pod:** The pod contains one rack of hardware consisting of layer-2 switch and cluster(s).
- **Cluster:** The cluster contains one or more hosts and primary storage.
- **Hosts:** The hosts consist of a single compute node, within which cloud services run.
- **Primary storage:** The primary storage stores the disk volumes for all VMs running on hosts in the cluster.
- **Secondary storage:** The secondary storage stores templates, ISO images, and disk volume snapshots.

2.3.5.4 OpenNebula

OpenNebula is an open source cloud computing tool. OpenNebula can manage the heterogeneity and complexity of distributed extensive infrastructure [IMCD16]. OpenNebula can be used to build a cloud computing environment that is highly scalable. Moreover, the public cloud is fully supported by the provision of interfaces and vital functions to VM, network, and storage management. Users can access services via control interfaces provided by the platform. The platform consists of three main components (core virtual infrastructure manager, capacity manager, and drivers).

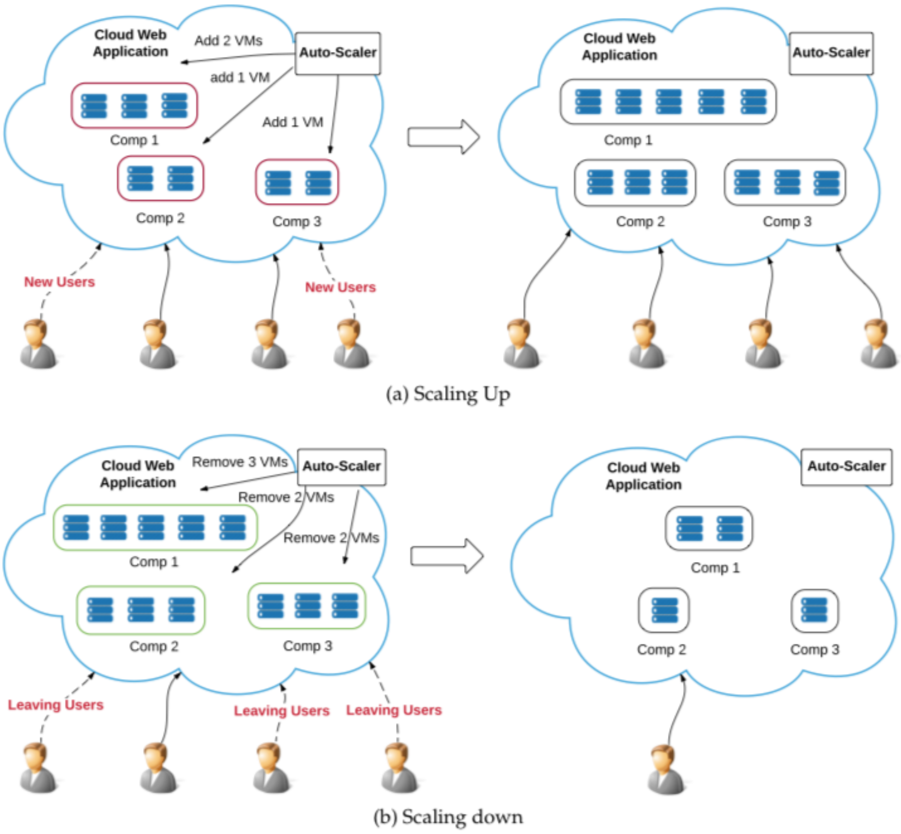
- **Core virtual infrastructure manager:** The core virtual infrastructure manager manages the life-cycle of virtual machines by ensuring a smooth running of basic operations, including migration, monitoring, and deployment.
- **Capacity manager (scheduler):** The scheduler module is responsible for managing all functionality from the OpenNebula core, such as workload balancing.
- **Drivers:** Drivers provide the necessary drivers to regulate data transferring and VM management irrespective of the hypervisors.

OpenNebula does not provide a Graphial User Interface (GUI), and this is the main drawback to the platform. OpenNebula, however, offers server consolidation, dynamic partitioning clustering, centralised management, load balancing, heterogeneous workloads, and virtual machine supply on demand.

2.3.6 Auto-scaling optimisation

Auto-scaling is a cloud computing optimisation feature. Cloud users use this feature to automatically scale up/down the processing resources (i.e. virtual machines and server capacities). Figure 2.13 shows two auto-scaling scenarios. Once needed, the auto-scaler can decide to allocate more resources to one application (Figure 2.13-a) and deallocate the resources (Figure 2.13-b).

Figure 2.14 shows various phases of auto-scaling of an application in a cloud environment. The phases are known as the MAPE (Monitoring, Analysis, Planning, and Execution) control loop[KC03].



(a) Scaling Up

(b) Scaling down

Figure 2.13: Auto-scaling resources

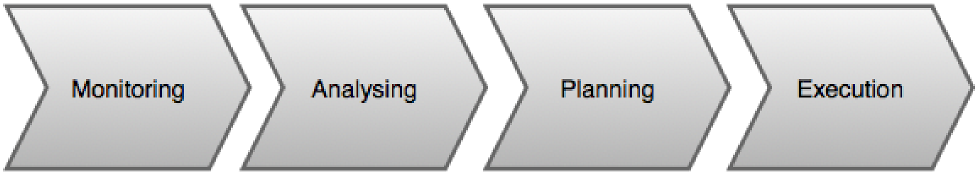


Figure 2.14: Different phases of the MAPE loop

- **Monitoring:** Monitoring is an essential phase in auto-scaling. It provides measurements used in decision making of scaling an application.
  - **Performance metrics:** There are many performance metrics regarding cloud services. The performance metrics include CPU usage, memory usage, execution time. Selection of the best performance indicators is essential to the success of an auto-scaler.
  - **Monitoring interval:** The monitoring interval determines the sensitivity of an auto-scaler. Choosing appropriate monitoring intervals cause to obtain balanced performance.
- **Analysing:** In the analysing phase, the system decides whether it needs

to perform scaling actions based on the monitored information.

  - **Scaling timing:** The system needs to decide when to perform the scaling actions.
  - **Workload prediction:** It is important accurately predict the future workload when proactively scaling chosen by the system.
  - **Adaptivity to changes:** The auto-scaler must be aware of any substantial changes in the application or workload and be ready for the new situation.

- **Planning:** The planning phase estimates the number of resources in total in the next scaling action that should be added/removed. The composition of resources should also be optimised to minimise financial costs.
  - **Resource estimation:** In the planning phase, the number of resources to handle the current or incoming workload is estimated.
- **Execution:** The execution phase is responsible for executing the scaling plan to add/remove resources. It is straightforward and can be implemented by calling the cloud provider's APIs. A challenging task here is supporting the APIs of different providers.

2.3.6.1 Taxonomy of auto-scaling

Figure 2.15 shows the taxonomy for auto-scaling applications. In this section, the auto-scaling sub-activities will be discussed in detail.

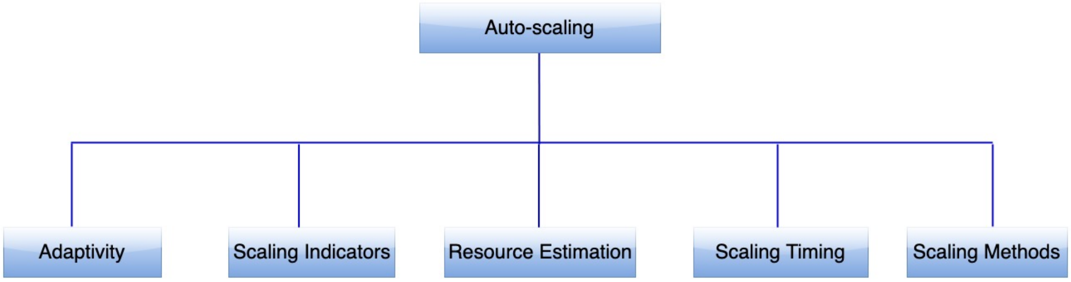


Figure 2.15: Taxonomy for auto-scaling applications in the cloud

- **Adaptivity:** One of the challenges is whether and how the auto-scaler adjusts itself to adapt to workload and application changes. As mentioned, the auto-scaler is responsible for adding/removing resources to the application to reach the target performance. In a dynamic cloud environment, the application, or workload can change at any time. Hence adaptivity is essential for the auto-scaler. For example, Amazon [Ama18] uses a rules-based approach to define rules, and the decisions are made based on the current input and these rules.

- **Scaling Indicators:** It is essential to know what metrics are monitored and measured to make scaling decisions. The auto-scaler is responsible for doing the right action based on the collected performance indicators of the application. These performance indicators can be collected and monitored at different levels of the system (low-level metrics and high-level metrics). The server information at the physical server/virtual machine layers such as the utilisation of CPU, memory, and network resources are considered low-level metrics. The information at the application layer such as resource rate, average response time, session creation rate, throughput, service time, and request mix is considered as high-level metrics. The auto-scaler can decide to scale up and scale down the resources, based on the metrics, to maintain the overall utilisation within a predefined upper and lower bound. For example, in work [SSCS10, KC14] , the information about the request service time and request mix is obtained to estimate how much resources needed. These metrics are not straightforward to measure. Service time is the time for processing the request, which is widely used in the queuing models to approximate the average response time. Kaur et al. [KC14] have used the past server logs to infer the mean service time. In another work, [GDK+14], the authors employed Kalman filters to estimate service time during runtime. In some work, [PHS+09, YF13, FdRL+14], both high-level and low-level metrics are monitored.

- **Resource Estimation:** Resource estimation identifies the minimum amount of computing resources required to process the workload to determine whether and how to perform scaling operations. Various attempts have been made to develop resource estimation models from basic approaches to methods with complex models. These can be categorised into six groups, namely rule-based, fuzzy inference, application profiling, analytical modelling, machine learning, and hybrid approaches. For example, in the rule-based method, a set of predefined rules (actions and conditions) can be defined [DTM12, HGGG12] such as: If CPU utilisation reaches 70%, add two instances or If CPU utilisation decreases below 40% remove one instance. In another fuzzy-based method, the core is a set of predefined "If-Else" rules, to make provision decisions [FLRC14]. In this model, the auto-scalers are commonly linked with machine learning techniques to automatically and dynamically learn the rule-set [LZ10, JPM16].

- **Scaling timing:** The critical question here is when auto-scalers should scale the application. However, there is no specific solution to this issue as different applications have various workload characteristics, and preference of cost and quality of service (QoS). Auto-scalers can be divided into two groups. In the first group, the application can be reactively scaled by auto-scalers based on the current status of the application and the workload. In the second group, the resources proactively add/remove from the application by considering the future needs of the application [KWQH17]. For applications with regular and smooth workload changes, reactive auto-scalers are usually preferred because they can save more resources without causing a large number of a service level agreement (SLA) violations.

- **Scaling methods:** Auto-scaling in a cloud environment can be done vertically, horizontally, or can be a hybrid.

  - **Vertical scaling:** Vertical scaling means removing or adding resources such as CPU, memory, I/O, and network to or from existing VMs. Vertical scaling is not suitable for highly scalable applications due to its limitations. However, some services or components that are difficult to replicate during runtime, such as a database server can benefit from by vertical scaling [YF12, SKZ$^+$14].

  - **Horizontal scaling:** The core of the elasticity feature of a cloud is horizontal scaling. It means adding more individual units of a resource doing the same job [SAAB$^+$18, CU16]. For example, in the case of servers, it could increase the performance by adding more servers as needed. Instead of one server, one can have two, ten, or more servers sharing the same work. Most cloud providers offer different size of VMs for users to choose. Some of them allow users to customise their VMs with a specific amount of cores, network bandwidth, and memory. Also, various pricing models exist in the current cloud market, which further increases the complexity of the provisioning problem.

– **Hybrid:** Mixing vertical scaling and horizontal scaling is called the hybrid model. This model is used in some work such as [HGGG12, YLS$^+$14] to scale the application. Optimisation techniques can be used to search for a scaling plan that results in the least cost using a hybrid of vertical and horizontal scaling [DGVV12, GDK$^+$14]. In the hybrid scaling model, both horizontal scaling and vertical scaling can be separately applied to different components of the application.

2.3.7 Cloud cost estimation

Cloud providers offer essential computational resources (such as storage and compute power) to users with the benefits of "on-demand" and "pay-per-use" features. Estimating the existing large variety of service offerings and choosing the best provider is a difficult task for a user. Hence, there are two critical challenges regarding choosing providers and understanding cost models for a user. The first challenge is understanding the cloud resources and their diverse pricing models. The second challenge is estimating the potential cost of obtaining the required services. The pricing model defines the associated cost of providing services on a per-use-basis. There are two types of pricing plan: static (fixed) pricing and dynamic pricing [SH17].

The static pricing model is pay-per-use. The cloud provider determines the prices of a variety of resources in advance. Figure 2.16 shows, as the demand increases, the static price does not change over time, and it causes profit loss [SH17].

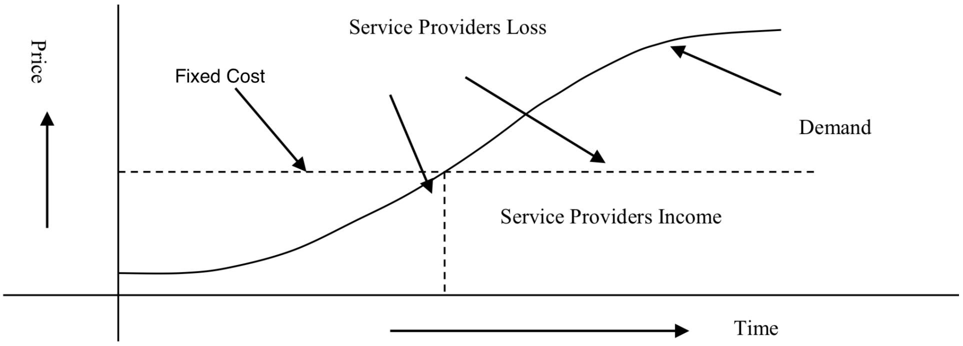


Figure 2.16: Static pricing limits service provider’s profit

In the dynamic pricing model, prices are changing dynamically based on the market conditions or status [SP11]. The price of the cloud services can be calculated automatically based on a real-time request.

Figure 2.17 shows the most important market players in different IaaS domains (i.e. Storage), and that the pricing models of each domain for each provider are different.

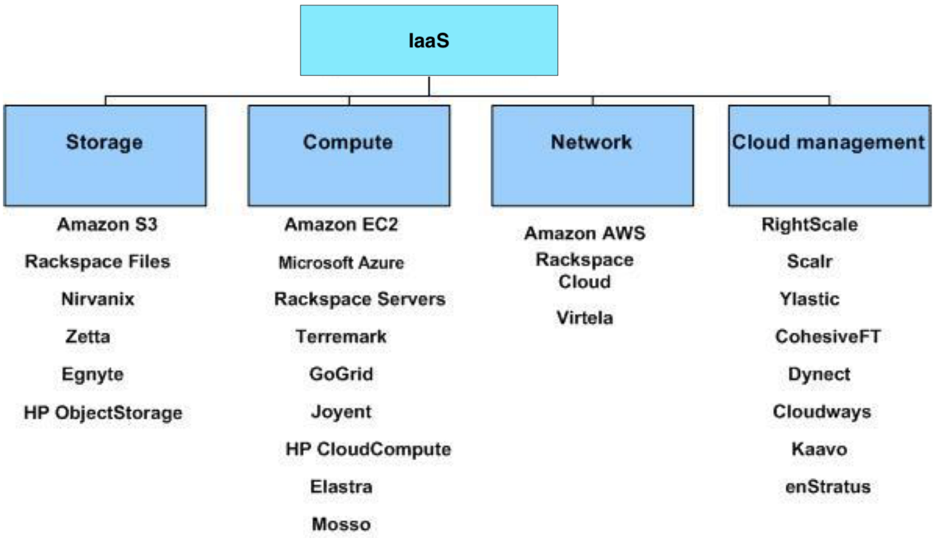


Figure 2.17: Public cloud services taxonomy

For example, in the computing area, based on various CPU, random access memory (RAM), and I/O resources, pricing models vary across different providers. Usually, providers offer different types of instances (with different configurations, for instance, small, medium, and large computing instance) to meet customer requirements. The most popular pricing models are on-demand, prepaid (reserved), and spot (auctioned).

- **On-demand model:** The on-demand model suits cases such as where the user does not have good estimation of the required resources, or for start-up companies who may require resources only for a short time.
- **Prepaid model:** The prepaid (reserved) model provides cost benefits when producing a long-term demand estimation for computational resources is feasible. This model works well for larger organisations and those who have an initial investment.
- **Spot model:** The spot model is usually used in a non-production environment where, for a short period, customers can afford to lose their virtual machine. Finding the cost of running an application that consumes resources such as network, CPU, and memory is a challenging task. For large-scale distributed applications, there is a need to use a third-

party framework that supports estimating required resources for a task. The third-party framework can tune the application so that eventually resources are utilised and costs are minimised while performance is improved and maintained.

Several studies propose different methods for metering and estimating the cost of running an application in a cloud environment. For example, Siew et al. [SHL12] used a queuing model for predicting the cloud computing resources for an application. A user can choose a high-quality cloud configuration and estimate the deployment cost. Huihong et al. [HH12] proposed an approach to estimate the cost of running an application on AWS during the design phase. They modelled the application execution service with a UML activity diagram. Different cost models using a cloud pricing schema are proposed [TD10, Dee08]. For example, Ewa [Dee08] et al. offer an amortised cost model of a long-running task where the cost is simply the price per virtual machine-hour multiplied by the total running time of the task in hours.

Mehrotra et al. [MSBA14] introduced an autonomic performance management approach that provides dynamic resource allocation for deploying a set of services on a cloud computing infrastructure by considering both the availability and the demand for the cloud computing resources. The authors [ALK15] provide a way to select appropriate cost-aware WorkFlow Scheduling (WFS) approaches from the available pool of alternatives; to achieve this objective, they conducted an extensive review to investigate and analyse the underlying concepts of the relevant approaches. A pricing model of cloud services based on a useful operational cost model of running an instance for different providers such as Amazon and Rackspace are proposed [NRRD12]. They used performance counters from hypervisor vendors to obtain cloud instance resource utilisation. In other work, cost-based scheduling is used to meet the performance expectations of workflow-based applications [BKK$^+$08]. A cost-aware provisioning system is built that exploits the resource heterogeneity of cloud infrastructures before any resource selection [SSSS11]. By using an offline profiling technique, this work was able to estimate the maximum performance capacity of a resource by running an application on different resource types, and subjecting them to a gradually increasing synthetic workload [DGVV12].

2.4 Chapter summary

Big data analytics includes the processing of large and varied data sets. Learning from vast and unstructured data brings significant opportunities to various areas. Traditional methods are not computationally efficient or scalable enough to analyse the data. In this chapter, the existing work in the area of big data analysis and associated technologies (e.g. cloud computing) are reviewed. Several data analytics and machine learning techniques for big data processing have been explored. The taxonomy of the existing tools, frameworks, and platforms in the area of big data computing are also discussed. It has been pointed out that the cloud computing environment is a suitable platform for big data analysis. Cloud computing provides an environment of flexible distributed resources that use highly efficient methods in the processing and management of data while reducing costs.

Several widely-used and useful technologies and data analysis techniques have been studied. Using machine learning over big data is challenging, especially when combined with cloud computing. Machine learning enabled distributed processing platforms are reviewed. Feature extraction underlies machine learning and involves techniques such as data segmentation. A need is identiied for a cloud-based distributed big data processing framework which supports the processes of data segmentation and feature extraction, machine learning model building, and classification.

In this work, a cloud-based machine learning architecture for big data is proposed that not only provides useful data pre-processing and machine learning techniques but also offers several optimisation components to enhance the framework's functionality and address the accuracy, performance, and cost efficiency challenges.

Chapter 3

Cloud-Based Classification Framework with Feature-category Model Optimisation

3.1 Introduction

In many areas of business and research, applying machine learning to large amounts of data (big data) is becoming increasingly important. Several solutions including hardware platforms, more efficient algorithms (e.g. machine learning algorithms), modern technologies (e.g. cloud computing), and various specific optimisation have been developed to help handle the demands of these big data applications.

A cloud-based framework, CLoud-based Optimisation Architecture for Machine-learning (CLOAM), is proposed to address the challenge of the efficiency of Machine Learning (ML) over big data. CLOAM offers a generic machine learning architecture and several optimisation components. The optimisation components improve the architecture performance and machine learning accuracy in a general way for any data type and any dataset. It offers self-adjusting and adaptive solutions and responds to the needs of a specific application (e.g. medical or environmental images) and specific dataset for that application.

The generic machine learning architecture includes stages of segmentation, feature extraction, model building, and classification. In this chapter, the framework, its machine learning architecture, and a proof-of-concept feature-category model optimisation component are presented. A case study of identifying the leaf types in an image dataset of different types of plant leaves is also designed to demonstrate the framework functionalities.

3.2 CLOAM framework

CLOAM (**CL**oud-based **O**ptimisation **A**rchitecture for **M**achine-learning) is a cloud-based framework for big data classification. Figure 3.1 shows the conceptual overview of CLOAM, which includes the underlying generic machine learning architecture and an optimisation component (feature-category model optimisation).

The CLOAM framework has two modes, Training Mode and Testing Mode. In the Training Mode, different Machine Learning (ML) models are built in parallel using a labelled training dataset. The ML models can be build using a single dataset (e.g. text or image), or from a combination of various types of information. For example, X-ray images and clinical notes associated with those images for the same issue can be used to create a comprehensive machine learning model. An optimisation component can evaluate the multiple models using a labelled testing dataset, and determine the best model under some criterion (e.g. accuracy). In the Testing Mode, a user can use the best selected ML model to classify (identify labels) for their unlabelled dataset.

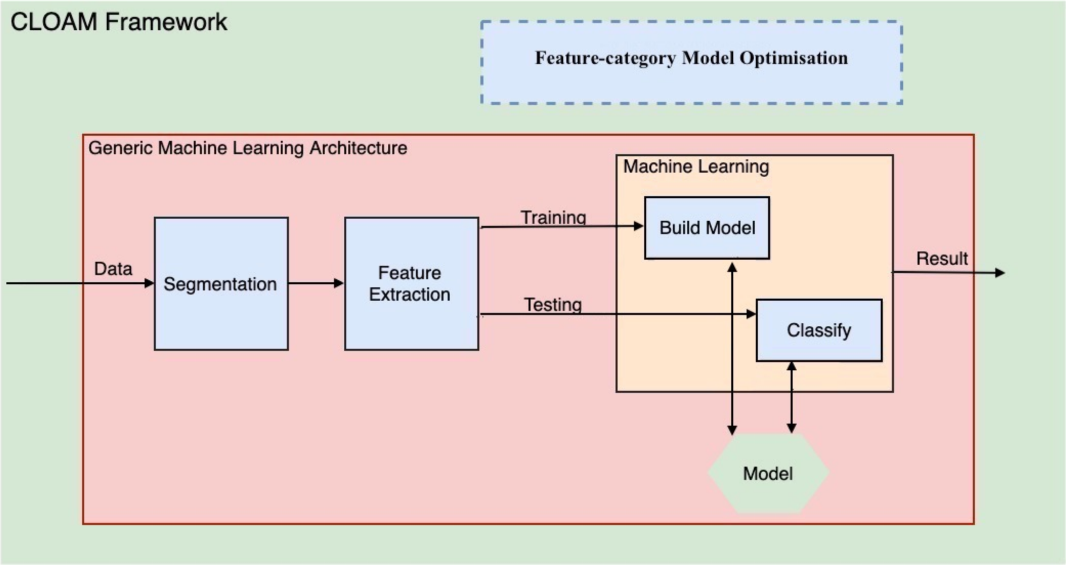


Figure 3.1: CLOAM framework

3.2.1 Private Cloud Environment

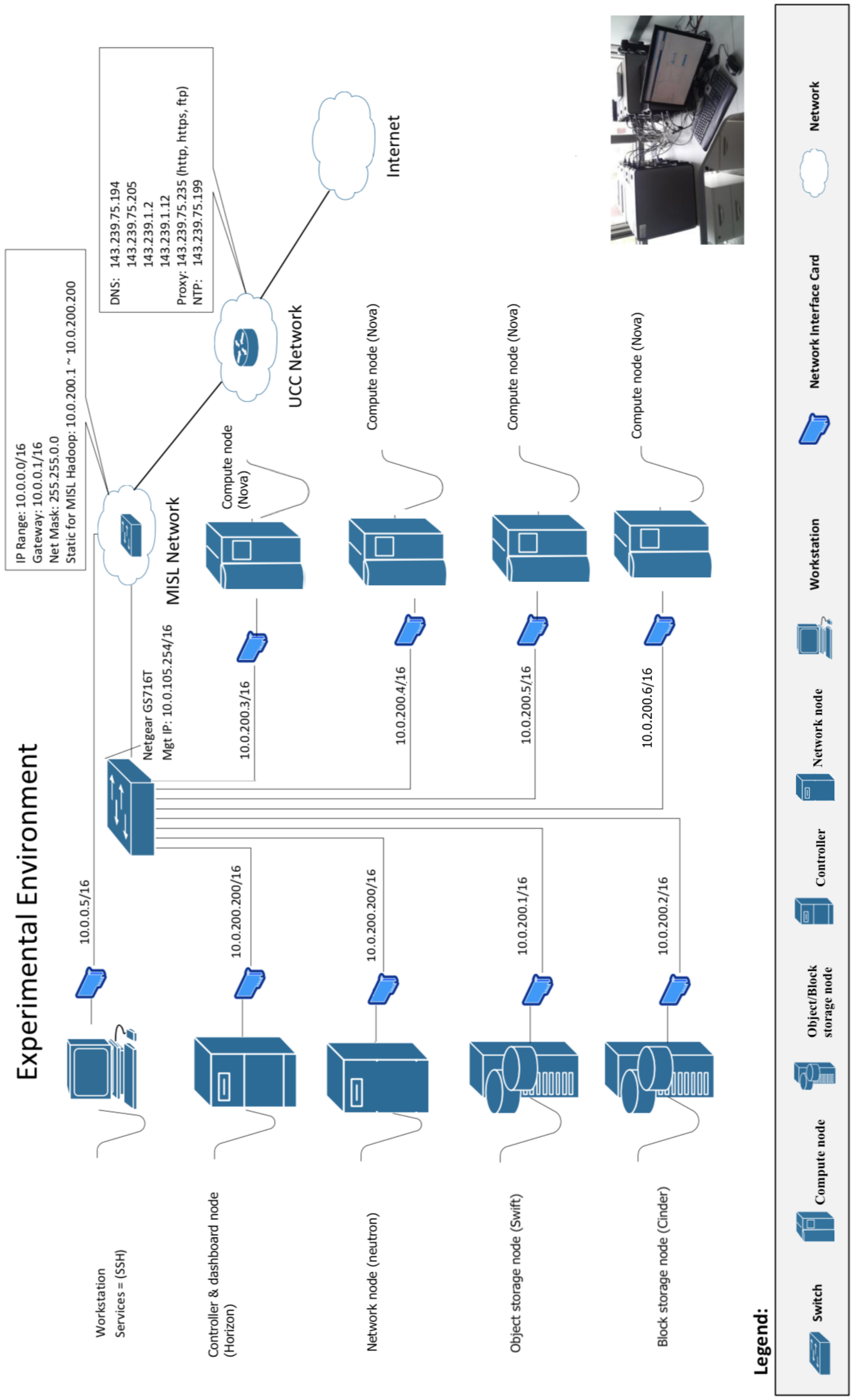


Figure 3.2: The configuration and architecture of our self-developed Cloud environment

Figure 3.2 provides a comprehensive view of our self-developed private cloud configuration and architecture.

3.2.2 Generic machine learning architecture

The generic machine learning architecture includes stages of segmentation, feature extraction, model building, and data classification. These are described here.

- **Segmentation:** Data segmentation is an essential technique in data preprocessing for ML. Segmentation is beneficial for many applications (e.g. medical imaging and locating objects in satellite images). In CLOAM, the segmentation stage is responsible for dividing and partitioning the data into smaller parts called segments. Based on the application, different methods (e.g. threshold, edge-based, region-based, and watershed-based) can be used for data segmentation [DY14]. Segmentation provides the basic elements (e.g. in a blood cell analysis scenario, individual cells are the basic elements) that are subject to subsequent analysis.
- **Feature extraction:** Feature extraction is another crucial step for data preprocessing. Feature extraction is the process of collecting discriminative information from data. As discussed in Chapter 2, Section 2.2.2, different features (low-level and high-level) can be extracted from unstructured data based on the type of data (e.g. image, text). The main goal of feature extraction is to obtain the most relevant information from the original data and to represent the information in a space of lower dimensionality, and create combinations of variables in order to obtain a reduced number of features while still describing the data with sufficient precision [SKN17]. CLOAM employs various extraction techniques to obtain features that are useful for data classification and predictive modelling. In this work, the features are categorised into different categories such as Shape and Colour.
- **Machine learning:** The CLOAM framework incorporates machine learning techniques of model creation and data classification. CLOAM provides two modes, Training Mode and Testing Mode. In the Training Mode, the extracted features (from a labelled training dataset) are used by a machine learning classifier (e.g. Decision Tree) to build a model. Different models can be created for the same dataset. For instance, the feature

category optimisation uses different models that are built using the same classifier but with different feature categories. The classifier ("ML-A") uses the feature category "Shape" and builds a model "Model-Shape" while the same classifier ("ML-A") uses another feature category "Colour" and builds a different model "Model-Colour". Accuracy and performance of the different models depend on several factors such as the ML classifier used to build the model, the size of the training dataset, the diversity of data objects in the training dataset, and also the extracted features that can represent the data differently. In the Testing Mode, the models can be used to classify unlabelled datasets.

3.2.3 Optimisation component

The CLOAM framework offers several optimisation components. They are self-adjusting and can respond to the needs of a particular application. In this work, five areas of optimisation are explored: feature-category model optimisation, classifier optimisation, auto-adjusting feature optimisation, computing resource optimisation, and performance-aware cost efficiency optimisation component. The generic machine learning framework works with these optimisation components to address the challenging demands of using machine learning over big data. In this chapter, the feature-category model optimisation component is presented. This component involves evaluating multiple different feature models in parallel, using sampling and feedback to choose the best model (highest accuracy) for a particular dataset.

3.3 CLOAM system-level architecture

The CLOAM system-level architecture and several services are presented (shown in Figure 3.3). Note that the generic machine learning framework in this figure (lower right hand section) is a simplified version of the generic framework shown in Figure 3.1 This is because, in this initial proof-of-concept case study, the segmentation and feature extraction stages are amalgamated into the Model Builder components. This simplification was used due to the fact that each model builder is using different subsets of category features. In all the other case studies the more general framework design with separate segmentation and feature extraction stages, as illustrated in Figure 3.1, is used.

The CLOAM architecture is deployed in a self-developed private cloud environment (using OpenStack open source software framework). In this architecture, each component has a dedicated queue that enables it to communicate with other components. The messaging system and component functionality are described here.

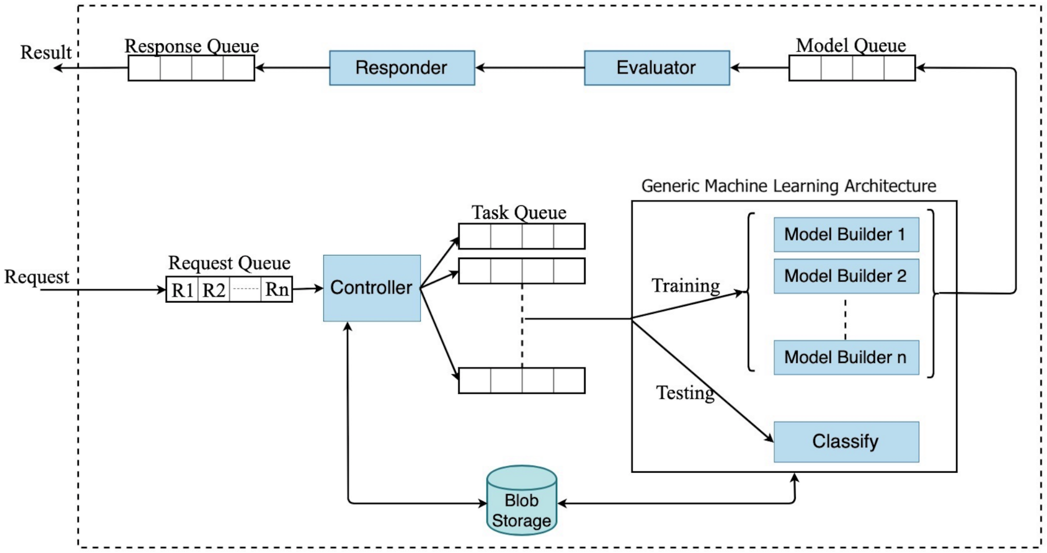
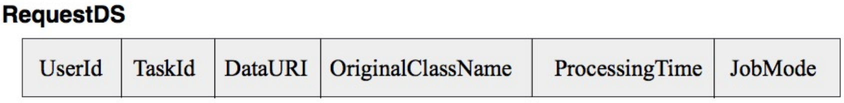


Figure 3.3: CLOAM system-level architecture

3.3.1 Messaging system

In the messaging system, queues are used between instances (nodes) to support asynchronous messaging. A message is stored in a queue until it is retrieved and deleted. Each message is processed. A processing component retrieves a message from a queue and processes it according to its responsibility. As Figure 3.3 shows, the CLOAM framework uses four different queues (Request Queue, Task Queue, Model Queue, and Response Queue) for sending/receiving messages between processing components. The message that is pushed to these queues contains a timestamp, unique identifier, and a textual field. Since both sides (sender and receiver) need to understand the structure of the message to interpret it, a data structure is designed for each queue. A technique is needed to serialise and deserialise an object to a text value (String). GSON is a library developed by Google to convert a programming object to a JSON (JavaScript Object Notation) value and vice versa, converting it back to a programming object. This lets the component convert their object to a string value (JSON),



RequestDS

UserId	TaskId	DataURI	OriginalClassName	ProcessingTime	JobMode

Figure 3.4: Request Data Structure

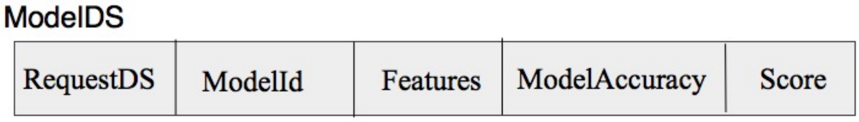
put it in a message, and push it to a queue. The other side receives the string value (JSON) and deserialises it to a programming object.

- **Request Queue:** Initially, a user sends a request to the CLOAM framework by pushing a message into the publicly visible "Request Queue." The framework has a controller instance that continuously monitors the "Request Queue", retrieves a request (sent by a user), and processes it. First, the message is deserialised and, then, is passed through the processing instances. The data structure of the "Request Queue" is described in Figure 3.4.
  - User Id: A User Id is an identifier that shows who owns the request so the result can be returned to the right requester.
  - Task Id: Each user can send an indefinite number of requests. Each task is given a unique identifier so that the result can be associated with the correct task.
  - Data URI: A request may contain a piece of data (an object or a full dataset). Since the queueing service is not able to handle large objects, the framework provides an API for uploading the object(s) to Blob storage (Swift). Swift is a service for storing large amounts of unstructured object data, such as text or binary data, which can be accessed through HTTP or HTTPS from anywhere in the world. The uploaded object(s) has a unique URI (Uniform Resource Identifier) which is accessible globally for permitted components within the framework.
  - Original Class Name: It is used in the Testing Mode of the framework, where the user has already classified the data and wants to test the framework on identifying the object.
  - Processing Time: As a request travels through the framework from one component to another, this attribute lets us know the processing time of each component and the waiting time of each request.

  - Job Mode: Job Mode attribute allows the controller to distinguish the mode of a request. Currently, the framework provides Training Mode and Testing Mode. In the Training Mode, there are two submodes: initial training using the training labelled dataset, and model evaluation using a testing labelled dataset in order to optimise the models.

- **Task Queue:** Each processing node has a queue called "Task Queue." The nodes keep monitoring their own queues and if there is a request they retrieve and process it. The data structure used in the "Task Queue" is the same as the "Request Queue."

- **Model Queue:** This queue is used in the Training Mode of the framework. The result of classifying a dataset using the training model will be prepared, and a message complying with the "ModelDS" will be generated. The data structure of the Model Queue is described below (Figure 3.5).



ModelDS

RequestDS	ModelId	Features	ModelAccuracy	Score

Figure 3.5: Model Data Structure

  - RequestDS: This contains all attributes in the RequestDS.

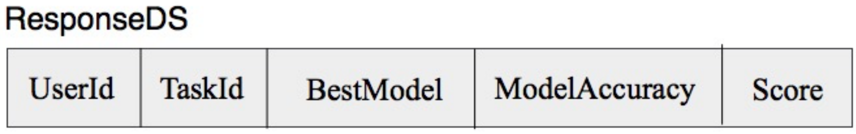
  - Model Id: A unique identifier for each model that is trained for a specific dataset. This identifier is used in the Training Mode of the framework, where different models are compared, and the best model is selected.

  - Features: There are a variety of features that can be extracted from a dataset. A machine learning classifier uses the extracted features to create a model. This attribute contains the name of the features used in training a model.

  - Model Accuracy: Machine learning model accuracy is the measurement used to determine which model is the best at identifying relationships and patterns.

  - Score: The summation of the correctly identified type of object in a dataset is the score of the model for object recognition.

- **Response Queue:** Once the Responder component receives the result from the Evaluator component, it sends a message complying with the



ResponseDS

UserId	TaskId	BestModel	ModelAccuracy	Score

Figure 3.6: Response Data Structure

"ResponseDS" to the Response Queue for the user. The data structure of the Response Queue is described below (Figure 3.6).

- User Id: The result will go back to the user requester.
- Task Id: The result will go back to the requester based on the Task Id.
- Best Model: The best model identified will be returned to the user.
- Model Accuracy: This contains the accuracy of the best-identified model.
- Score: This attribute contains the score of the best model.

3.3.2 Processing components

The framework consists of five processing components (Controller, Model Builder, Classifier, Evaluator, and Responder). They have different responsibilities. In this instantiation of the framework (Figure 3.3), one instance is assigned to the Controller, several instances are assigned to the Model Builders, one instance is assigned to the Classifier, and one instance is assigned to each Evaluator and Responder. The following are the descriptions of each processing component.

- **Controller:** The Controller is responsible for the control and management of incoming data and the addition of new messages to the Task Queue.
- **Model Builder:** This component is used in the Training Mode to create a machine learning model. Once a request is sent to the framework with a training dataset, this processing component reads a message from its Task Queue, and extracts a set of high-level features (e.g. shape and colour) from the dataset. Then a machine learning classifier (e.g. J48) uses the extracted features to build a model. At this stage, as shown in Figure 3.3, multiple Model Builder components can work in parallel to create different models. In order to rank the models and find an appropriate model

Table 3.1: Virtual machine instance configurations

Cloud Vendor	Instance Type	VCPU Core	Memory
OpenStack	tiny	1 Core	512 MB
OpenStack	small	1 Core	2 GB
OpenStack	medium	2 Core	4 GB
OpenStack	large	4 Core	8 GB
OpenStack	xlarge	8 Core	16 GB

(based on the best performance or the best accuracy), they should be evaluated. So, each Model Builder component evaluates its model using a labelled testing dataset. Each of them sends its evaluation result (the accuracy) to the Evaluator component.

- **Evaluator:** This component receives the evaluation results from all Model Builder components and compares them to identify the most accurate model. The best-identified model will be stored in a data storage and its related information (e.g. identifier and its URI) is sent to the Responder.
- **Classifier:** If the CLOAM framework is in Testing Mode, this component reads a message from its Task Queue and performs the extraction of features used in building the model and the classification of the dataset (unlabelled dataset) using the extracted features and the machine learning model.
- **Responder:** This processing component is responsible for submitting the identified model to the Response Queue, so the users can receive the result of their requests.

3.4 Testbed

OpenStack is used to build and manage the private cloud environment. It has a set of service for the creation and management of private cloud computing. The Nova service allows the user to provision and manages virtual machines (VMs) of various configurations. In OpenStack, when creating an instance, different sizes of predefined flavour such as tiny, small, medium, large, xlarge (Table 3.1) can be used.

A flavour defines the compute, memory, and storage capacity of a virtual server. Each instance has an IP address for accessing the image remotely. Based on the configuration of CLOAM, the number of used instances will be different. Eight physical machines are used in developing our private cloud environment. Four of them are used for the compute nodes (Nova), one for the controller node and dashboard node (Horizon), one for the block storage node (Cinder), one for the object storage node(Swift), and one for the network node (neutron). These nodes are described in detail in Chapter 2, Section 2.3.5.1. In this work, the Nova nodes build a pool of computing resources and, using this, several virtual machines (VMs)/instances can be instantiated. The number of VMs can vary based on the configuration of the CLOAM framework.

In the framework, the instances require a messaging system for communication. For this purpose, a queueing service, RabbitMQ, will be used. RabbitMQ is a messaging service that is implemented in OpenStack for internal communication between various OpenStack components and services. It has a general purpose message broker, employing several variations of a point to point and request/reply communication styles patterns. Communication in RabbitMQ can be either synchronous or asynchronous as needed.

3.5 Case study

In order to demonstrate that the proposed framework and its components function properly, a proof-of-concept experiment is performed. The framework can be evaluated using different kind of dataset. In this section, a case study is designed to build a model by processing a leaf dataset, use the best-identified model to classify a testing leaf dataset, and report the accuracy of the classification.

In this plant classification case study, several different features are extracted from the plant leaf. Each leaf contains significant information that can represent the type of the plant and support identifying and classifying the plant type. The dataset used consists of three types of leaf images [CR11]. They are divided into three classes based on the leaf type (Pittosporum Tobira, Betula Pendula, and Cercis Siliquastrum) shown in Figure 3.7. In this section, from the labelled training dataset, 120 images (40 images for each class) are used to build ML models, and 99 images (33 images for each class) are used for evaluating the built models. Finally, using a labelled testing dataset and by removing the labels from it, the models are used to classify the dataset, and using the known labels and the newly identified labels, the actual accuracy of the models for that dataset will be calculated and reported.

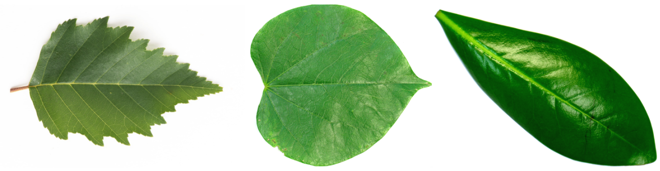


Figure 3.7: Leaves: Betula Pendula (Left), Cercis Siliquastrum (Middle), and Pittosporum Tobira (Right)

3.5.1 Methodology

Different data features (e.g. colour and shape) are extracted from an image using various feature extraction methods. Supervised machine learning is used in learning the patterns and creating different ML models to classify the objects in a specific dataset (in this case study, a leaf dataset). The proposed optimisation component evaluates the models and selects the best model with the highest accuracy. In this experiment, in order to evaluate the result of the proposed optimisation component, all available models will be used to classify an unlabelled testing dataset (given that its labelled version is known), the known labels and identified labels are compared, and the actual accuracy of each model is calculated. If the best-identified model of the optimisation component is the same as, or at least is one of the top three of, the testing best model(s), then the optimisation component is evaluated as successful. In this section, the feature extraction and machine learning methods will be described.

3.5.1.1 Feature extraction

In this experiment, several methods are used to extract the two high-level feature categories, **Basic features** category (e.g. edge and corner) and **Colour** category. In addition to these, I have designed an innovative method that produces two unique shape-based signatures (**Ordered-Signature** and **Sorted-Signature**) of an object in an image using the outer boundary shape of the object. In this section, the methods used in extracting the required features are explained in detail.

3.5.1.1.1 Edge detection: Edge detection is a fundamental technique used in image processing for feature detection and extraction [SHA08]. Edge detection is used to find discontinuities and identify points in a digital image where the brightness of the image changes sharply. The purpose of edge detection is to significantly decrease the amount of data in an image and store the structural properties that represent the object in the image for further image processing. The Canny edge detector technique is chosen because of having superior accuracy to other algorithms [ZHA01]. An example of Canny edge detection for one type of leaf (Cercis Siliquastrum) is shown in Figure 3.8. The white lines are the edges identified from this image by the Canny edge detector algorithm.

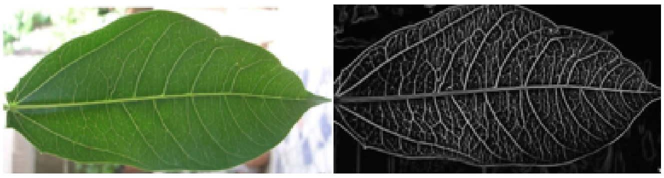


Figure 3.8: Left side: original image; Right side: Canny edge operator applied.

3.5.1.1.2 Corner detection: Corner feature is an important feature used in image processing [SZ13]. Corner detection is a low-level image processing technique, widely used in various computer vision applications. Generally, a corner in an image is a point on the contour at which two straight edges meet at a certain angle or the location at which the contour direction changes significantly. The Harris corner detection algorithm is one of the most famous corner extraction methods [HS88]. Harris corner detection is based on the autocorrelation of image intensity values or image gradient values. The corner features extracted by using the Harris corner detector method are analysed for different values of sigma, threshold and radius [SSa11]. Figure 3.9 shows the Harris corner detection for one type of leaf (Cercis Siliquastrum). In this image, all white spots are detected as corners.

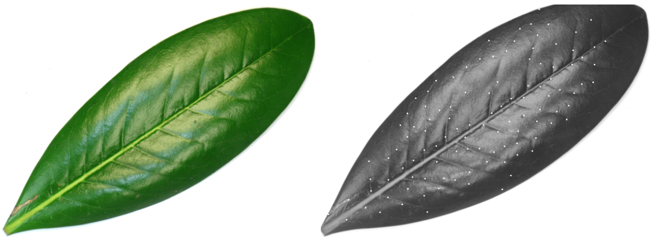


Figure 3.9: Left side: original image; Right side: Harris corner detector applied.

3.5.1.1.3 Colour detection: Colour space is a mathematical model with three or four different colour components representing colour information. For various applications, such as computer graphics, image processing, and computer vision, different colour spaces (models) are used. There are various colour spaces for detecting objects [PY11](shown in Figure 3.10).

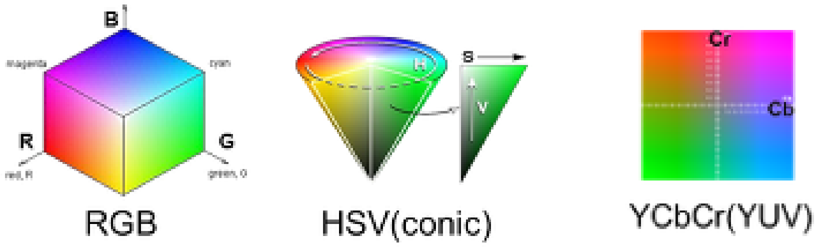


Figure 3.10: Colour model

The colour spaces are RGB based colour space (RGB, normalized RGB), Hue based colour space (HSI, HSV, and HSL), and Luminance based colour space (YCbCr, YIQ, and YUV). One or more colour spaces can provide an optimal threshold to detect object pixels in a given image. The choice of appropriate colour space is often determined by the object detection method and the application. The first step is to resize the picture to 256 * 256 pixels. Secondly, the average of each space is calculated for all pixels, such as R, G and B. This process will be done for all spaces, and finally, nine numbers are produced. The next step is to obtain a histogram to calculate its statistical moments (mean, standard deviation , and bias). After this process, we have 27 image features (3*(RGB+YCbCr+HSV component))* 3 features(mean, standard deviation , skewness value)).

3.5.1.1.4 Shape detection: Shape representation and description techniques can be generally performed in two ways: contour-based and region-based approaches [SHA05]. Contour-based approaches are more popular than the region-based approaches [ZHA04]. This is because human beings are thought to discriminate shapes mainly by their contour features. Another reason is that, in many of the shape applications, the shape contour is of interest while the shape interior content is not important. The contour-based method is used in this case study. The contour-based shape detection technique exploits shape boundary information. In this work, an innovative shape-based feature is proposed that represents a signature of the outer boundary of an object. The proposed feature comes in two forms of Ordered-Signature and Sorted-Signature.

As shown in Figure 3.11, (a) contour detection is used to detect the outer boundary of an object; then, based on the contour, the centre of the object and the radius is calculated. In the mask image that is shown in (b), the lines are drawn based on a predefined angle step (e.g. 30 degrees). This is the angle between successive radius lines drawn from the centre to the boundary. For example, in Figure 3.11, an angle step of 45 degrees produces eight lines from the centre of the circle to the boundary. The overlap of (a) and (b) will produce (c). From (c), the intersection of the object outer boundary and lines can be extracted.

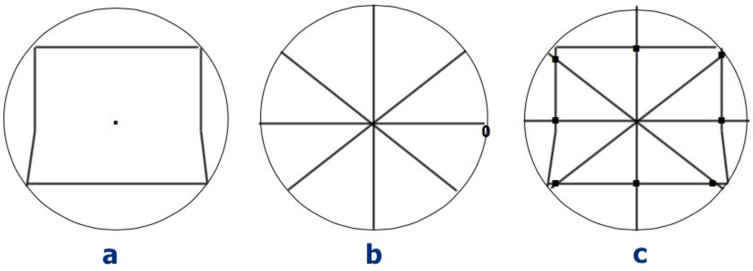


Figure 3.11: (a : Contour of object, b : Clock-wise lines, c: Intersection of lines and contour

The distance between the centre of the object and the intersections are absolute distance. Absolute distance does not work well as it depends on the scale. However, if the absolute distance is divided by the radius of the outer contour circle, it gives us a scale-invariant number between zero to one. By using this technique, it does not matter how small or big the object is, and the numbers produced by this technique will be the same for any size of the same object.

Sorting this list from high to low produces another signature, the Sorted-Signature. However if the same object is rotated, the signature will change, so the solution to this issue is to use a signature called the Ordered-Signature. Starting from the longest radius line and continuing clockwise, the list of radius lengths for an object produces the Ordered-Signature. These two object signatures are size and rotation invariant.

3.5.1.2 Modelling

Under supervised learning, the machine learning model is built using training data which contains labelled input, indicating the expected output. With the help of these, the model learns its function of providing an appropriate output when presented with an unseen input [MOH17]. Supervised learning methods are commonly used to generate predictive models. Tasks like classification are generally performed using supervised learning. In this section, the Weka machine-learning package [Wek15] is used as the basis for developing the machine learning components for model creation and data classification. Different classifier algorithms are available in Weka such as Naive-Bayes, Neural Network, Decision Tree known as J48, etc. (described In Chapter 2, Section 2.2.4).

3.5.2 Evaluation

The proposed framework provides various functionalities, such as building different models for a dataset (using a labelled training dataset), finding the best model (e.g. highest accuracy) and classifying a dataset (using an unlabelled testing dataset). Eleven models are built in parallel in the Training Mode of the CLOAM framework where each of them uses one or some combination of the high-level feature categories.

1. M1: Basic features (edge and corner).
2. M2: Colour (Three colour schemes, RGB, YCbCr, and HSV, and also the combination of all of them).
3. M3: Ordered-Signature.
4. M4: Sorted-Signature.
5. M5: Basic features and Colour.
6. M6: Ordered-Signature and Basic features.
7. M7: Sorted-Signature and Basic features.
8. M8: Ordered-Signature and Colour.
9. M9: Sorted-Signature and Colour.
10. M10: Colour, Ordered-Signature, and Basic features.

11. M11: Colour, Sorted-Signature, and Basic features.

For example, "Model Builder 1" (shown in Figure 3.3) extracts the Basic features (e.g. edge and corner) from an image dataset and uses an ML classifier (J48) to create a model (M1) while "Model Builder 2" extracts Colour features and uses the same ML classifier (J48) to create another model (M2). The feature-category model optimisation component identifies the best model with the highest accuracy. Figure 3.12 compares the accuracy of all the models and shows that three models, M1 (Basic), M7 (Sorted-Signature and Basic features, and M9 (Sorted-Signature and Colour features), provide the highest accuracy. The results show that even though a feature category, on its own, supports an ML model to provide high accuracy (e.g. M1 model that uses Basic features), a combination with another feature category does not necessarily provide higher accuracy. This issue can be seen by comparing the effect of the Colour feature when combined with others. The model of Colour feature (M2) provides 43% accuracy; the model of Basic and Colour feature (M5) provides 31% which is lower than that of their individual models (M1 and M2); however, the Colour feature positively affects the accuracy when combined with the Ordered-Signature to provide accuracy of 78% which is higher than their individual models (M2 and M3). These results differ for different datasets. So there is no permanent best combination of features that can be used in training an ML model, which will always provide the highest accuracy. Based on the results, the proposed feature-category model optimisation component selects the model that uses Sorted-Signature and Colour features (M9) as it provides the highest accuracy of 93%. Furthermore, it nominates the M7 and M1 models as the second-best and third-best models with high accuracy for this dataset (shown with green colour in Figure 3.12).

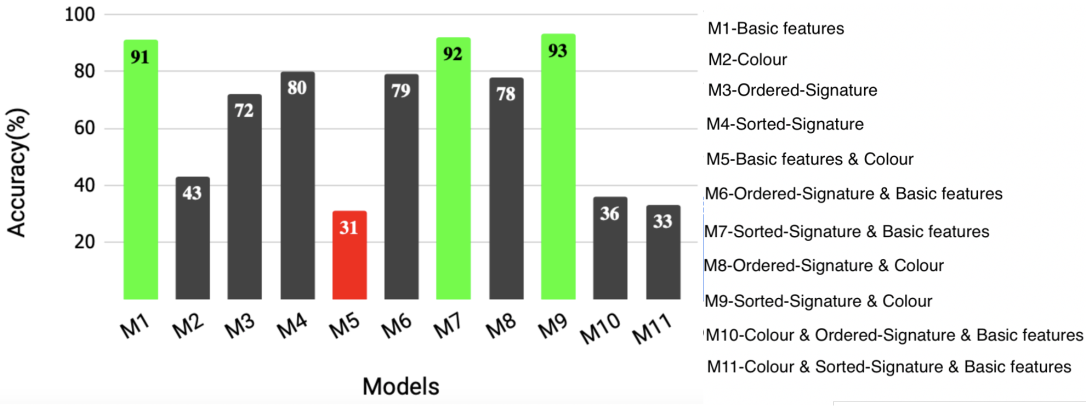


Figure 3.12: The actual accuracy of the built models in the Training Mode

Figure 3.13 reports the actual accuracy of the models in Testing Mode and the accuracy of models in Training Mode. The results show that the best-selected model of the optimisation component (M9) is among the top three models with highest-accuracy in Testing Mode (M7, M9, and M1). The best-selected model (M9), using the training dataset, provide accuracy of 93%. Figure 3.13 shows that the models performs almost as expected in processing the training and testing datasets; but sometimes it may not provide the expected result as the data representation of the training dataset is different with testing dataset. For instance, in the Training Mode, the accuracy of the model M9 (93%) is higher than the model M7 (92%), but interestingly in the Testing Mode, the accuracy of the model M9 (86%) is lower than the model M7 (89%).

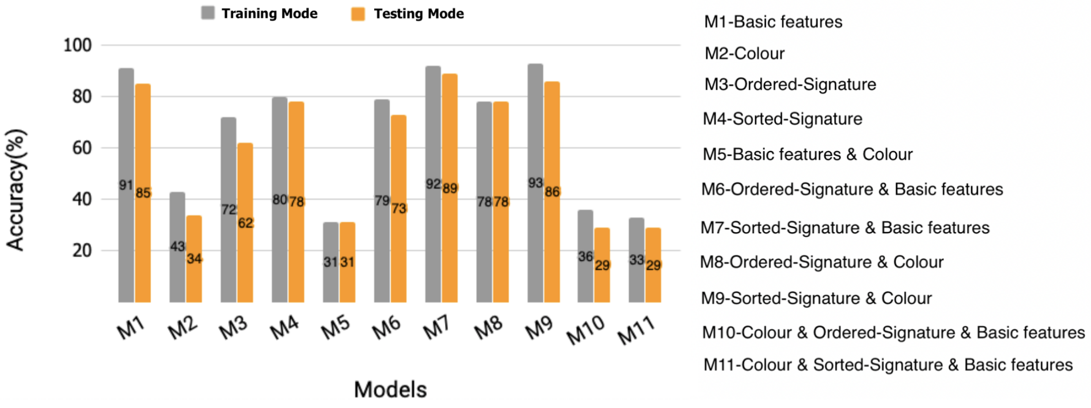


Figure 3.13: Model accuracy in Training Mode vs Testing Mode

3.5.3 Discussion & Conclusion

In this experiment, various models were created using different sets of data features extracted from a labelled training dataset. The models were evaluated, and the model with the highest accuracy was selected. Using an unlabelled testing dataset, in which the data labels are known to us, all built models were evaluated. The accuracy of the models in Training Mode and Testing Mode were compared, and the results show that the proposed optimisation component can successfully identify the best ML models for a specific dataset that can later be used to classify unlabelled datasets and perform well to their results in Training Mode.

All features may not be useful to recognise an object. The models (built by a specific ML classifier) with the most useful feature category provide the highest accuracy for a specific dataset. In conclusion, the proposed feature-category model optimisation was evaluated, and the results showed its usefulness and strength in identifying the best model (with the highest accuracy) by sampling a labelled dataset and then classifying an unlabelled dataset.

3.6 Chapter summary

An efficient cloud-based framework (CLOAM) is proposed. The framework offers a generic machine learning architecture and several optimisation components. The generic machine learning architecture uses different data segmentation and feature extraction methods and machine learning algorithms to create ML models and perform data classification.

In this chapter, in a plant recognition scenario, an image dataset of different types of leaves is used in order to evaluate the functionality of the CLOAM framework. A labelled training dataset and a labelled testing dataset were used in evaluating the proposed feature-category model optimisation component. The framework offers two modes, Training Mode and Testing Mode. In the Training Mode, several models are built by using several virtual machines (VMs) in parallel. Then, the models are evaluated by a new labelled dataset to show the usefulness of the optimisation component. The best model selected is the one with a distinct combination of features (i.e. a combination of Basic and Colour features) that were most effective for one specific ML classifier (e.g. Decision Tree) to train a model that provides the highest accuracy for that specific dataset. In the following chapters, more optimisations will be proposed to support the CLOAM framework in delivering better services to the user.

Chapter 4

Auto-adjusting: Feature Optimisation and Classifier Optimisation

4.1 Introduction

In the previous chapter, the CLOAM framework and its components such as the segmentation, feature extraction, machine learning, and fundamental model optimisation component were introduced. In this chapter, two additional components, a classifier optimisation component and a feature optimisation component, are proposed.

In Chapter 3, in the Training Mode, the proposed feature-category model optimisation component evaluates different ML models, built using the same ML classifier (e.g. Decision Tree) with a different set of features, to identify the best model for classifying the data. The best model selected is the one with the specific combination of features (e.g. a combination of shape and colour features) that were most effective for one specific ML classifier (e.g. Decision Tree) training a model for that specific dataset.

In this chapter, two proposed optimisation components enhance the framework by identifying the most useful subset of low-level features and best ML classifier based on the user preferences (i.e. desirable accuracy and/or desirable performance) for a specific dataset.

4.2 Feature optimisation

The feature optimisation component is designed to evaluate the features and sub-features and identify the ones that are significantly effective on the trained machine learning model. This process is called auto feature selection. It is performed by an ML classifier that uses the Correlation-based Feature Selection (CFS) method. Considering there are various categories of features (e.g. shape and colour), and each of them includes several sub-features (e.g. colour-red, colour-green, brightness, and shape-edge), the auto feature selection classifier selects a combination subset of the sub-features from all extracted feature categories. The most useful features and sub-features identified are those that can best support machine learning functionalities (e.g. classification) for a specific dataset.

4.3 Classifier optimisation

The classifier optimisation component is proposed to identify the most useful ML classifier in training the machine learning (ML) model for a specific dataset. The best model is selected based on user preferences such as best-accuracy and high-performance. Training a large size dataset is a time-consuming and costly task. So, the optimisation component uses a sampling, evaluation, and feedback mechanism. In the sampling section, a fraction (sample size) of the dataset is randomly selected. Different ML classifiers (e.g. Decision Tree and Naive-Bayes) and the extracted features from a fraction of a labelled training dataset will be used to build different ML models in parallel. The models will be evaluated using n-fold cross-validation method (in this study, 10-fold cross-validation) and the results are used to identify the most useful ML classifier for a specific dataset.

The challenge here is whether the accuracy for a fraction of a dataset corresponds to the accuracy for the whole dataset. A study [Doh08] showed that 35% of their dataset provided an accuracy close to that of the whole dataset with the accuracy error of 10%. In other words, training 35% of a dataset produces a model, which provides 90% accuracy of the original model (trained from the whole dataset). While, in general, the accuracy of a sample will depend on various factors such as the size of the dataset and its statistical distribution, the basic assumption for this optimisation is that the sample is a good reflection of the whole dataset.

4.4 Updated CLOAM framework

This instantiation of the CLOAM framework (Figure 4.1) uses the full generic architeture and offers three different modes, Training Mode, Auto-training Mode, and Testing Mode. Two optimisation components, a classifier optimisation component and a feature optimisation component, are integrated. These auto-adjusting optimisation components can automatically adjust the feature selection and ML classifier selection to improve the accuracy and performance of training a model and classifying a dataset as well as enhancing the overall performance of CLOAM.

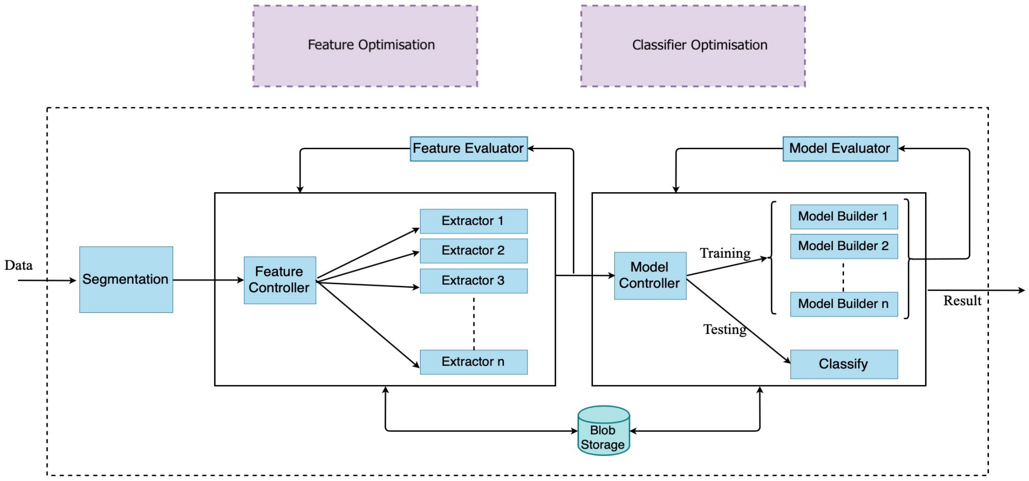


Figure 4.1: The updated CLOAM framework

As Figure 4.1 shows, the CLOAM framework uses several components to perform various tasks. Each component is deployed to a virtual machine instance created on the top of our self-developed OpenStack-based cloud environment. For instance, in the feature extraction stage, several VMs are allocated to the process of extracting various features from a dataset in parallel. Also, in the machine learning stage, several VMs are used to build various models in parallel. The VM instances work together to accomplish a task (e.g. training models and classifying a dataset). When the VMs in a stage (e.g. feature extraction stage) finish their tasks, they communicate to the other VMs instances using the message queues (described in Chapter 3, Section 3.3). In this section, the updated Training Mode of the framework and the new mode of Auto-training Mode are discussed.

4.4.1 Training Mode

In the Training Mode, the classifier optimisation component is used to build a training model. In this mode, a request containing the framework mode (Training Mode) and a dataset of a data type (e.g. image or text) is sent to the framework. As discussed in the previous chapter, first, the data segmentation is performed, and data is segmented into primary elements that are useful for subsequent analysis. Secondly, the Feature Controller sends a set of requests to the feature extraction instances that extract a set of distinct features (described in Chapter 2, Section 2.2.2) from the training dataset. For instance, "Extractor 1" (shown in Figure 4.1) uses the corner detection algorithm to extract corner features from an image, and "Extractor 2" uses a colour detection algorithm to extract colour features. Once the Model Controller receives all the extracted features, it combines them and sends them to various machine learning Model Builders. Each of these Model Builders uses a distinct ML classification algorithm (classifier). For example, "Model Builder 1" uses the decision tree classifier (J48) and "Model Builder 2" uses the Naive-Bayes classifier. In order to identify the best classifier for a specific dataset, the models will be evaluated. At the final stage, based on the user preference of high accuracy, performance, and any other preferences (i.e. balanced-accuracy-performance), the classifier optimisation component provides the best classifier and that will then be used to build a model for the whole dataset.

4.4.2 Auto-training Mode

In the Auto-training Mode, the goal is not only to identify the best classifier, based on user preference, but also to evaluate the features and sub-features and identify the ones that are more useful in building a predictive model. In this mode, from a sampled fraction of the training dataset, the most useful subset of features is selected by the feature optimisation component. Using the selected features, several learning classifiers build models and the classifier optimisation component identifies the most effective classifier for the training dataset.

For instance, in processing an image in this study, in the colour category, 27 low-level features are extracted (described in Chapter 3, Section 3.5.1.1). So, using a sample training dataset, the extracted features in all categories are evaluated, and the feature optimisation component selects the most useful features for that particular dataset. The best selected features may contain a subset of features in each category (for instance, in the colour feature category, 11 of the 27 sub-features). Feedback, containing the selected features, is sent to the controller. At this point, the classifier optimisation component then identifies the most efficient machine learning classifier using the best selected features. Finally, the whole dataset is trained by using the selected ML classifier and the best selected features.

4.5 Case study

A case study is designed to demonstrate an example of processing an image dataset from medicine (Scenario A) and botany (Scenario B). In both scenarios, the introduced optimisation components are used, in order to create an appropriate training model based on the most effective sub-features and the most efficient classifier for that dataset based on the user preference (e.g. accuracy or performance). These two distinct scenarios are used to demonstrate the usefulness of the framework in different application areas.

4.5.1 Scenario A: leaf images

In the first scenario, a leaf image dataset is used [CR11] (the same dataset that was used in Chapter 3, Section 3.5). The leaf image dataset consists of three plant classes based on their leaf types (Pittosporum Tobira, Betula Pendula, and Cercis Siliquastrum). Different sizes of training datasets (100, 200, 300, 400, and 500) and testing datasets (100, 300, and 500) are used.

4.5.1.1 Methodology

This section discusses the methods used to extract features from an image and machine learning techniques used to build a training model and classify new objects (in this scenario, a leaf). As previously discussed, selecting a normally distributed sample dataset is a challenging task. In this scenario, based on our control assumptions, a fraction (35% of the data, minimum of 20 data items, maximum of 50 data items) of the training dataset is used by the feature optimisation and classifier optimisation components.

4.5.1.1.1 Feature extraction:

Several methods are used to extract features such as Basic (e.g. edges, lines, and corners), Shape and Colour feature categories (described in Chapter 3, Section 3.5.1). In this section, in addition to the mentioned features, some other textural features (e.g. energy and contrast) are extracted from images. The textural features reflect the statistical arrangement of the pixels in an image. The method used to acquire textural characteristics from the images in this chapter is described.

- **Textural feature extraction:** A Grey-level Co-occurrence Matrix (GLCM) is a statistical method of examining the textures. It considers the spatial relationship of the pixels [MSG13]. It functions to characterise the texture of an image by calculating how often pairs of pixels with specific values and in a specified spatial relationship occur in an image, creating a GLCM and then extracting statistical measures from this matrix such as Energy, Entropy, Contrast, Variance, and Homogeneity [ZP12].
  - **Energy:** This statistic measures the textural uniformity of pixel pair repetitions. It detects the disorder in textures. Energy has a maximum value equal to one. High energy values occur when the grey level distribution has a constant or periodic form.
  - **Entropy:** This statistic measures the disorder or complexity of an image. The entropy is large when the image is not texturally uniform, and many GLCM elements have minimum values. Complex textures tend to have high entropy.
  - **Contrast:** This is the difference between the highest and the lowest values of a contiguous set of pixels. It measures the number of local variations present in the image.
  - **Variance:** This statistic is a measure of heterogeneity and is a standard statistical variable. Variance increases when the grey level values differ from their mean.
  - **Homogeneity:** This statistic is also called the Inverse Difference Mo-

ment. It measures image homogeneity as it assumes larger values for smaller grey tone differences in pair elements. It is more sensitive to the presence of near diagonal elements in the GLCM. It has the maximum value when all elements in the image are the same.

4.5.1.1.2 Machine learning: Four common machine learning (ML) classifiers, Naive Bayesian (Naive Bayes), IBK (Instance-Based learning with parameter K), Decision Tree (J48), and Attribute Selected Classifier (ASC) are used to build different models. Each of the classifiers uses 10-fold cross-validation to evaluate its model. The data is randomly split into ten parts with equal size. One fold is designated as the validation set while the remaining nine folds are all combined and used for training. When all the classifiers produce their results (i.e. in terms of the accuracy and performance), the best classifiers (in terms of best accuracy, best performance, and mixed preferences such as accuracy-performance-balanced) can be chosen among all classifiers. Then the best selected classifier is used to build a model using the whole dataset.

- **Accuracy model:** The Accuracy model is the model produced by a classifier that provides the highest accuracy.
- **Performance model:** The Performance model is the model produced by a classifier that provides the best performance (shortest execution time).
- **Accuracy-Performance model:** This model has a combination of preferences, for accuracy and performance, such as accuracy above 70% with the best performance.

4.5.1.2 Evaluation & Discussion

In this section, the experimental results for model creation and data classification using the leaf image dataset, in the three modes of the framework (Training Mode, Auto-training mode, and Testing mode), are discussed.

4.5.1.2.1 Model building (using Training Mode and Auto-training Mode): In this section, the classifier optimisation and feature optimisation components are used to build the best model for the leaf image dataset.

Table 4.1 shows the number of extracted features from the leaf image dataset in the Training Mode and Auto-training Mode. In the Training Mode, 400 sub-

Table 4.1: Best features in leaf images dataset

Instances	Extracted Features	Best Sub-Features Selected
100	400	11
200	400	11
300	400	16
400	400	18
500	400	20

features (low-level and high-level features) are extracted from the datasets with 100, 200, 300, 400, and 500 instances. In the Auto-training Mode, the feature optimisation component uses a randomly chosen fraction of the training datasets to identify the most useful and practical features for training a model for each dataset. In this section, the ones that are more useful in training a model are identified as the best features for each dataset. For example, for the training dataset with 500 instances, only 20 sub-features are identified as the useful features. Extracting a subset of features in constructing a model is crucial as it can help to improve the accuracy and performance of the model. The selected sub-features can vary from one dataset to another as the selection process is dataset dependent. For instance, the eleven features (mentioned below) selected for the dataset with 100 instances can be different to the eleven features selected for the dataset of 200 instances. The selected subset of sub-features from the whole set of features (400 features) for the dataset with 100 instances are the following:

> *colour-green, colour-brightness, shape-corner, shape-outer-boundary-3, centre-of-gravity, sorted-signature-01, sorted-signature-02, basic-edge, edge-03, colour-hsv-h-std,* and *exposure-level-mean*.

The extracted features are used by four ML classifiers, NaiveBayes, IBK, J48, and ASC, to build four different models in Training Mode and Auto-training Mode. In the Training Mode, the classifiers use the combination of all extracted features (400 features) while in the Auto-training Mode, the classifiers use only the subset of most useful features (e.g. for the dataset with 500 instances, 20 features).

Table 4.2 shows the evaluation results (the accuracy and the performance of the classifiers) of the machine learning models built by different ML classifiers for 35% of the leaf image dataset (containing 500 instances). The accuracy and the performance(execution time) of the classifiers vary and depend on the

Table 4.2: Comparing the accuracy and the performance of the classifiers in Training Mode and Auto-training Mode (sample of leaf image dataset)

Classifier	T-Accuracy(%)	AT-Accuracy(%)	T-Performance(s)	AT-Performance(s)
NaiveBayes	91	97	5.8	1.2
IBK	89	97	3	0.9
J48	97	98	1.6	0.2
ASC	99	99	6.3	1.3
AT: Auto-training **T: Training**				

features and ML classifier used. Both the accuracy and the performance of all three classifiers (Naive Bayes, IBK, and J48) are improved in the Auto-training Mode. While the accuracy of the ASC classifier stays the same but it performs better (in terms of execution time) in the Auto-training mode. The accuracy of the Naive Bayes classifier is improved from 91% to 97%; its performance is also improved by decreasing the execution time from 5.8s to 1.2s. The reason for the improvement is related to the number of important features (in this experiment, 20 features) that are chosen in Auto-training Mode, so the ML classifiers learn from a smaller set of the features to classify the data items. Also, as the selected subset of features are the most useful features in learning from the dataset, the accuracy is also improved.

Finally, the ASC classifier provides the highest accuracy in both modes. It is used to build the Accuracy model by training the whole training dataset. On the other hand, the J48 classifier is selected as the best performance classifier because of having the shortest execution time in both modes. The Performance model is created by the J48 classifier training the whole dataset. J48 also has high accuracy (above the accuracy threshold) and is the best under the combined accuracy-performance criteria. Therefore, for this dataset, the Accuracy-Performance model is the same as the Performance model.

- **The performance of the CLOAM framework in different modes**

Total duration for building a model in the CLOAM framework differs in its different modes. Figure 4.2 shows the total duration for building a model in the Training Mode and Auto-training Mode for processing different versions of the leaf image dataset (100, 200, 300, 400, and 500 instances). In this section, the total duration is calculated from initial time of receiving a request to process a dataset to the building and storing of an ML model. The processing involves feature extraction, feature selection (for the Auto-training Mode), classifier selection, and model creation.

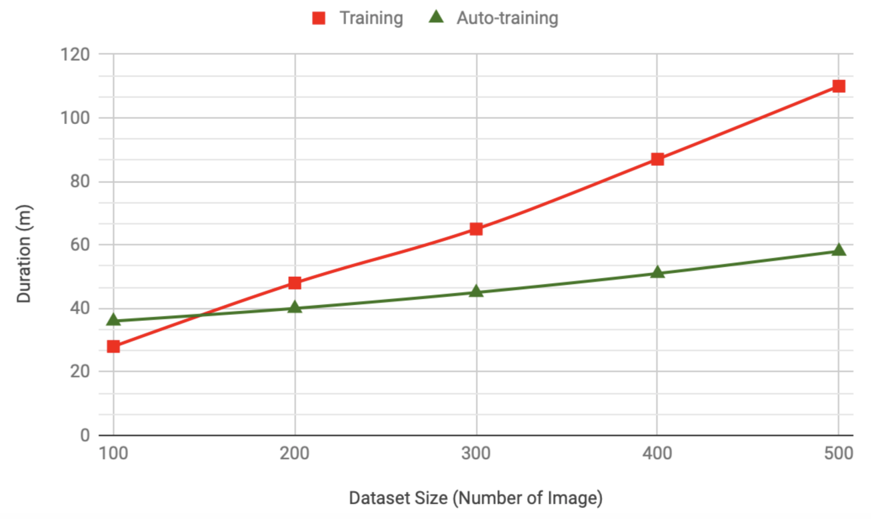


Figure 4.2: Total duration for building a model in Training and Auto-Training Mode (leaf image datasets)

In this experiment, 400 image-based features were extracted from the images using different image processing algorithms. The features not only include the ones discussed in Chapter 3 but also include the alternative low-level features such as edge, corner, etc., shape-based features such as lines, circles, etc., colour features, and several other features such as textural-based features using several different algorithms. Figure 4.2 shows that the Auto-training Mode may not provide better performance for small size datasets (in this case study, below 150 images), but it will improve the performance for larger datasets. In the Auto-training Mode, there will be additional processing in identifying the most useful features for a fraction of the dataset, but there will be less processing time in extracting the useful features, by the feature extraction components, for the original dataset. So, in the Auto-training Mode, the CLOAM framework performance is improved. For instance, for the dataset with 500 instances, the classifiers use 400 features in the Training Mode, but they use 20 features in the Auto-training Mode. The overall processing time for the dataset in the Training Mode is ≈110 minutes, while in the Auto-training Mode, it is ≈58 minutes.

4.5.1.2.2 Leaf image classification (using Testing Mode): Different unlabelled leaf image datasets (with a size of 100, 300, and 500 instances) are used to evaluate the best models (Accuracy and Performance models) created in the Auto-training Mode.

Because in this section, the Auto-training Mode gives the best performance (ex-

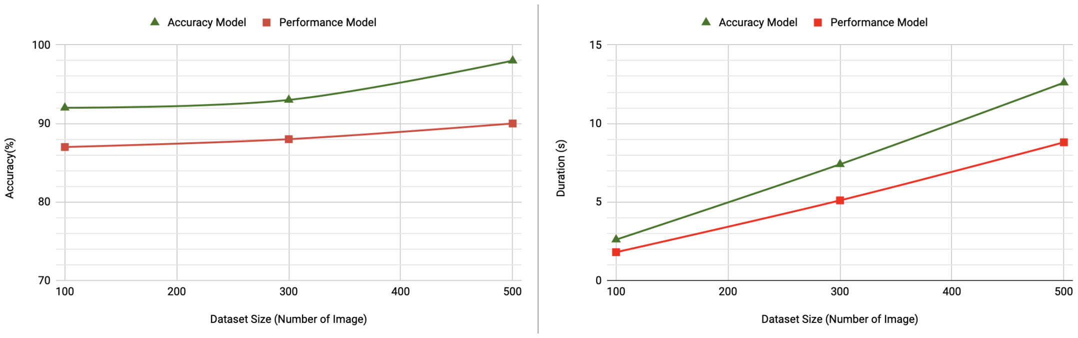


Figure 4.3: Evaluating the Accuracy model and the Performance model using the unlabelled leaf image dataset

ecution time for data classification) and accuracy of ML classifiers, so, only the best models (the Performance and the Accuracy models) created in the Auto-training Mode of the framework are evaluated using the unlabelled leaf image datasets. The leaf image datasets are classified using those models in the Testing Mode of the CLOAM framework. As shown in Figure 4.3, the more accurate results are achieved using the Accuracy model, for the classification of the datasets, while the results produced in the shortest possible execution time using the Performance model. The reported duration is the amount of time used by a classifier to classify all the data items in a dataset. For instance, the dataset with 300 instances can be classified with an accuracy of 87% in ≈5 seconds; it also can be classified with higher accuracy of 92% but that takes more time (≈7 seconds). This analysis shows that classifying a dataset with higher accuracy requires more processing time. So, based on the user criteria, one of the models can be selected to apply on the testing dataset.

Figure 4.4 shows the total duration for classification of the leaf images dataset in the Testing Mode of the framework. In this section, the duration is calculated from the start of feature extraction to the end of data classification. The same features that are extracted for the model building (the Accuracy and Performance model) in the Auto-training Mode, are extracted from these datasets. Then, the models are used to classify the datasets. The results show that for classifying the datasets, using the Performance model, the results are acquired quicker than where the Accuracy model is used. For instance, as shown in Figure 4.4, the dataset with 300 instances is classified in ≈15.5 minutes using the Performance model while it is classified in a longer time (≈16.7 minutes) using the Accuracy model.

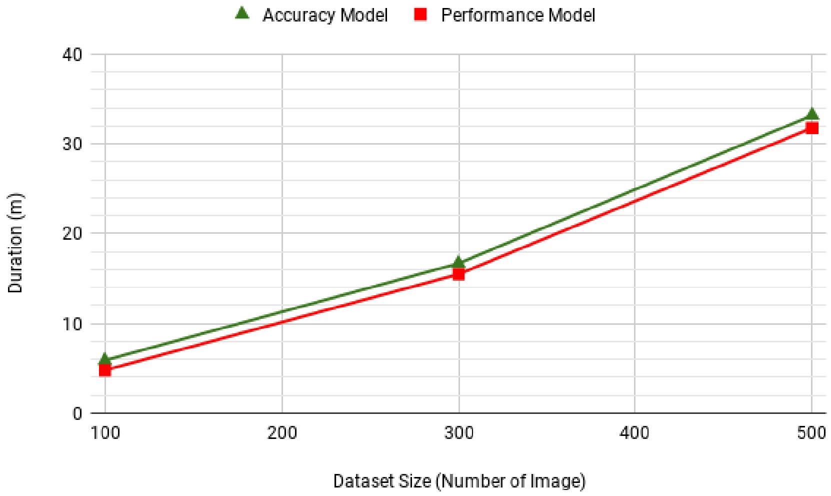


Figure 4.4: Total duration for data classification using the Accuracy model and the Performance model (unlabelled leaf image dataset)

4.5.2 Scenario B: medical images

Scenario A demonstrated that the Auto-training Mode (used for optimisation) offers advantages of performance and accuracy over Training Mode. Therefore scenario B uses Auto-training Mode. In this scenario, a medical image dataset (Acute Lymphoblastic Leukemia (ALL) lymphoblasts) [LPS11] is used for building a model using Auto-training Mode. This is an image dataset but provides a more challenging task as it contains blood cells that are important in diagnosing different diseases.

4.5.2.1 Methodology

The ALL lymphoblasts image dataset contains white blood cells (WBCs) and red blood cells (RBCs) microscope images. WBCs play a significant role in the diagnosis of different diseases and, therefore, extracting information about them is valuable for haematologists. As the classification of the lymphocytes in microscope images is onerous for pathologists, digital image processing techniques can assist them in their analysis and diagnosis [SD15]. In this section, first, the images are segmented, and individual WBCs extracted from the lymphoblasts images to sub-images. At this stage, each sub-image only contains one WBC. The WBC is then segmented into the cell, nucleus, and cytoplasm. In this scenario, different sizes of training datasets (100, 200, 300, 400, and 500) are used to demonstrate the usefulness of using the CLOAM framework.

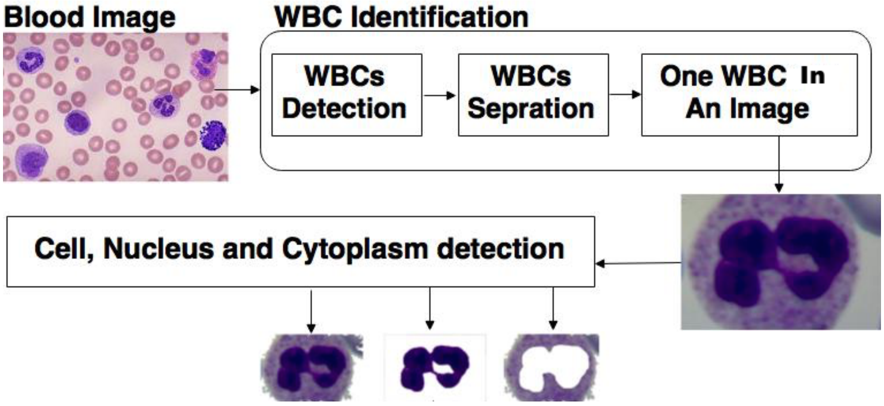


Figure 4.5: White blood cell segmentation

- **White Blood Cell (leukocytes) detection:**

  In the preprocessing step, the colour space of the input image is RGB space. A matrix represents a colour space, and typically, the matrix consists of three or four dimensions (e.g., RGB, HSI, CMYK). The RGB colour space is converted to the CYMK (C denotes cyan, Y denotes yellow, M denotes magenta, and K refers to a key plate (black)) space [SAB07]. It is observed that leukocytes have more contrast in the Y component of the CMYK colour model; this is because the yellow colour is represented in all the elements of the image except in leukocytes, where it practically is absent (Figure 4.6 shows an example). A redistribution of grey image levels is necessary in order to make the subsequent segmentation process easier. Then to obtain the WBC, Otsu thresholding [YOU15], which automatically calculates a threshold value from the image histogram is used.

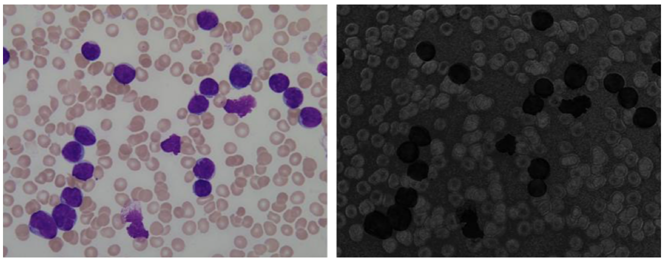


Figure 4.6: Left Side:Original RGB image and Right Side: Y component image

  In order to achieve a better result, the background of the image must be removed. Morphological operations such as opening and closing can be used to remove the background [ZRL77]. Opening operations allow deleting of all the objects with a size smaller than the structuring element [Vin92]. The structuring element used has a circular shape (Figure 4.7).

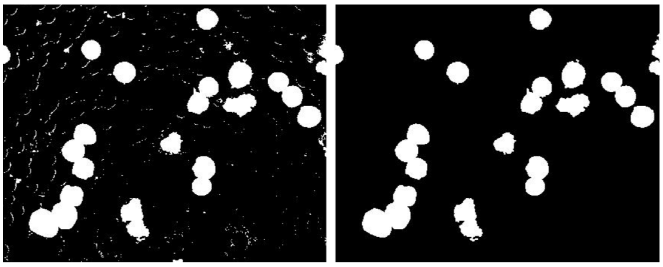


Figure 4.7: Background removal result and area Opening result

- **Grouped leukocytes separation and extraction of the nucleus:** Once the image containing only the WBC is obtained, where adjacent cells occur, they should be separated. In this section, the roundness value is used. All connected components that have a low roundness value are classified as grouped leukocytes, and so they must be divided. The watershed algorithm [Lin02] is used for dividing. Once the leukocytes are divided and identified, the second segmentation level that performs the selection of the nucleus and cytoplasm will be done. A threshold operation using Otsu [?] is applied to obtain a mask. The mask allows the nucleus to be extracted. In the end, to remove the cytoplasm, a subtraction operation between the binary image containing the whole leukocyte and the image containing only the nucleus is performed.

4.5.2.2 Evaluation & Discussion

In this section, the experimental results for model creation, using the WBC image dataset is presented. In scenario A, the results showed that building a model in Auto-training Mode is more efficient (in terms of accuracy and performance) than Training Mode, so, in this scenario, models are built using the Auto-training Mode only.

Table 4.3: Best features in White Blood Cell image dataset

Instances	Extracted Features	Best Sub-Features Selected
100	400	9
200	400	9
300	400	11
400	400	12
500	400	13

Table 4.3 shows the number of features extracted from the WBC image datasets in the Training Mode and Auto-training Mode. In the Training Mode, 400 low-level and high-level features (same as scenario A) are extracted from the image

Table 4.4: Comparing the accuracy and performance of different classifier algorithms for selecting the best classifier in the blood cell images dataset

Classifier	Accuracy(%)	Performance(s)
NaiveBayes	72	1.1
IBK	64	0.6
J48	67	0.3
ASC	71	1.2

datasets (100, 200, 300, 400, and 500 instances). In the Auto-training Mode, the feature optimisation component uses a randomly selected fraction of the training datasets to identify the most useful features for training a model for each dataset. For each dataset, the most useful features are identified from 35% of the training datasets. For example, for the dataset with 400 instances, only 12 sub-features are identified as the best features and used to train a model. The extracted features are used by four different ML classifiers (Naive Bayes, IBK, J48, and ASC) to build different models. In this experiment, the training dataset with 400 labelled images (in two classes, 200 blast cell images and 200 normal cell images) is used.

Table 4.4 shows the evaluation results of four different training models, each of them using a specific ML classifier. Each model uses the selected sub-features (12 features) to build a model for 35% of the dataset. For this dataset, the best classifiers are chosen based on the highest accuracy, best execution time, and a combination of accuracy and performance. For example, the NaiveBayes classifier provides the highest accuracy (72%) for 35% of the dataset. It is selected as the best classifier and is used to build the Accuracy model. J48 is selected as the best classifier in terms of performance by having an execution time of 4 seconds for the sample of the dataset and is used to build the Performance model. For creating the Accuracy-Performance model, both ASC and NaiveBayes provide accuracy above the 70% threshold. NaiveBayes is chosen from the shortlisted classifiers because it provides better performance for the sample dataset.

In this experiment, we demonstrated that the CLOAM framework can support a medical-based case study. The best performing machine learning models, in terms of accuracy and shorter execution time, can be built using the best-identified data features. The proposed framework can support scientists and researchers to achieve a better analysis of their datasets. The CLOAM framework can also facilitate the scenario of analysing mixed media dataset that includes text, images, videos, etc. For instance, a medical application may have textual clinical notes and lung X-ray images for each patient. The CLOAM framework can process both datasets, simultaneously in parallel, extracting features and building machine learning models. These models (with an additional correlation component) would support, for example, investigation of any links between lifestyle or occupation with certain lung diseases.

4.6 Chapter summary

In this chapter, two optimisation components, classifier optimisation and feature optimisation, are introduced. They use a general method of sampling, evaluation, and feedback to improve the accuracy and performance of classification (for any data type) and overall performance of the CLOAM framework. The optimisation components help the framework to learn which features are most useful and also to identify which algorithm(s) is best for the classification of a particular dataset. The feature optimisation component is responsible for selecting the sub-features (low-level features) that are most useful in data classification. The ML algorithms use these sub-features to build improved models. The classifier optimisation component is used to evaluate the built models and find the best classifier(s) based on user preferences (i.e. desirable accuracy and desirable performance) for a specific dataset. Finally, the final model is built by training the whole dataset using the selected features and classifier. Experimental results showed significant improvement achieved by the proposed optimisation components.

Chapter 5

Auto-scaling: Computing Resource Optimisation

5.1 Introduction

Dynamic scalability is a crucial function of a cloud-based computing solution. NIST [BGPCV12] defines scaling as the ability to request, receive, and release as many resources as necessary. The resources include the number of virtual machines (VMs), network bandwidth, and storage capacity of the VM. The cloud resources can be scaled up, down, in, and out automatically without human intervention under a dynamic workload to minimise resource costs while meeting the quality of service (QoS) requirements. Scaling up (down) resources results in increasing (decreasing) computing power, memory size, and/or network bandwidth. Scaling out (in), on the other hand, is a technique that instead of changing the capacity of currently available resources, expands (contracts) the architecture by adding (removing) resources. Auto-scaling functionality can help users achieve cost savings and better resource utilisation.

In this chapter, in order to enhance the effectiveness and overall performance of big data processing in the CLOAM framework, a computing resource optimisation component is proposed. The optimisation component involves dynamically adjusting (auto-scaling) computing power and also adding new computing instance in order to address the processing demand. Auto-scaling allows the framework to scale up/down the capacity of the available resources and scale out/in the resources as needed, and is essential for resource availability and optimum use. Depending on the application, the virtual machine instances can be rescaled to make efficient use of the hardware resources. Cloud users can specify a threshold value for a specific parameter (e.g. free memory and idle CPU usage) to be observed, and if the threshold is met, based on the auto-scaling rules, the framework can take action and improve the performance of the processing. CLOAM is deployed on the OpenStack software environment, and various OpenStack components are used in implementing the computing resource optimisation component to provide the auto-scaling functionality of the framework.

5.2 Computing resource optimisation

In this chapter, in addition to the previously introduced optimisation components, another optimisation component (computing resources optimisation) is introduced. Figure 5.1 shows the extended CLOAM framework.

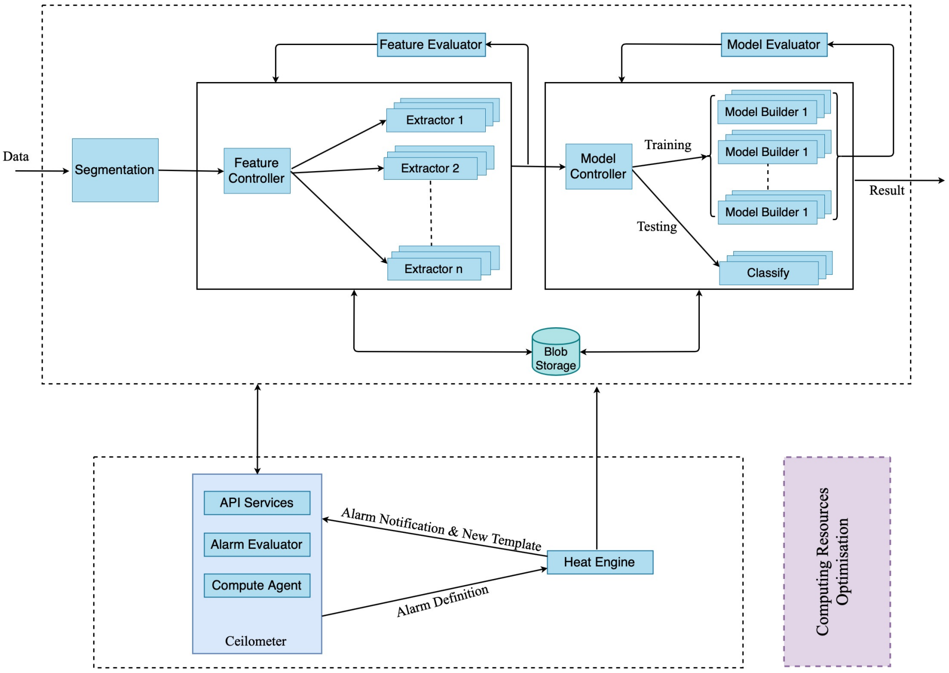


Figure 5.1: Auto-scaling in the CLOAM framework

There are several virtual machines (VMs) at each stage (feature extraction and machine learning) that work in parallel to perform various tasks. For example, in the feature extraction stage, one VM can be assigned for the shape feature extraction, and another VM can be assigned for the colour feature extraction.

One or more VMs can be allocated (depending on the application requirement) for processing a single task. For example, if the shape feature extraction algorithm consumes a large amount of time that results in slowing down the overall framework job, more VM instances can be allocated to accomplish that task more quickly. This functionality is performed by the computing resources optimisation component in the CLOAM framework. It dynamically adjusts (autoscales) the VM nodes hardware specification and resources to achieve better performance.

Cloud providers allow users to purchase as many VM nodes as they want. However, users need to specify how much resources are needed and for how long. In this situation, the CLOAM framework can provide auto-scaling functionality that allows a user to upload a big data application to the framework, and then the number of VMs can be automatically scaled based on the application. Scaling of resources are either horizontal (out/in) or vertical (up/down) (described in Chapter 2, Section 2.3.6). VM nodes are added or released to/from the system as needed in horizontal scaling. Resources (such as more CPU power or memory) are allocated in vertical scaling.

Two OpenStack services, Heat and Ceilometer, are used by the computing resources optimisation component to conduct auto-scaling of VMs (computing resources). Heat is the OpenStack orchestration engine. Heat manages the life cycle of cloud applications. Initially, resources required for cloud applications such as servers, storage, floating IPs along with their relationships are specified by a user in a text file (template). These resources are created by Heat using the OpenStack API, and the complete application is launched. Ceilometer is the telemetry infrastructure of OpenStack that collects measurements from the cloud. It gathers information about resource utilisation and performance. Its primary functions are monitoring and metering of resources and providing APIs for the retrieval of the collected data. Ceilometer contains storage and three main service components, as illustrated in Figure 5.2.

- Compute Agent: This component is responsible for gathering statistics (metrics) about resource usage.
- Alarm Evaluator: This component is responsible for triggering the alarm if the metric crosses a certain threshold.
- API Service: This component is responsible for the statistics for the last time window that determines the action to be taken when the alarm is

triggered. The API Service asynchronously notifies the alarm to the Heat engine.

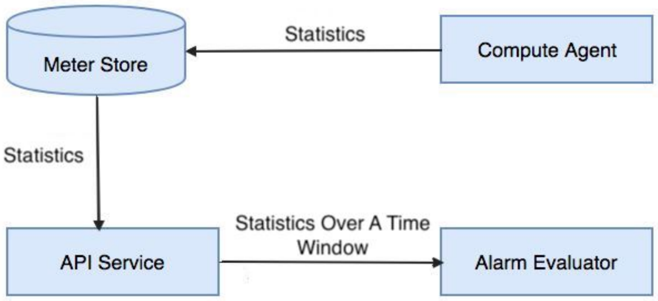


Figure 5.2: Ceilometer service components

The computing resource optimisation component uses Heat together with Ceilometer to produce an auto-scaling service. Heat is responsible for allocating resources dynamically by using a scaling group alongside using monitoring alerts (such as CPU utilisation) provided by Ceilometer. Compute instances (VM nodes) can be added on demand or removed when no longer required. When a scaling group is specified as a resource in a template, Heat provides auto-scaling based on alarms raised by Ceilometer.

In this section, the CLOAM framework starts with a minimum hardware configuration for each VM node. Each node can be scaled out/in based on the application needs in order to make efficient use of the resources. As Figure 5.3 shows:

1. A HOT (Heat Orchestration Template) file is sent to the Heat engine;
2. A new auto-scaling group is created by launching a group of VM nodes;
3. The Ceilometer alarms which, monitor all of the VM nodes in the auto-scaling group, are defined;
4. The system checks the alarm metric at each Ceilometer interval time;
5. The alarm will be triggered when the alarm metric passes the defined threshold values;
6. Scaling out/in will be performed based on the policy defined in the HOT file to improve the performance of the framework.

The framework may decide to resend a request to another node if a response to a specific request does not appear before a threshold waiting time.

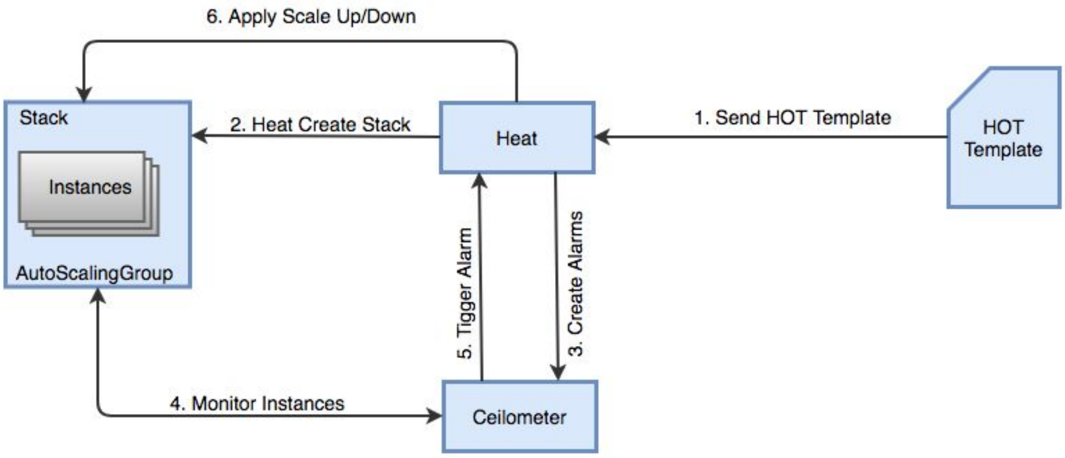


Figure 5.3: Autoscaling strategy

5.3 Case study

The proposed framework is evaluated with the introduced optimisation component. In order to evaluate the auto-scaling functionality of the CLOAM framework, a training labelled dataset containing movie reviews [kag16] is used. The dataset contains positive and negative user reviews of movies. The goal is to perform a sentiment analysis of the movie reviews. Opinion mining (also known as sentiment analysis) processes the subjective information in a text. The main goal of opinion mining is to classify feelings into positive or negative classes [HEM17]. Opinion mining can help in applications such as search engines, recommendation systems, and market research [Vu17]. Different sizes of training datasets (1000, 2000, 3000, 4000 and 5000 images) are used.

5.3.1 Methodology

This section describes a mechanism for auto-scaling in the CLOAM framework and techniques for extracting features from a text dataset to generate a model.

5.3.1.1 Auto-scaling mechanism

The model used by the computing optimisation component for management of the auto-scaling mechanism, is the MAPE (Monitoring, Analysis, Planning, and Execution) loop, proposed by IBM [McB05] (described in Chapter 2, Section 2.3.6).

- Monitor: A monitoring system is responsible for providing measured metrics about user demand and system status during the auto-scaling process. A variety of metrics such as CPU utilisation per VM, disk access, network interface access, and memory usage can be used as drivers for scaling decisions. OpenStack provides such information through the Ceilometer component.
- Analyse: The Analyse phase is for the analysis of the metrics collected from the monitoring system to determine the current system status.
- Plan: In the Plan phase, an intelligent controller makes a plan and decision based on the data taken from the Analyse phase such as the decision of adding or removing a node with a specific flavour to an application.
- Execute: The Execute phase consists of executing the scaling actions decided in the previous phase.

In order to decide the auto-scaling action in the Plan phase, a threshold-based rule is used [NCCA15]. Threshold-based auto-scaling policies are very popular among cloud providers like Amazon EC2 and third-party tools like RightScale [Rig18]. A threshold-based rule consists of some conditions that, when met, trigger some actions in the underlying cloud infrastructure. The conditions involve various performance metrics such as CPU usage, disk IO or memory usage. Upper and lower thresholds can be assigned to each performance metric. Whenever the observed performance metric is above or below a certain threshold, a predefined number of VM nodes will be added or removed from the system. This automation enhances the dynamic scalability benefits of the cloud by adding, in a transparent way, more resources to manage increasing workload and by shutting down unnecessary machines. In this way, the necessary resources for running an application can be adjusted in real-time based on its workload.

5.3.1.2 Feature extraction

One approach to detecting sentiment from text is the use of lexical resources such as a dictionary of opinion terms. One such resource is SentiWordNet [ES06], which contains information about words/terms extracted from the WordNet database. WordNet[1] is a large English lexical database. Names, verbs, adjectives, and adverbs are grouped into cognitive synonyms (synsets), each of which expresses a separate concept. SentiWordNet provides an annotation based on three numerical sentiment scores (positivity, negativity, and neutrality) for each WordNet synset. A Part-Of-Speech Tagger (POS Tagger) is used to tag the entire sentence [GSM$^+$11]. POS Tagger is a piece of software that reads the text in a particular language and assigns parts of speech, such as noun, verb, adjective, to each word. Then the words are passed to the SentiWordNet to check both the score and the polarity of the word. SentiWordNet returns the sentiment-type of that word (positive, negative, neutral based on score). Overall scores are calculated for each part of the text, together with score ratios for many terms/words. In this work, 28 features are generated from the dataset. Examples of the extracted features are summarised below.

- Sum of positive and negative scores and term count for Adjectives.
- Sum of positive and negative scores and term count for Adverbs.
- Sum of positive and negative scores and term count for Verbs.
- Sum of positive and negative scores and term count for Nouns.
- Ratio of the number of positive terms to the total number of positive and negative terms.

5.3.2 Evaluation & Discussion

This experiment is designed to demonstrate the analysis of textual data and to evaluate the efficiency of the auto-scaling of cloud resources. In this experiment, the training execution time of the Normal framework (without auto-scaling) and Improved framework (with auto-scaling) are compared.

[1] `https://wordnet.princeton.edu`

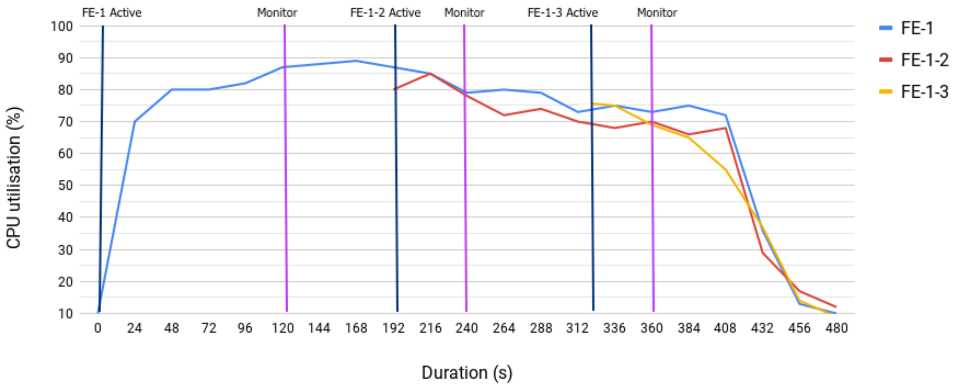


Figure 5.4: Scaling out/in of a feature extraction node based on the CPU utilisation

Figure 5.4 shows the scaling out/in of a single node ("Extractor 1" presented in Figure 5.1) based on CPU utilisation. This node is responsible for extracting a set of text features from a training dataset with 5000 reviews. CPU usage rapidly increases. Ceilometer collects CPU-related measurements every 2 minutes from the VM nodes and sends the result to the Heat service. A new node can be started and activated based on the Heat template. In the Heat template, the upper and lower thresholds for CPU utilisation are 80% and 20%. If the CPU usage of a node exceeds 80%, a new node will be added to the framework. Also, if the CPU usage is less than 20% then the VM node is removed.

In this experiment, the CLOAM framework starts with a minimum hardware configuration (1CPU, 4GB memory) for each VM node. As Figure 5.4 shows, the node FE-1 (Extractor 1) starts at time zero. Ceilometer collects and monitors the CPU utilisation of this node every 2 minutes and sends the result to the Heat service. Since the CPU usage of the first node (FE-1) exceeds 80% at time 120 seconds, the Heat service sends a request to the system to create a second node. The second node (FE-1-2) is started and activated after 192 seconds. The input data is then evenly divided between these two VM nodes. Similarly, the CPU usage of the FE-1 and FE-1-2 VM nodes are monitored at 240 seconds, and both are above the upper threshold, 80%, and so a third node (FE-1-3) is activated and started at 320 seconds. The input data is then divided between the three VM nodes, each performing the same job. At 2 minutes (360 seconds), the VM nodes are monitored again. Figure 5.4 shows that the CPU usage has decreased for these three VM nodes. Once the CPU reaches the lower threshold (20%), the Heat service sends the system a request to terminate the VM nodes.

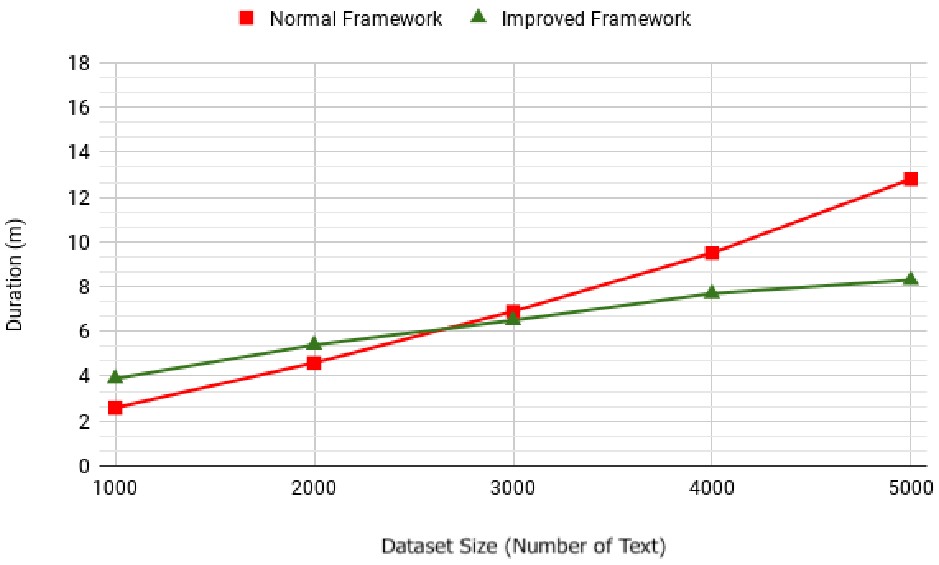


Figure 5.5: Job processing duration in Normal framework vs Improved framework

Figure 5.5 shows the processing time comparison for different jobs with the training dataset (1000, 2000, 3000, 4000 and 5000) in the Normal framework (without auto-scaling) versus the Improved framework (with auto-scaling). The configuration is set to allocate more resources with the maximum number of three VM nodes. The feature extraction is the most time-consuming stage. Since the Improved framework increases the number of VM nodes, the framework performance is improved. The Improved framework may not provide better performance for small size datasets (below 2000 items in this experiment), but improves the performance for larger datasets.

5.4 Chapter summary

In this chapter, the computing resource optimisation component is integrated into the CLOAM framework. This dynamically adjusts computing capacity and also adds/removes new computing instances (VMs) to address processing requirements. The CLOAM framework uses this optimisation component to address challenges such as where one task is slowing down the framework, or where a user requires a better performance. Auto-scaling allows the framework to dynamically improve efficiency based on the real-time performance of the computing resources (VM).

In this chapter, the OpenStack orchestration engine, Heat, is used to automatically scale out (or scale in) the compute resources based on alarms raised by Ceilometer, which collects measurements from the VM nodes and sends the results to the Heat service. Scaling actions are then carried out according to the scaling policy defined in the Heat template file. Experimental evaluation showed that the time to create the model in the Improved framework is lower than the Normal framework and that the overall performance of the framework is improved for datasets larger than 2000 items.

Chapter 6

Performance-aware Cost-efficiency Optimisation

6.1 Introduction

Data analytics can consume a large amount of computing power. With the increased computational power that cloud computing offers, data analytics is becoming easier and quicker but can be expensive, especially for big datasets. Cost-efficient use of cloud resources is, therefore, also important for big data analytics.

Users can access on-demand cloud resources to run their applications. They usually pay for what they use (pay-per-use model), but they may not always make the most cost-efficient use of the cloud computing resources, more especially where there are many cloud service providers each of which offers many different cost models. Users need to have sufficient knowledge about available cloud service providers and their pricing models to choose the most suitable cost-efficient provider, as well as predicting the computing demands of their application. These can be complicated tasks and not easy for users to perform. The cost model varies depending on various parameters such as the number of CPUs, memory and disk size, operating system, and network.

The previous chapter presents a computing resource optimisation that supports improving the overall performance of processing an application. Taking the cost into account suggests the idea of a cost combined efficiency optimisation to meet the user budget, and at the same time, improve the performance. In this chapter, a performance-aware cost-efficiency optimisation component is proposed.

The proposed optimisation component uses a high-level cost model based on the cost of available public cloud computing options and determines an appropriate cost model for a specific application based on user-defined criteria. The user can choose to prioritise performance, cost-efficiency, or can specify better performance but within a budget.

The optimisation component uses a sampling and feedback mechanism in which a sample of the dataset is evaluated, and some performance metrics such as CPU usage, memory usage, and execution time are extracted. The extracted information helps to identify the CPU and memory requirements for the processing instances (VMs). Using this information, a matching algorithm is used to find the best cost model where the performance is improved, and the cost is within the requested budget (if specified). Finally, the framework provides the user with the recommended cloud provider and recommended VM type(s).

6.2 Performance-aware cost-efficiency optimisation component

The overview of the CLOAM framework and the proposed optimisation component is shown in Figure 6.1.

Once a request of processing a dataset arrives in the CLOAM framework, a portion of that dataset is selected as a sample dataset, and the usual workflow of feature extracting and building a model will be performed. At this stage, several virtual machines (VMs) are assigned to the different stages of processing an application. As the framework is in sampling mode, the performance-aware cost-efficiency optimisation component is responsible for requesting measurements from the processing nodes, and for collecting the performance measurements (i.e. CPU usage, memory usage, and execution time) from all VMs. Ceilometer (an OpenStack service described in Chapter 5) collects measurements from VM nodes. Using this measurements, a matching algorithm is used to find the best cost model and the best type(s) of VM for a specific application. The optimisation component identifies the CPU and memory requirements of the VMs and, using this, it allocates an improved instance configuration (VM type) to any

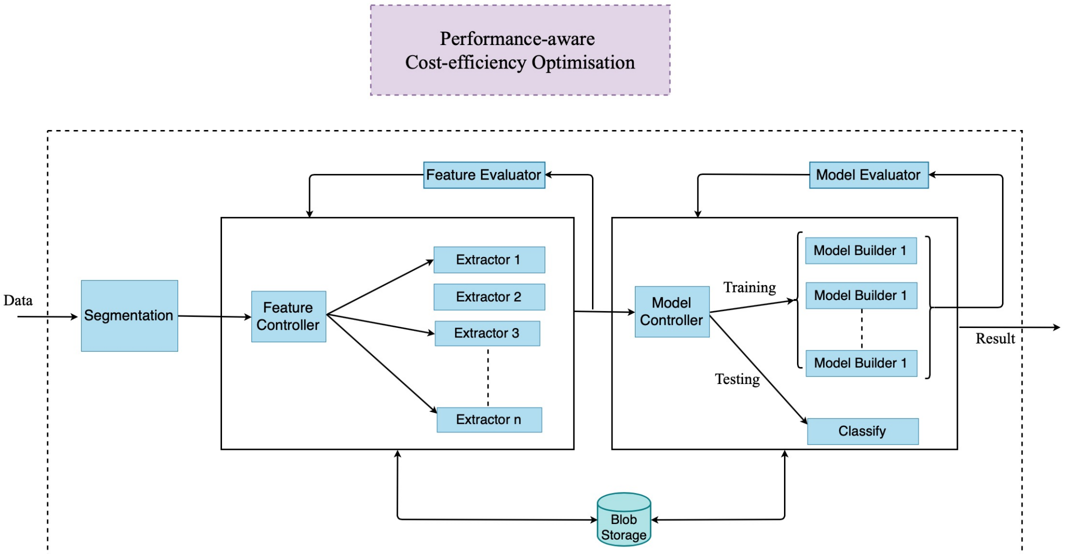


Figure 6.1: Performance-aware cost-efficiency optimisation in the CLOAM framework

instances which require better performance.

6.3 Experiment

In order to demonstrate how CLOAM and the performance-aware cost-efficiency optimisation component can cooperate to assist a user, two scenarios having different data types (image and text) are proposed.

6.3.1 Testbed

We have upgraded our private cloud environment by adding more compute nodes. In this new set-up, there are twelve physical machines in total, eight of which are used for computing nodes. We have set this to allocate the maximum number of 30 VM instances. The default instance configuration is 1CPU, 4GB memory, and 20GB disk.

6.3.2 Pricing model

In the proposed case studies, our private OpenStack-based cloud environment is used to experiment apart from the Amazon Web Services (AWS) pricing models that help to estimate the cost of running an application in the cloud.

Table 6.1: On-Demand virtual machine pricing model

Category	VM Type	Memory	Cores	AWS Price($)
General-Purpose	m3.Medium	3.75	1	0.07
	m3.Large	7.5	2	0.14
	m3.XLarge	15	4	0.28
	m3.2XLarge	30	8	0.56
Compute-Optimised	c3.Large	3.75	2	0.11
	c3.XLarge	7.5	4	0.21
	c3.2XLarge	15	8	0.42
	c3.4XLarge	30	16	0.84
Memory-Optimised	m2.Xlarge	17	2	0.25
	m2.2Xlarge	34.2	4	0.49
	m2.4Xlarge	68.4	8	0.98

There are different categories of on-demand VM models and, based on what the application is (e.g. compute intensive application or memory intensive application), one of them can be selected. The VM families are collections of VM types designed to meet a common goal. In this research, Amazon EC2 VM types are grouped into families based on target application profiles. For example, Amazon EC2 has ten different VM types, distributed across six VM families. The Amazon CPU and Memory configurations are used as the basis for performance testing. Disk I/O is not relevant to this work as diskless in-memory processing is used based on an object storage service (Swift). The on-demand VM pricing model for the three VM families is shown in Table 6.1.

- General-Purpose: This family provides a balance of CPU, memory, and network resources making it an excellent choice for many applications.
- Compute-Optimised: This family is useful for applications that need high computing power. Compute-Optimised VMs have a higher ratio of CPUs to memory than other families and the lowest cost per CPU of all the Amazon EC2 VM types.
- Memory-Optimised: This family is designed for memory-intensive applications. VMs in this family have the lowest cost per GB of RAM of all Amazon EC2 VM types.

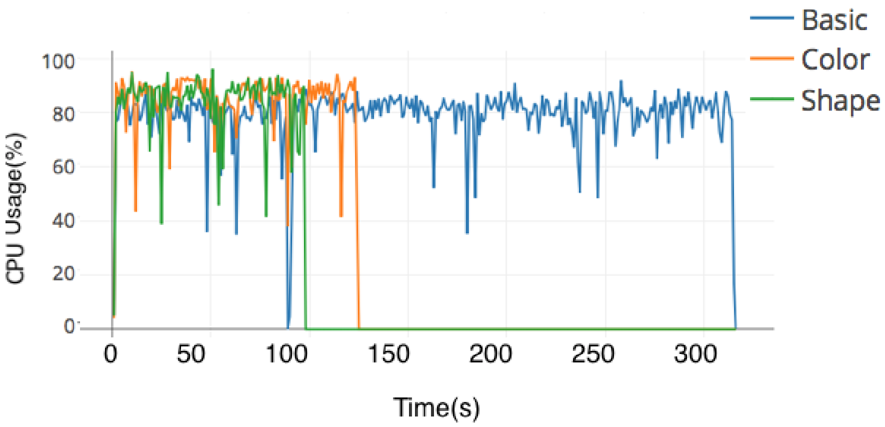


Figure 6.2: CPU usage for feature extraction (sample image dataset)

6.3.3 Case study

A case study is designed to demonstrate the processing requirements of an image dataset (Scenario A) and a text dataset (Scenario B). This case study involves evaluating a sample-size dataset, performing the CLOAM feature extraction and model building functionalities, identifying the CPU consumption and memory requirements for the framework, and finding the best cost model while the performance can be improved or maintained. These two different scenarios are used to show the differences between processing an image dataset and a text dataset. They show differences in the feature extraction stage. In scenario A, three feature extraction components named Shape, Basic, and Colour are employed to extract sub-features from the image dataset. In scenario B, three different feature extraction components named Feature 1, Feature 2, and Feature 3 are used to extract several sub-features to support high-level features such as sentiment, opinion, mood, and emotion.

In scenario A, a sample (10%) of the image dataset is selected. Figure 6.2 shows the CPU usage, and Figure 6.3 shows the memory usage in three VMs for the Basic, Colour, and Shape feature extraction algorithms. In this part of the experiment, the configuration of 1 vCPU, 4GB memory, and 20GB of disk space is designated to each of the VM instances.

6.3.4 Evaluation and discussion

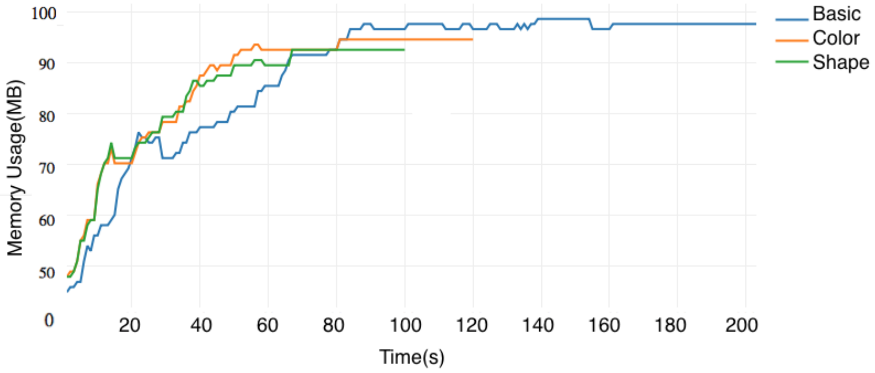


Figure 6.3: Memory usage for feature extraction (sample image dataset)

Figure 6.2 shows that the CPU is heavily loaded, reaching 87% at its highest usage. The figure shows that, for this specific dataset, the Shape feature extraction node finishes its task quicker than the others in ≈109 seconds, the Colour feature extraction task is finished in ≈140 seconds, and the longest processing time belongs to the Basic feature extraction task that takes ≈300 seconds. The Basic feature extraction node is identified as a high CPU-consuming node. Figure 6.3 presents the memory usage of the feature extraction components. The figure shows no pressure on memory since each node consumes less than 100 megabytes of memory while four gigabyte of memory is allocated to each node. These results imply that, for this specific dataset, the memory size is sufficient.

In scenario B, a textual dataset is processed. Figure 6.4 shows the CPU usage and Figure 6.5 shows the memory usage for the three feature extraction nodes in processing this dataset. The results show that the CPUs and memories are not heavily loaded. The CPU is not fully loaded (CPU usage is less than 60%).

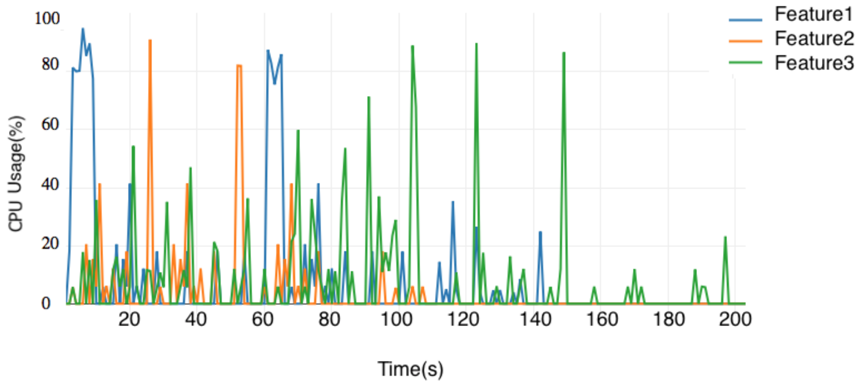


Figure 6.4: CPU usage for feature extraction (sample text dataset)

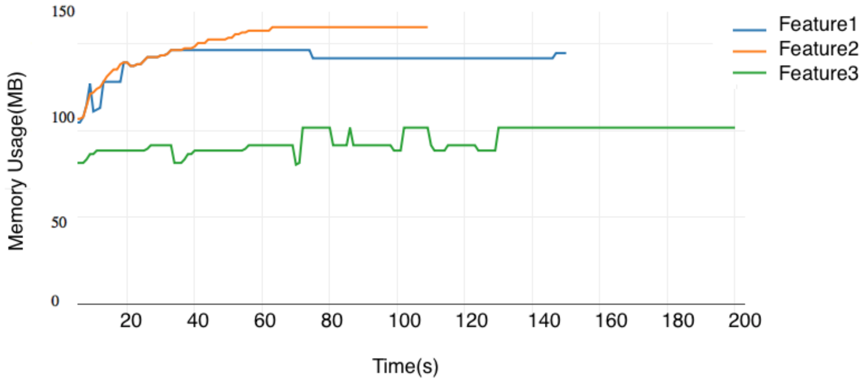


Figure 6.5: Memory usage for feature extraction (sample text dataset)

In this section, the model building stage (for machine learning), for the image dataset and its related performance measurements are discussed. Figure 6.6 and Figure 6.7 show the CPU and memory usage for four machine learning algorithms (M1-J48, M2-NaiveBayes, M3-ASC, and M4-IBK) for the sample image dataset. There is a high CPU load for the machine learning nodes. No memory pressure is detected, and the memory usage reaches 120 megabyte at its highest usage.

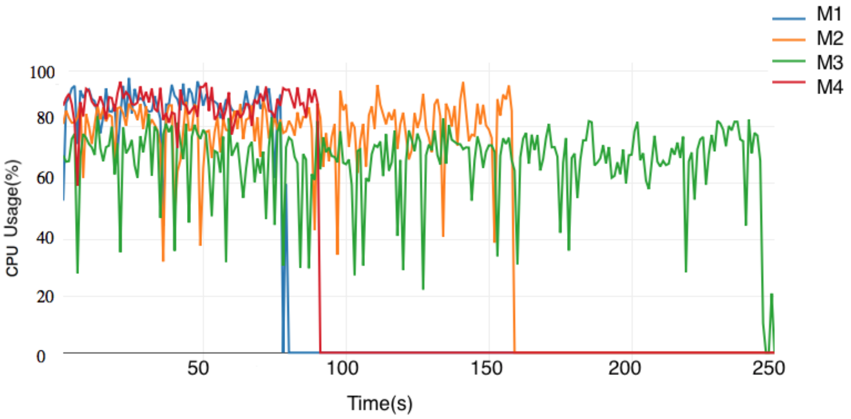


Figure 6.6: CPU usage for model building (sample image dataset)

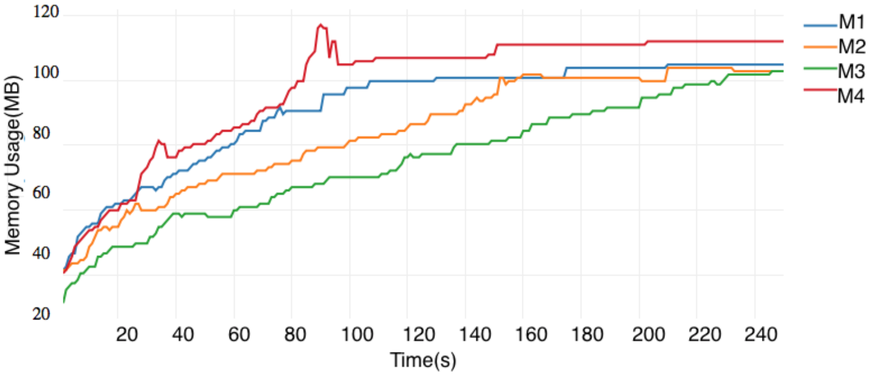


Figure 6.7: Memory usage for model building (sample image dataset)

Table 6.2: The required memory, CPU, and execution time for feature extraction for both sample datasets

Dataset	Feature	Average CPU Usage(%)	Averag Memory Usage	Execution Time(s)
Scenario A	Basic	87%	<1GB	309
Image	Colour	84%	<1GB	140
	Shape	83%	<1GB	110
Scenario B	Feature1	34%	<1GB	150
Text	Feature2	57%	<1GB	109
	Feature3	39%	<1GB	200

Table 6.2 shows the CPU usage, memory usage, and execution times of feature

extraction algorithms in both scenarios. The results show the differences between processing an image dataset and a text dataset at the feature extraction stage. The feature extraction algorithms loaded the CPU over 80% in processing the image datasets while the CPU is not loaded (below 50%) in processing text dataset. These findings indicate that the performance of the feature extraction stage differ for different applications.

Table 6.3: The required memory, CPU, and execution time for model building for the sample dataset.

Classifier	Average CPU Usage(%)	Average Memory Usage	Execution Time(s)
M1	87%	<1GB	84
M2	82%	<1GB	169
M3	84%	<1GB	213
M4	84%	<1GB	102
M1:J48, M2:NaiveBayes, M3:ASC, M4:IBK			

A more detailed analysis is performed (shown in Table 6.3) to calculate the execution time of each machine learning processing node. It reports that some processing nodes finish their task quicker than others. In this scenario, the M1 node performs its task in ≈84 seconds (which is the quickest), the M4 node is finished in ≈102 seconds, the M2 node is done in ≈169 seconds, and the M3 node finishes in ≈213 seconds.

On the one hand, knowing the performance measurements can help the optimisation component to determine an appropriate cost model for a specific application. On the other hand, knowing the performance measurements and identifying what part of framework slows down the job assists the optimisation component to consider allocating better computing options (e.g. High CPU) to the slower parts of the framework while still meeting the cost budget (if specified). For example, in scenario A, as different feature extraction nodes use different algorithms that performed differently, they finished their tasks at different time. However, as in the machine learning stage, all the extracted features are required to train a model, the machine learning nodes must wait for the feature extraction nodes which has not finished their tasks yet. In this stage, the optimisation component uses the performance measurements to identify the nodes that cause the delay (CPU-consuming), and reallocates a higher hardware configuration to them from an appropriate cost model.

Figure 6.8(a) shows the CPU usage of three processing nodes for the feature extraction components (Basic, Colour, and Shape) using a standard hardware

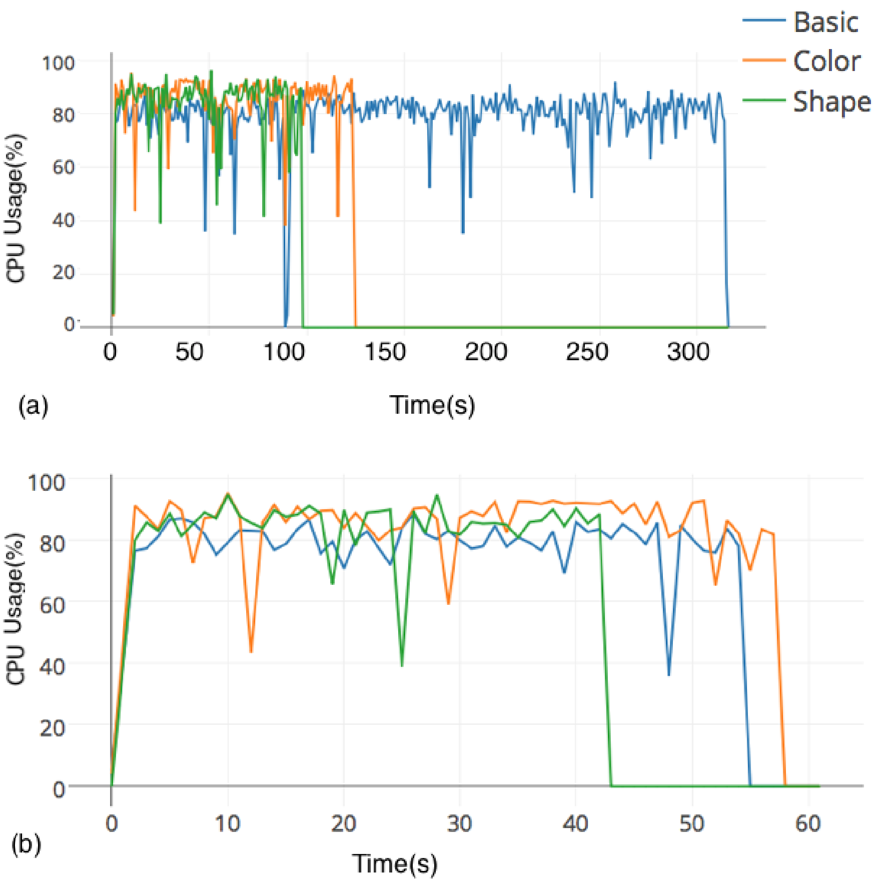


Figure 6.8: Compare CPU utilisation of the feature extraction from the image dataset with two configuration

configuration (1 vCPU, 4GB memory, and 20GB storage). Figure 6.8(b) presents the CPU performance of the same task but with the identified best-matched hardware configuration for that specific dataset. In this configuration, a node with 4 vCPU, 8GB memory and the 80GB disk is used for the Basic feature extraction node (which was identified as a node causing a delay). For the Colour and Shape feature extraction algorithms, two upgraded nodes, each of which has 2 vCPU, 4GB memory and 40GB disk, are used. Figure 6.8(b) shows the performance improvement of the feature extraction stage in comparison to Figure 6.8(a).

Table 6.4 presents a detailed comparison of two experiments including the costs for the one uses the standard configuration and the other ones use the best-matched configuration. Firstly, the feature extraction nodes were identified as CPU-intensive nodes, so the new configuration is selected from the Compute-Optimised category (refer to Table 6.1). Among the processing nodes, the Basic

Table 6.4: Comparing Standard configuration and Compute-Optimised configuration for image feature extraction nodes

Feature	Cost($)/per Hour	VM Type	Execution Time(s)	Cost($)

Basic	m3.medium(0.07)
	c3.Xlarge(0.21)
		Standard
		High CPU
			309
			57
				0.0060
				0.0033

Colour	m3.medium(0.07)
	c3.Large(0.11)
		Standard
		High CPU
			140
			58.8
				0.0023
				0.0017

Shape	m3.medium(0.07)
	c3.Large(0.11)
		Standard
		High CPU
			110
			42
				0.0019
				0.0012

feature extraction node required the most upgrading, so it is assigned c3.Xlarge VM type. The other two feature extractors are assigned c3.Large VM type. As presented in Table 6.4, based on the reduced execution time (i.e. Basic feature extraction is reduced from 309.6s to 57s), the cost is also reduced from 0.0060$ to 0.0033$. This result implies that the new configuration is able not only to improve the performance but also to reduce the cost.

Table 6.5: Comparing Standard configuration and Compute-Optimised configuration for model creation nodes

Models	Cost($)/per Hour	VM Type	Execution Time(s)	Cost($)

M1	m3.medium(0.07)
	c3.Large(0.11)
		Standard
		High CPU
			84.6
			40.2
				0.0016
				0.0012

M2	m3.medium(0.07)
	c3.XLarge(0.21)
		Standard
		High CPU
			169.8
			33.6
				0.0033
				0.0019

M3	m3.medium(0.07)
	c3.XLarge(0.21)
		Standard
		High CPU
			213
			28.8
				0.0041
				0.0016

M4	m3.medium(0.07)
	c3.Large(0.11)
		Standard
		High CPU
			102
			46.8
				0.0019
				0.0014

Table 6.5 presents a detailed comparison model creation of the execution time and the cost for nodes with the standard hardware configuration and the upgraded configuration. In this experiment, first, the model creation nodes are assigns from Compute-Optimised category. Then, among the processing nodes, the M2 and M3 model creation nodes require the most upgrading, so they are assigned the c3.Xlarge VM type (4 vCPU and 8GB memory). The other two model creation nodes (M1 and M4) are assigned c3.Large VM type (2 vCPU and 4GB memory). The results in Table 6.5 show that, based on the reduced execution time (i.e. for M2, it reduced from 169.8s to 33.6s), the cost is also reduced from 0.0033$ to 0.0019$. This outcome demonstrates that the new

configuration for the model creation stage can both enhance efficiency and result in cost reductions.

6.4 Conclusion

In this chapter, a performance-aware cost-efficiency optimisation component is proposed. A case study was designed to show the usefulness of the proposed performance-aware cost-efficiency optimisation component in two different scenarios. The CLOAM framework was used in its Auto-training mode to process an image dataset (Scenario A) and a text dataset (Scenario B). Analysing the resource consumption (CPU and memory) identified bottleneck and provided the basis for deciding where the VM configuration requires an upgrade. We demonstrated that in these two scenarios, components needed to be upgraded differently. For instance, in processing the image dataset, the feature extraction stage required a new configuration with more powerful virtual machines, as proposed by the optimisation component. The results showed that using the suggested configuration could both reduce the cost and also improve the overall performance. On the other hand, in processing the text dataset, the feature extraction stage did not require a new configuration; however, in the machine learning stage, a new configuration was suggested. For this stage, it is shown that using the new configuration, not only is the performance of the machine learning stage improved, but also the cost is reduced. In conclusion, the proposed optimisation component is evaluated, and the results show that the cost can be significantly reduced along with an improvement in performance.

Chapter 7

Conclusion

Big data analytics especially for making predictions and providing suggestions based on data with minimal human intervention, is challenging due to the need to process large amounts of data using machine learning algorithms [KNV$^+$18]. Using machine learning over big data and learning from vast and unstructured data brings significant opportunities to various areas (e.g. pathology and phytology).

Current intelligent machine-learning systems are performance driven, and their focus is on the predictive/classification accuracy, based on known properties learned from the training samples. The existing data analysis frameworks often depend on specialised hardware and algorithms and work for a specific application or data type. Hence, designing a more general and performance-driven machine learning framework, which provides auto-adjusting and adaptivity, is needed to help big data classification. This research set out with the goal of investigating and leveraging the state-of-the-art software and hardware technologies to mitigate big data analysis problems.

In seeking solutions, we conducted a literature review (Chapter 2) on current and emerging techniques and technologies in order to identify their suitability and potential for effective big data classification. We summarised our findings for classifying big data in a distributed environment (cloud environment) in parallel. Firstly, a generic cloud-based machine learning framework is required that provides multiple stages of segmentation, feature extraction, model building and classification. In each of these stages, several different software solutions can be integrated to improve and enhance the functionalities of the framework such as using advanced machine learning algorithms, employing various feature extraction techniques, and leveraging distributed big data processing frameworks. Optimisation components are also required to support optimising the performance, classification accuracy, use of resources, and cost.

In this chapter, we begin by summarising our contributions, followed by possible future work, and finally discussing the implications of the research.

7.1 Contributions

A generic machine learning architecture was designed and implemented, several optimisation components were designed to improve the efficiency of processing big data and the accuracy of machine learning models. The effectiveness of the framework was evaluated by real-world datasets. Figure 7.1 shows a comprehensive overview of the CLOAM framework that includes all the optimisation components.

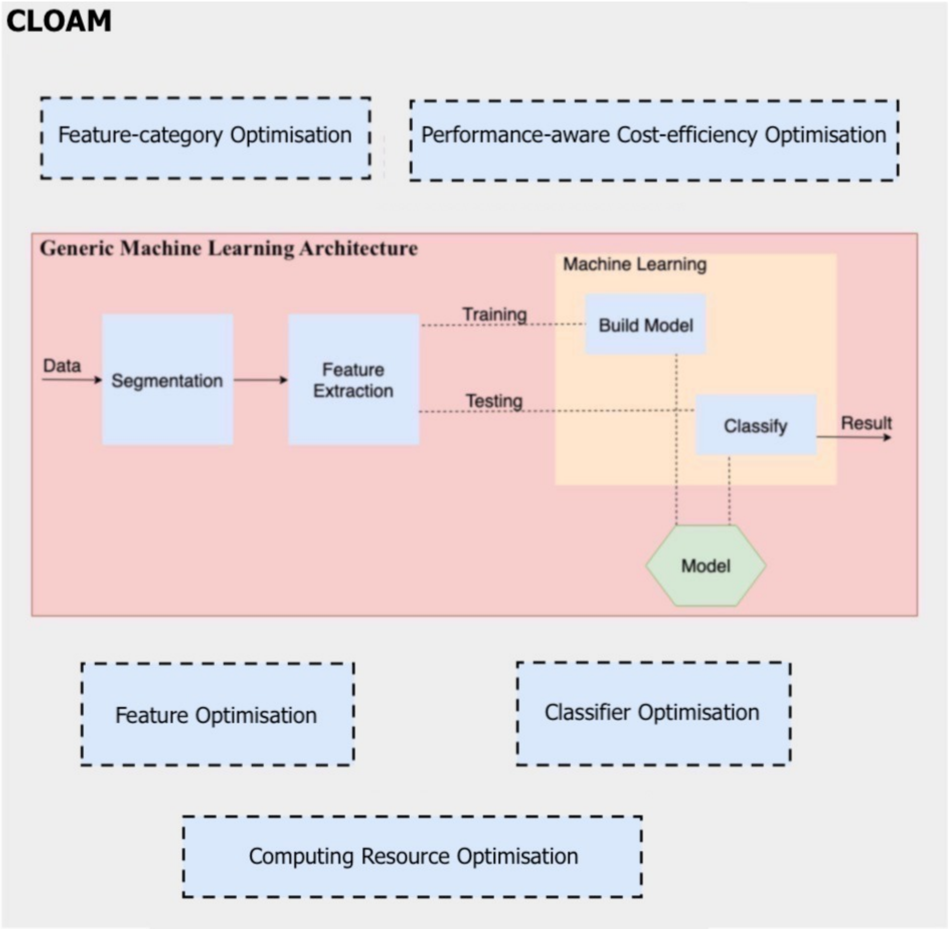

CLOAM
Feature-category Optimisation
Performance-aware Cost-efficiency Optimisation
Generic Machine Learning Architecture
Machine Learning
Data
Segmentation
Feature Extraction
Training
Testing
Build Model
Classify
Result
Model
Feature Optimisation
Classifier Optimisation
Computing Resource Optimisation

Figure 7.1: CLOAM framework

In this thesis, we presented several generic optimisation schemes having different properties suitable for a variety of big data analysis scenarios. This section is organised in such a way that we map the research questions set up in Chapter 1 to our contributions.

- **Research Question:** whether we can create a generic machine learning framework covering all stages of data pre-processing, model building and classification that can be used in distinct application domains and can support different types of unstructured data and different algorithms for machine learning.

  - **Our first contribution** is the design of the cloud-based framework that combines standard stages of data pre-processing, data segmentation, feature extraction, and machine learning that support training a model and classifying data (Chapter 3). It can be used for various application areas (e.g. medical and botany) and various machine learning algorithms (e.g. Naive Bayes and K-nearest neighbour). The generic architecture can support different data types (e.g. text and images) and datasets. It provides parallel processing in data segmentation, feature extraction, and machine learning stages. It is implemented using an asynchronous architecture incorporating queues, based on standard message formats for tasks and results. The architecture is technology invariant and can be implemented using any cloud software platform.

- **Research Question:** whether the proposed machine learning framework can incorporate generic optimisations that optimise feature selection (high-level and low-level) and ML algorithm selection so that they adjust for the application (e.g. pathology or phytology), and the specific dataset (e.g. a particular set of blood cell images or leaf images).

  - **Our second contribution** is the design of the feature-category optimisation component on top of the underlying architecture (Chapter 3). The optimisation component assists in selecting the most useful features for a specific ML algorithm and specific dataset. The best model selected is the one with a distinct combination of features (e.g. a combination of high-level features such as basic and colour features) that were most effective for one specific ML classifier (e.g. Decision Tree) to train a model that provides the highest accuracy for that specific dataset.

  - **Our third contribution** is the development of two optimisation components that support identifying the most useful features for classification and selecting the best algorithm under user-specified criteria (Chapter 4). The generic data-driven mechanisms used in developing these components provide sufficient information without processing large amounts of data and can auto-adjust for the particular dataset.

- **Research Question:** whether the proposed cloud-based framework can optimise its use of resources in the cloud, and can it also incorporate a cost model, allowing optimisation with regard to both efficiency and cost of cloud resources.

  - **Our fourth contribution** is the resource optimisation component incorporating monitoring the use of resources and then using scaling to leverage cloud computing resources in an efficient way (Chapter 5), and computing cost (Chapter 6). These optimisations auto-adjust for the data and the algorithms. The monitoring can indicate bottlenecks and allow a balanced, more efficient architecture to be achieved through appropriate scaling of resources. The cost-model of cloud resource offerings supports optimisation within an individual service provider (e.g. Amazon) or across service providers.

7.2 Future Work

The CLOAM framework is a generic framework with generic optimisation components, and this makes it ideal for extension and building future applications.

- **Supporting other unstructured data**

  The current framework uses some standard segmentation, and feature extraction services and the framework should be extended with services to deal with other media. Audio and video are important data sources for machine learning and new services for segmentation and feature extraction should be developed. Data pre-processing services for dealing with other useful sources, such as webpages and emails, could also be added to the framework.

- **Unsupervised machine learning**

  The focus of the current framework is supervised learning, but much of the functionality (e.g. for segmentation and feature extraction) is needed also for unsupervised learning, and the framework could be extended with new functionality supporting unsupervised data analysis methods, such as cluster analysis and linear regression. A number of problems can benefit from a hybrid of supervised and unsupervised learning, and the extended framework would also support this.

- **Security**

  The current framework does not deal directly with security, and this is an important issue to be addressed in the future. One aspect is the anonymisation of datasets involved in training and testing, and another is the encryption of sensitive information. In applications with multiple data sources (e.g. personal medical text and image datasets), some parts of the data may require security while others may not, and this can be easily accommodated by the task-oriented organisation of the framework.

- **Optimisation-as-a-service**

  The asynchronous message-passing architecture is mostly service-oriented, and the framework should be generalised as fully service-oriented architecture, offering standard services such as for data preprocessing, segmentation, feature extraction, and machine learning services. The generic optimisation components should also be offered as services with standard APIs to the other services so that they can then be used in generating samples, evaluating results, and providing optimisation feedback, i.e. offering a type of optimisation-as-a-service.

7.3 Summary

The CLOAM framework has been designed as a generic framework with generic optimisation components to support supervised learning over multiple datatypes. The general asynchronous message-passing architecture supports parallel processing of multiple datasets, and even datatypes, and provides a good basis for future extension. It provides a generic framework which can be quickly adapted to different problems. The optimisation components address, in a general manner, optimisation issues related to features, classifiers, computation resources and computation cost. The optimisations are independent of application and datatype, but are data-driven and auto-adjust for the particular dataset being processed. Evaluation has shown the benefits of these optimisations.

Many different solutions are needed to address the challenges of machine learning over big data, including innovative learning algorithms and more efficient and cost-effective computation resources. The CLOAM optimisations are complementary to these other advances and offer opportunities for additional improvement of solutions.

References

[AEAA15] A A L C Amarathunga, E P W C Ellawala, G N Abeysekara, and C R J Amalraj. Expert System For Diagnosis Of Skin Diseases. *International Journal of Scientific & Technology Research*, 4(01):174–178, 2015.

[Ale17] Katarina Elyamany Hany Capretz Miriam. Alexandra, Grolinger. Machine Learning with Big Data: Challenges and Approaches. *IEEE Access*, 5:7776–7797, 2017.

[ALK15] Ehab Nabiel Alkhanak, Sai Peck Lee, and Saif Ur Rehman Khan. Cost-aware challenges for workflow scheduling approaches in cloud computing environments: Taxonomy and opportunities. *Future Generation Computer Systems*, 50:3–21, sep 2015.

[Ama18] Amazon. Amazon, 2018. `https://aws.amazon.com/autoscaling/`, Accessed: 2019.02.27.

[Ans16] Daniel Ansari. Sentiment Polarity Classification Using Structural Features. In *Proceedings - 15th IEEE International Conference on Data Mining Workshop, ICDMW 2015*, pages 1270–1273. IEEE, nov 2016.

[AOO14] B.O Adegoke, B O Ola, and M.E Omotayo. Review of Feature Selection Methods in Medical Image Processing. *IOSR Journal of Engineering*, 04(01):1–5, 2014.

[ASHL18] Tri Dev Acharya, Anoj Subedi, He Huang, and Dong Ha Lee. Classification of Surface Water Using Machine Learning Methods from Landsat Data in Nepal. *Proceedings*, 4(1):43, nov 2018.

[ASNG10] S Arivazhagan, R Newlin Shebiah, S Selva Nidhyanandhan, and L Ganesan. Fruit Recognition using Color and Texture Features. *Information Sciences*, 1(2):90–94, 2010.

[AV17] Umme Aiman and Virendra P. Vishwakarma. Face recognition using modified deep learning neural network. In *8th International Conference on Computing, Communications and Networking Technologies, ICCCNT 2017*, pages 1–5. IEEE, jul 2017.

[Avr14] M.G. Avram. Advantages and Challenges of Adopting Cloud Computing from an Enterprise Perspective. *Procedia Technology*, 12:529–534, 2014.

[BGPCV12] M L Badger, T Grance, R Patt-Corner, and J Voas. Cloud computing synopsis and recommendations. Technical report, National Institute of Standards and Technology, Gaithersburg, MD, 2012.

[BIG18] BIGDATA. Bigdata, 2018. `https://analyticsweek.com/content/big-data-facts/`, Accessed: 2019.02.27.

[BKK+08] Eun-Kyu Byun, Jin-Soo Kim, Yang-Suk Kee, Ewa Deelman, Karan Vahi, and Gaurang Mehta. Efficient Resource Capacity Estimate of Workflow Applications for Provisioning Resources. *4th IEEE International Conference on e-Science (eScience'08), to appear*, 2008.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, April 1998.

[BRW+15] Peter Burnap, Omer Rana, Matthew Williams, William Housley, Adam Edwards, Jeffrey Morgan, Luke Sloan, and Javier Conejero. COSMOS: Towards an integrated and scalable service for analysing social media on demand. *International Journal of Parallel, Emergent and Distributed Systems*, 30(2):80–100, 2015.

[BV16] Anatoliy Batyuk and Volodymyr Voityshyn. Apache storm based on topology for real-time processing of streaming data from social networks. In *Proceedings of the 2016 IEEE 1st International Conference on Data Stream Mining and Processing, DSMP 2016*, pages 345–349. IEEE, aug 2016.

[CCD+12] Nicolò M Calcavecchia, Bogdan A Caprarescu, Elisabetta Di Nitto, Daniel J Dubois, and Dana Petcu. DEPAS: A decentralized probabilistic algorithm for auto-scaling. *Computing*, 94(8-10):701–730, 2012.

[Cha14] Goutam Chakraborty. Analysis of Unstructured Data : Applications of Text Analytics and Sentiment Mining. In *SAS Global Forum, Washington D.C.*, page 15, 2014.

[CHH+17] Min Chen, Yixue Hao, Kai Hwang, Lin Wang, and Lu Wang. Disease Prediction by Machine Learning over Big Data from Healthcare Communities. *IEEE Access*, 5:8869–8879, 2017.

[CJ15] Jincy B Chrystal and Stephy Joseph. Text Mining and Classification of Product Reviews Using Structured Support Vector Machine. *The International Journal of Multimedia & Its Applications (IJMA)*, 5(4):21–31, 2015.

[CR11] Jyotismita Chaki and Parekh Ranjan. Plant Leaf Recognition using Shape based Features and Neural Network classifiers. *IJACSA) International Journal of Advanced Computer Science and Applications*, 2(10):41–47, 2011.

[CU16] Dr. Amit Chaturvedi and Malay Upadhyay. A Review of Dynamic Resource Scaling applications in the Cloud Computing. *International Journal of Computer Trends and Technology*, 33(2):47–52, 2016.

[Cza14] Paweł Czarnul. A workflow application for parallel processing of big data from an Internet portal. In *Procedia Computer Science*, volume 29, pages 499–508, 2014.

[Dau18] Distefano Salvatore Dautov, Rustem. Quantifying volume, velocity, and variety to support (Big) data-intensive application development. In *Proceedings - 2017 IEEE International Conference on Big Data, Big Data 2017*, volume 2018-Janua, pages 2843–2852. IEEE, dec 2018.

[Dee08] Ewa Deelman. The cost of doing scince on the cloud: The montage example. *International Conference for High Perofrmance Computing, Networking, Storage and Analysis, 2008.pp.1-12*, 2008.

[DG11] A Kanaka Durga and A Govardhan. Ontology Based Text Categorization - Telugu Documents. *International Journal of Scientific & Engineering Research*, 2(9):1–4, 2011.

[DGG15] Aparna Datt, Anita Goel, and S. C. Gupta. Monitoring list for compute infrastructure in eucalyptus cloud. In *Proceedings - 2015*

IEEE 24th International Conference on Enabling Technologies: Infrastructures for Collaborative Enterprises, WETICE 2015, pages 69–71. IEEE, jun 2015.

[DGVV12] Sourav Dutta, Sankalp Gera, Akshat Verma, and Balaji Viswanathan. SmartScale: Automatic application scaling in enterprise clouds. In *Proceedings - 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012*, pages 221–228. IEEE, jun 2012.

[Doh08] Blighe Michael Smeaton Alan O'Connor. Doherty, Aiden. Combining image descriptors to effectively retrieve events from visual lifelogs. In *Centre for Digital Video Processing and CLARITY: Centre for Sensor Web Technologies*, page 10, 2008.

[DTM12] Wesam Dawoud, Ibrahim Takouna, and Christoph Meinel. Elastic Virtual Machine for Fine-grained Cloud Resource Provisioning. Technical report, Communications in Computer and Information Science, 2012.

[DY14] Kaur Dilpreet and Kaur Yadwinder. Various image segmentation techniques: A review. *International Journal of Computer Science and Mobile Computing*, 3(5):809–814, 2014.

[DZDW19] Happiness Ugochi Dike, Yimin Zhou, Kranthi Kumar Deveerasetty, and Qingtian Wu. Unsupervised Learning Based On Artificial Neural Network: A Review. In *2018 IEEE International Conference on Cyborg and Bionic Systems, CBS 2018*, pages 322–327. IEEE, oct 2019.

[EDM19] Abderrahmane Ed-Daoudy and Khalil Maalmi. Real-time machine learning for early detection of heart disease using big data approach. In *2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*, pages 1–5. IEEE, apr 2019.

[ES06] Andrea Esuli and Fabrizio Sebastiani. SENTIWORDNET: A Publicly Available Lexical Resource for Opinion Mining. In *In Proceedings of the 5th Conference on Language Resources and Evaluation LREC 06*, pages 417–422, 2006.

[Eva15] Robert Evans. Apache Storm, a Hands on Tutorial. In *2015 IEEE International Conference on Cloud Engineering*, pages 2–2. IEEE, mar 2015.

[EVA18] G. M. Evgenyevich, B. A. Valerievich, and B. M. Alekseevna. Using apache spark to collect analytic from the streaming data processing application logs. In *2018 7th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–4, June 2018.

[FBG+17] Lex Fridman, Daniel E. Brown, Michael Glazer, William Angell, Spencer Dodd, Benedikt Jenik, Jack Terwilliger, Aleksandr Patsekin, Julia Kindelsberger, Li Ding, Sean Seaman, Alea Mehler, Andrew Sipperley, Anthony Pettinato, Bobbie Seppelt, Linda Angell, Bruce Mehler, and Bryan Reimer. MIT Autonomous Vehicle Technology Study: Large-Scale Deep Learning Based Analysis of Driver Behavior and Interaction with Automation. *arXiv preprint arXiv:1711.06976*, nov 2017.

[FdRL+14] Alberto Fernández, Sara del Río, Victoria López, Abdullah Bawakid, María J. del Jesus, José M Benítez, and Francisco Herrera. Big Data with Cloud Computing: An insight on the computing environment, MapReduce, and programming frameworks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(5):380–409, 2014.

[FF17] Joao Pedro de Magalhaes Fabio Fabris, Alex A. Freitas. A review of supervised machine learning applied to ageing research, 2017.

[FLRC14] Stefan Frey, Claudia Lüthje, Christoph Reich, and Nathan Clarke. Cloud QoS scaling by fuzzy logic. In *Proceedings - 2014 IEEE International Conference on Cloud Engineering, IC2E 2014*, pages 343–348. IEEE, mar 2014.

[FRMT18] Hajar Filali, Jamal Riffi, Adnane Mohamed Mahraz, and Hamid Tairi. Multiple face detection based on machine learning. In *2018 International Conference on Intelligent Systems and Computer Vision, ISCV 2018*, volume 2018-May, pages 1–8. IEEE, apr 2018.

[GDK+14] Anshul Gandhi, Parijat Dube, Alexei Karve, Andrzej Kochut, and Li Zhang. Modeling the impact of workload on cloud resource scaling. In *Proceedings - Symposium on Computer Architecture and High Performance Computing*, pages 310–317, 2014.

[GEN13] Yang Zhi GENG, Xia. Data Mining in Cloud Computing. In *Proceedings of the 2013 International Conference on Information Science and Computer Applications (ISCA 2013)*, Paris, France, 2013. Atlantis Press.

[GFK$^+$13] Matthias Gander, Michael Felderer, Basel Katt, Adrian Tolbaru, Ruth Breu, and Alessandro Moschitti. Anomaly Detection in the Cloud: Detecting Security Incidents via Machine Learning. In *Communications in Computer and Information Science*, volume 379 CCIS, pages 103–116, 2013.

[Gho11] Krishnamurthy Rajasekar Pednault Edwin Reinwald Berthold Sindhwani Vikas Tatikonda Shirish Tian Yuanyuan Vaithyanathan Shivakumar Ghoting, Amol. SystemML: Declarative machine learning on MapReduce. In *Proceedings - International Conference on Data Engineering*, pages 231–242, 2011.

[Gom15] Apurva Gomashe. A novel approach of color histogram based image search/retrieval. Technical Report 6, Journal of Computer Science and Mobile Computing, 2015.

[Goo19] Google. Google, 2019. `https://cloud.google.com/compute/`, Accessed: 2019.02.27.

[Gou18] Georgios Gousios. Big data software analytics with apache spark. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*, pages 542–543, May 2018.

[GPP18] Swarna Latha Garapati, Andhra Pradesh, and Andhra Pradesh. Application of Big Data Analytics: An Innovation in Health Care. *International Journal of Computational Intelligence Research*, 14(1):15–27, 2018.

[GSM$^+$11] Kevin Gimpel, Nathan Schneider, Daniel Mills, Brendan O'Connor, Dipanjan Das, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, Noah A Smith, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 42–47, 2011.

[Hak18] Özcan Caner Oğul Iskender Ülgen Hakdağli, Özlem. Stream text data analysis on twitter using apache spark streaming. In *26th IEEE Signal Processing and Communications Applications Conference, SIU 2018*, pages 1–4, 2018.

[HEM17] Sohrabi Mohammad Karim HEMmatian, Fatemeh. A survey on classification techniques for opinion mining and sentiment analysis-Springer Science+Business Media B.V., part of Springer Nature 2017. *Artificial Intelligence Review*, dec 2017.

[HGGG12] Rui Han, Li Guo, Moustafa M. Ghanem, and Yike Guo. Lightweight resource scaling for cloud applications. In *Proceedings - 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012*, pages 644–651. IEEE, may 2012.

[HH12] Zhiyi MaXiang Li Huihong He. An approach to estimating cost of running cloud applications based on aws, ser. *19thAsia-PacificSoftwareEngineeringConference.*, 2012.

[HLS^{+}11] Michael Hogan, Fang Liu, Annie Sokol, Jin Tong, Gary Locke, and Patrick D Gallagher. NIST Cloud Computing Standards Roadmap. *NIST Cloud Computing Standards*, 2011.

[HRY16] Gang He, Shijie Ruan, and Dechen Yu. Implementation of a Chinese word segmentation in big data environment. In *Proceedings - 2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics, IHMSC 2016*, volume 2, pages 34–38. IEEE, aug 2016.

[HS88] Chris Harris and Mike Stephens. A Combined Corner and Edge Detector. In *Procedings of the Alvey Vision Conference 1988*, pages 23.1–23.6, 1988.

[IMCD16] Salam Ismaeel, Ali Miri, Dharmendra Chourishi, and S. M. Reza Dibaj. Open Source Cloud Management Platforms: A Review. In *Proceedings - 2nd IEEE International Conference on Cyber Security and Cloud Computing, CSCloud 2015 - IEEE International Symposium of Smart Cloud, IEEE SSC 2015*, pages 470–475. IEEE, nov 2016.

[Ing12] Vaishali A. Ingle. Processing of unstructured data for information extraction. In *3rd Nirma University International Conference on Engineering, NUiCONE 2012*, pages 1–4. IEEE, dec 2012.

[JH15] Ammar Hadishaheed Sarah Haji Ahmad Azizahbt Jasim Hadi, Hameed Shnain. Big Data and Five V'S Characteristics. *International Journal of Advances in Electronics and Computer Science*, (2):2393–2835, 2015.

[JJZ+17] Fei Jiang, Yong Jiang, Hui Zhi, Yi Dong, Hao Li, Sufeng Ma, Yilong Wang, Qiang Dong, Haipeng Shen, and Yongjun Wang. Artificial intelligence in healthcare: Past, present and future, dec 2017.

[Jo10] Taeho Jo. NTC (Neural Text Categorizer): Neural Network for Text Categorization. *International Journal of Information Studies*, 2(2):83–96, 2010.

[JPM16] Pooyan Jamshidi, Claus Pahl, and Nabor C. Mendonca. Managing Uncertainty in Autonomic Cloud Elasticity Controllers. *IEEE Cloud Computing*, 3(3):50–60, 2016.

[kag16] kaggle. kaggle, 2016. `https://www.kaggle.com`, Accessed: 2019.02.27.

[KC03] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, 36(1), 2003.

[KC14] Pankaj Deep Kaur and Inderveer Chana. A resource elasticity framework for QoS-aware execution of cloud applications. *Future Generation Computer Systems*, 37:14–25, jul 2014.

[KDn18] KDnuggets. Kdnuggets, 2018. `https://www.kdnuggets.com/polls/2014/languages-analytics-data-mining-data-science.html`, Accessed: 2019.05.27.

[KGC+18] Gaganjot Kaur Kang, Jerry Zeyu Gao, Sen Chiao, Shengqiang Lu, and Gang Xie. Air Quality Prediction: Big Data and Machine Learning Approaches. *International Journal of Environmental Science and Development*, 9(1):8–16, 2018.

[KGL+18] Ugur Kursuncu, Manas Gaur, Usha Lokala, Krishnaprasad Thirunarayan, Amit Sheth, and I. Budak Arpinar. Predictive Analysis on Twitter: Techniques and Applications. *Emerging Re-*

search Challenges and Opportunities in Computational Social Network Analysis, 2018.

[KNV$^+$18] M. G. Kibria, K. Nguyen, G. P. Villardi, O. Zhao, K. Ishizu, and F. Kojima. Big data analytics, machine learning, and artificial intelligence in next-generation wireless networks. *IEEE Access*, 6:32328–32338, 2018.

[KS18] T Karthikayini and N. K. Srinath. Comparative Polarity Analysis on Amazon Product Reviews Using Existing Machine Learning Algorithms. In *2nd International Conference on Computational Systems and Information Technology for Sustainable Solutions, CSITSS 2017*, pages 1–6. IEEE, dec 2018.

[KT12] Fanyu Kong and Jindong Tan. DietCam: Automatic dietary assessment with mobile camera phones. *Pervasive and Mobile Computing*, 8(1):147–163, feb 2012.

[KTC14] Gang-Hoon Kim, Silvana Trimi, and Ji-Hyong Chung. Big-data applications in the government sector. *Communications of the ACM*, 57(3):78–85, 2014.

[KWQH17] In Kee Kim, Wei Wang, Yanjun Qi, and Marty Humphrey. Empirical evaluation of workload forecasting techniques for predictive cloud resource scaling. In *IEEE International Conference on Cloud Computing, CLOUD*, pages 1–10, 2017.

[KYM$^+$18] Qusay Kanaan Kadhim, Robiah Yusof, Hamid Sadeq Mahdi, Sayed Samer Ali Al-Shami, and Siti Rahayu Selamat. A Review Study on Cloud Computing Issues. In *Journal of Physics: Conference Series*, volume 1018, page 12006, 2018.

[KZB04] Timor Kadir, Andrew Zisserman, and Michael Brady. An affine invariant salient region detector. *Image Rochester NY*, 3021(6):228–241, 2004.

[LCT18] C. Liao, R. Chen, and S. Tai. Emotion stress detection using eeg signal and deep learning technologies. In *2018 IEEE International Conference on Applied System Invention (ICASI)*, pages 90–93, April 2018.

[Lef15] Daniel Richard Leff. Big Data for Precision Medicine, 2015.

[Lin02] Joakim Lindblad. *Development of Algorithms for Digital Image Cytometry*. PhD thesis, Faculty of Science and Technology, 2002.

[Liz14] Dr.M.Sasikumar Lizy Abraham. Detection of Bridges using Different Types of High Resolution Satellite Images. *Int. J. on Recent Trends in Engineering and Technology,*, 10(2):10, 2014.

[LKK$^+$14] Jihyun Lee, Jinmee Kim, Dong-Jae Kang, Namwoo Kim, and Sungin Jung. Cloud service broker portal: Main entry point for multi-cloud service providers and consumers. In *16th International Conference on Advanced Communication Technology*, pages 1108–1112. Global IT Research Institute (GIRI), feb 2014.

[LLW18] Shaoyu Lu, Sina Lin, and Beibei Wang. Recognition and Classification of Fast Food Images. *Global Journal of Computer Science and Technology*, 18, 2018.

[LPS11] Ruggero Donida Labati, Vincenzo Piuri, and Fabio Scotti. All-IDB: The acute lymphoblastic leukemia image database for image processing. In *2011 18th IEEE International Conference on Image Processing*, pages 2045–2048. IEEE, sep 2011.

[LZ10] Palden Lama and Xiaobo Zhou. Autonomic provisioning with self-adaptive neural fuzzy control for end-to-end delay guarantee. In *Proceedings - 18th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS 2010*, pages 151–160. IEEE, aug 2010.

[Mad12] Sam Madden. From databases to big data. *IEEE Internet Computing*, 16(3):4–6, May 2012.

[Mal11] Jyoti Malik. Harris Operator Corner Detection using Sliding Window Method. *International Journal of Computer Applications*, 22(1):28–37, 2011.

[Mal18] Mohammad Ilyas Malik. CLOUD COMPUTING-TECHNOLOGIES. *International Journal of Advanced Research in Computer Science*, 9(2):379–384, apr 2018.

[MASL18] Ismail Yaqub Maolood, Yahya Eneid Abdulridha Al-Salhi, and Songfeng Lu. Thresholding for medical image segmentation for cancer using fuzzy entropy with level set algorithm. *Open Medicine*, 13(1):374–383, 2018.

[McB05] James R McBride. An architectural blueprint for autonomic computing. Technical Report 12, IBM, 2005.

[MGS12] Hiroya Matsuura, Masaru Ganse, and Toyotaro Suzumura. A highly efficient consolidated platform for stream computing and Hadoop. In *Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2012*, pages 2026–2034. IEEE, may 2012.

[Mis18] Misra Anuranjan Mishra, Suyash. Structured and Unstructured Big Data Analytics. In *International Conference on Current Trends in Computer, Electrical, Electronics and Communication, CTCEEC 2017*, pages 740–746. IEEE, sep 2018.

[Moh12] Mehryar. Mohri. Foundations of Machine Learning - Book. *The MIT Press*, page 414, 2012.

[MOH17] MOHamed. Comparative Study of Four Supervised Machine Learning Techniques for Classification. *International Journal of Applied Science and Technology*, 7(2):5–18, 2017.

[MS16] Jaison Paul Mullerikkal and Yedhu Sastri. A Comparative Study of OpenStack and CloudStack. In *Proceedings - 2015 5th International Conference on Advances in Computing and Communications, ICACC 2015*, pages 81–84. IEEE, sep 2016.

[MSBA14] Rajat Mehrotra, Srishti Srivastava, Ioana Banicescu, and Sherif Abdelwahed. An Interaction Balance Based Approach for Autonomic Performance Management in a Cloud Computing Environment. In *. . . and Scheduling for . . . .* Published in ARMS-CC@PODC 2014, 2014.

[MSG13] P Mohanaiah, P Sathyanarayana, and L Gurukumar. Image Texture Feature Extraction Using GLCM Approach. *International Journal of Scientific and Research Publications*, 3(1):2250–3153, 2013.

[MSR$^+$18] R. Meena Prakash, G. P. Saraswathy, G. Ramalakshmi, K. H. Mangaleswari, and T. Kaviya. Detection of leaf diseases and classification using digital image processing. In *Proceedings of 2017 International Conference on Innovations in Information, Embedded and Communication Systems, ICIIECS 2017*, volume 2018-Janua, pages 1–4. IEEE, mar 2018.

[NCCA15] Marco a S Netto, Carlos Cardonha, Renato L F Cunha, and Marcos D. Assuncao. Evaluating auto-scaling strategies for cloud computing environments. *Proceedings - IEEE Computer Society's Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, MASCOTS*, 2015-Febru(February):187–196, 2015.

[NLC15] Bernardo Pereira Nunes, Giseli Rabello Lopes, and Marco Antonio Casanova. Automatic classification and taxonomy generation for semi-structured data. In *Proceedings - 15th IEEE International Conference on Computer and Information Technology, CIT 2015, 14th IEEE International Conference on Ubiquitous Computing and Communications, IUCC 2015, 13th IEEE International Conference on Dependable, Autonomic and Se*, pages 207–214. IEEE, oct 2015.

[NRRD12] Akshay Narayan, Shrisha Rao, Gaurav Ranjan, and Kumar Dheenadayalan. Smart metering of cloud services. In *6th Annual IEEE International Systems Conference (IEEE SysCon 2012)*, Vancouver, BC, Canada, March 2012. doi:10.1109/SysCon.2012.6189462.

[NSA$^+$19] Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, and Khaled Shaalan. Speech Recognition Using Deep Neural Networks: A Systematic Review. *IEEE Access*, 7:19143–19165, 2019.

[NT16] Mitisha Narottambhai and Purvi Tandel. A Survey on Feature Extraction Techniques for Shape based Object Recognition. *International Journal of Computer Applications*, 137(6):16–20, 2016.

[PBA$^+$08] Daryl C Plummer, Thomas J Bittman, Tom Austin, David W Cearley, and David Mitchell Smith. Cloud Computing : Defining and Describing an Emerging Phenomenon. *Analysis*, 51(June):1–9, 2008.

[PH15] Rezvan Pakdel and John Herbert. A cloud-based data analysis framework for object recognition. In *CLOSER 2015 - Proceedings of the 5th International Conference on Cloud Computing and Services Science, Lisbon, Portugal, 20-22 May, 2015.*, pages 79–86, 2015.

[PH16a] Rezvan Pakdel and John Herbert. Efficient Cloud-Based Framework for Big Data Classification. In *Proceedings - 2016 IEEE 2nd International Conference on Big Data Computing Service and Ap-*

plications, BigDataService 2016, pages 195–201, Oxford, UK, mar 2016. IEEE.

[PH16b] Rezvan Pakdel and John Herbert. Scalable Cloud-Based Analysis Framework for Medical Big-Data. In *Proceedings - International Computer Software and Applications Conference*, volume 2, pages 647–652, Atlanta, GA, USA, jun 2016. IEEE.

[PH17] Rezvan Pakdel and John Herbert. Adaptive Cost Efficient Framework for Cloud-Based Machine Learning. In *Proceedings - International Computer Software and Applications Conference*, volume 2, pages 155–160, Turin, Italy, jul 2017. IEEE.

[PHS$^+$09] Pradeep Padala, Kai-Yuan Hou, Kang G. Shin, Xiaoyun Zhu, Mustafa Uysal, Zhikui Wang, Sharad Singhal, and Arif Merchant. Automated control of multiple virtualized resources. In *Proceedings of the fourth ACM european conference on Computer systems - EuroSys '09*, page 13, New York, New York, USA, 2009. ACM Press.

[PIP16] Daniel Pop, Gabriel Iuhasz, and Dana Petcu. Distributed platforms and cloud services: Enabling machine learning for big data. In *Data Science and Big Data Computing: Frameworks and Methodologies*, pages 139–159. Springer, 2016.

[PY11] Neelamma K Patil and Ravi M Yadahalli. Comparison between HSV and YCbCr Color Model Color-Texture based Classification of the Food Grains. *International Journal of Computer Applications*, 34(4):51–57, 2011.

[R10] Zhang Q Cheng L Boutaba R. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 2010.

[Rac19] Rackspace. Rackspace, 2019. `http://www.rackspace.com/`, Accessed: 2019.02.27.

[RAG14] Wullianallur RAGHUPATHI. Big data analytics in healthcare: promise and potential. *Health Information Science and Systems*, 2(1), December 2014.

[Rig18] RightScale. Rightscale, 2018. `http://rightscale.com`, Accessed: 2019.02.27.

[RKS13] Rashmi, Mukesh Kumar, and Rohini Saxena. Algorithm and Technique on Various Edge Detection : A Survey. *Signal & Image Processing : An International Journal*, 4(3):65–75, 2013.

[RMV$^+$18] Megha Rathi, Aditya Malik, Daksh Varshney, Rachita Sharma, and Sarthak Mendiratta. Sentiment Analysis of Tweets Using Machine Learning Approach. In *2018 11th International Conference on Contemporary Computing, IC3 2018*, pages 1–3. IEEE, aug 2018.

[RSG$^+$14] Matthias Reif, Faisal Shafait, Markus Goldstein, Thomas Breuel, and Andreas Dengel. Automatic classifier selection for non-experts. *Pattern Analysis and Applications*, 17(1):83–96, feb 2014.

[SAAB$^+$18] Shashank Shekhar, Hamzah Abdel-Aziz, Anirban Bhattacharjee, Aniruddha Gokhale, and Xenofon Koutsoukos. Performance Interference-Aware Vertical Elasticity for Cloud-Hosted Latency-Sensitive Applications. In *IEEE International Conference on Cloud Computing, CLOUD*, volume 2018-July, pages 82–89. IEEE, jul 2018.

[SAB07] SABINE. Color Spaces, Color Encodings, and Color Image Encodings. pages 394–398. Published in IS&T/SPIE Electronic Imaging 2001, 2007.

[SAE12] Omar Sefraoui, Mohammed Aissaoui, and Mohsine Eleuldj. OpenStack: Toward an Open-Source Solution for Cloud Computing. *International Journal of Computer Applications*, 55(03):38–42, 2012.

[SD15] Omid Sarrafzadeh and Alireza Mehri Dehnavi. Nucleus and cytoplasm segmentation in microscopic images using K-means clustering and region growing. *Advanced biomedical research*, 4:174, 2015.

[SH17] Aishwarya Soni and Muzammil Hasan. Pricing schemes in cloud computing:a review. *International Journal of Advanced Computer Research*, 7(29):2277–7970, 2017.

[SHA05] Blake Andrew SHARIFZADEH, Sara. Contour-based learning for object detection. *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, 2005.

[SHA08] Centre Epsrc Manufacturing Innovative SHARIFZADEH, Sara. Edge detection techniques : Evaluations and comparisons Edge Detection Techniques : Evaluations and Comparisons. *Applied Mathematical Sciences*, 2(May 2016):1507–1520, 2008.

[SHL12] Ya-Yunn Su Siew Huei Liew. Cloudguide: Helping users estimate cloud deployment cost and performance for legacy web applications, ser. *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings.*, 2012.

[Sim17] Eric Simmon. Draft-Evaluation of Cloud Computing Services Based on NIST 800-145. *Evaluation of Cloud Computing Services Based on NIST*, pages 1–24, 2017.

[SK18] Ashwini T Sapkal and Uday V Kulkarni. Comparative study of Leaf Disease Diagnosis system using AUTOMATIC LEAF DISEASE DETECTION. *International Journal of Applied Engineering Research*, 13(19):14334–14340, 2018.

[SKIW17] Uthayasankar Sivarajah, Muhammad Mustafa Kamal, Zahir Irani, and Vishanth Weerakkody. Critical analysis of Big Data challenges and analytical methods. *Journal of Business Research*, 70:263–286, jan 2017.

[SKN17] Dixa Saxena, S. K., and K. N. Survey Paper on Feature Extraction Methods in Text Categorization. *International Journal of Computer Applications*, 166(11):11–17, 2017.

[SKZ$^+$14] Simon Spinner, Samuel Kounev, Xiaoyun Zhu, Lei Lu, Mustafa Uysal, Anne Holler, and Rean Griffith. Runtime Vertical Scaling of Virtualized Applications via Online Model Estimation. In *International Conference on Self-Adaptive and Self-Organizing Systems, SASO*, pages 157–166. IEEE, sep 2014.

[SM17] Soumen Santra and Kalyani Mali. Pixel variation problem identification in image segmentation for big image data set in cloud platform. In *2016 IEEE International Conference on Advances in Computer Applications, ICACA 2016*, pages 318–320. IEEE, oct 2017.

[SP11] Parnia Samimi and Ahmed Patel. Review of pricing models for grid & cloud computing. *2011 IEEE Symposium on Computers & Informatics*, pages 634–639, 2011.

[SRB19] Nida Shahid, Tim Rappon, and Whitney Berta. Applications of artificial neural networks in health care organizational decision-making: A scoping review, 2019.

[SRMG18] SA Senthilkumar, Bharatendara K Rai, Amruta A Meshram, and Angappa Gunasekaran. Big Data in Healthcare Management: A Review of Literature. *American Journal of Theoretical and Applied Business*, 4(2):57–69, 2018.

[SS17] Veda C. Storey and Il-Yeol Song. Big data technologies and management. *Data Knowl. Eng.*, 108(C):50–67, March 2017.

[SSa11] Lt SSanthosh Baboo. Image Retrieval using Harris Corners and Histogram of Oriented Gradients. *International Journal of Computer Applications*, 24(7):975–8887, 2011.

[SSCS10] Rahul Singh, Upendra Sharma, Emmanuel Cecchet, and Prashant Shenoy. Autonomic mix-aware provisioning for non-stationary data center workloads. In *Proceeding of the 7th international conference on Autonomic computing - ICAC '10*, page 21, 2010.

[SSSS11] Upendra Sharma, Prashant Shenoy, Sambit Sahu, and Anees Shaikh. A cost-aware elasticity provisioning system for the cloud. In *Proceedings - International Conference on Distributed Computing Systems*, pages 559–570. IEEE, jun 2011.

[SYWJ17] B. Shangguan, P. Yue, Z. Wu, and L. Jiang. Big spatial data processing with apache spark. In *2017 6th International Conference on Agro-Geoinformatics*, pages 1–4, Aug 2017.

[SZ13] Peng Lang Shui and Wei Chuan Zhang. Corner detection and classification using anisotropic directional derivative representations. *IEEE Transactions on Image Processing*, 22(8):3204–3218, aug 2013.

[TD10] Hong Linh Truong and Schahram Dustdar. Composable cost estimation and monitoring for computational applications in cloud computing environments. In *Procedia Computer Science*, volume 1, pages 2175–2184, 2010.

[VC15] Sridhar Vemula and Christopher Crick. Hadoop image processing framework. In *2015 IEEE International Congress on Big Data*, pages 506–513, 2015.

[VDWF12] Gregor Von Laszewski, Javier Diaz, Fugang Wang, and Geoffrey C. Fox. Comparison of multiple cloud frameworks. In *Proceedings - 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012*, pages 734–741. IEEE, jun 2012.

[Vin92] Luc Vincent. Morphological {{Area Openings}} and {{Closings}} for {{Grey}}-{{Scale Images}}. In {{*NATO Shape*}} *in* {{*Picture Workshop*}}, pages 197–208. Springer-Verlag, 1992.

[Vu17] Linh Vu. A lexicon-based method for Sentiment Analysis using social network data. In *Int'l Conf. Information and Knowledge Engineering*, pages 10–16, 2017.

[Wan10] Sowell Benjamin-Wang Xun Wang, Salles. Behavioral simulations in mapreduce. In *Proceedings of the VLDB Endowment*, volume 3, pages 952–963, 2010.

[Wek15] Weka. Weka, 2015. `http://www.cs.waikato.ac.nz/ml/weka/`, Accessed: 2019.02.27.

[WGJ18] Li-sheng Wei, Quan Gan, and Tao Ji. Skin Disease Recognition Method Based on Image Color and Texture Features. *Computational and Mathematical Methods in Medicine*, 2018:1–10, aug 2018.

[WGVB17] Sjaak Wolfert, Lan Ge, Cor Verdouw, and Marc Jeroen Bogaardt. Big Data in Smart Farming – A review, 2017.

[WZD17] Xiaoping Wang, Fei Zhang, and Jianli Ding. Evaluation of water quality based on a machine learning algorithm and water quality index for the Ebinur Lake Watershed, China. *Scientific Reports*, 7(1):12858, dec 2017.

[XFZ$^+$19] D. Xiao, F. Fang, J. Zheng, C. C. Pain, and I. M. Navon. Machine learning-based rapid response tools for regional air pollution modelling. *Atmospheric Environment*, 199:463–473, feb 2019.

[XLL13] J. Xuan, X. Luo, and J. Lu. Mining websites preferences on web events in big data environment. In *2013 IEEE 16th International Conference on Computational Science and Engineering*, pages 1043–1050, Dec 2013.

[Yad13] Sonali Yadav. Comparative Study on Open Source Software for Cloud Computing Platform : Eucalyptus , Openstack and Opennebula. *International Journal Of Engineering And Science*, 3(10):51–54, 2013.

[YF12] Lenar Yazdanov and Christof Fetzer. Vertical scaling for prioritized VMs provisioning. In *Proceedings - 2nd International Conference on Cloud and Green Computing and 2nd International Conference on Social Computing and Its Applications, CGC/SCA 2012*, pages 118–125. IEEE, nov 2012.

[YF13] Lenar Yazdanov and Christof Fetzer. VScaler: Autonomic virtual machine scaling. In *IEEE International Conference on Cloud Computing, CLOUD*, pages 212–219. IEEE, jun 2013.

[YLS$^+$14] Jingqi Yang, Chuanchang Liu, Yanlei Shang, Bo Cheng, Zexiang Mao, Chunhong Liu, Lisha Niu, and Junliang Chen. A cost-aware auto-scaling approach using the workload prediction in service clouds. *Information Systems Frontiers*, 16(1):7–18, mar 2014.

[YOU15] Jamileh YOUsefi. Image Binarization using Otsu Thresholding Algorithm. *Research Gate*, (May), 2015.

[Yu14] Stella Casey-Schueller Kriss Yu, Feng. OPEN JOURNAL OF MOBILE COMPUTING AND CLOUD COMPUTING A Design of Heterogeneous Cloud Infrastructure for Big Data and Cloud Computing Services. *OPEN JOURNAL OF MOBILE COMPUTING AND CLOUD COMPUTING*, 1(2), 2014.

[ZH18] Z. Zong and C. Hong. On application of natural language processing in machine translation. In *2018 3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, pages 506–510, Sep. 2018.

[ZHA01] Lin ZHANG. image adaptive edge detection based on canny operator and multiwavelet denoising. In *2014 International Conference on Computer Science and Electronic Technology (ICCSET 2014)*. Atlantis Press, 2015/01.

[ZHA04] Guojun ZHANG, Dengsheng. Review of shape representation and description techniques. *Pattern Recognition*, 37(1):1–19, 2004.

[ZJ17] Sayali Zirpe and Bela Joglekar. Polarity shift detection approaches in sentiment analysis: A survey. In *Proceedings of the International Conference on Inventive Systems and Control, ICISC 2017*, pages 1–5. IEEE, jan 2017.

[ZP12] Ns Zulpe and Vp Pawar. GLCM textural features for brain tumor classification. *International Journal of Computer Science*, 9(3):354–359, 2012.

[ZRL77] G W Zack, W E Rogers, and S A Latt. Automatic measurement of sister chromatid exchange frequency. *Journal of Histochemistry and Cytochemistry*, 25(7):741–753, jul 1977.

[ZYF$^+$18] Bin Zhang, Le Yu, Yunbo Feng, Lijun Liu, and Shuai Zhao. Application of Workflow Technology for Big Data Analysis Service. *Applied Sciences*, 8(4):591, apr 2018.