
# Unsupervised Machine Learning Application for the Identification of Kimberlite Ore Facie using Convolutional Neural Networks and Deep Embedded Clustering

Sean Langton

Supervisor: Dr Sebnem Er

A Minor Dissertation submitted to the faculty of Science, University of Cape Town, Cape Town, in partial fulfilment of the requirements for the Master of Science in Data Science.

Cape Town, March 2021

# Acknowledgement

To my wife, Claudia Langton, thank you for being my rock and unwavering support base through a very challenging couple of years and global pandemic. Your support and encouragement has made all the difference, all while bringing our son into this world.

To my parents, Mark and Katie Langton, thank you for your never ending financial and moral support of my academic endeavours. You have taught me the importance of an education and the value of hard work. To my little sister Kerri Langton, thank you for continuing to be an example of dedication and excellence in all aspects of life, even in the face of isolation through the pandemic.

A special thank you to Lorne de Kock, who's never ending support during the implementation of the project at Kao Mine was absolutely critical to the project's success. You went out of your way to to assist in every way possible, and I am extremely grateful for all of it. Thanks to John Ward, your guidance and input from a geological perspective was key in directing the focus of this project.

A big thank you to the entire team at Stone Three Mining. Your generosity in providing the hardware and software platform for data collection, as well as your continued input and guidance during the modelling phase, provided all the tools I needed to get this project over the line. The ground breaking work you continue to do in the mining and computer vision space is an inspiration.

Finally, I would like to thank my supervisor Dr Sebnem Er. Your relentless positivity and support has carried me through even the toughest days. Thank you for creating an open environment that allowed me to explore the world of data science and machine learning. The effort and time you put into my development is hugely appreciated.

# Abstract

Mining is a key economic contributor to many regions globally - especially those in developing nations. The design and operation of the processing plants associated with each of these mines is highly dependant on the composition of the feed material. The aim of this research is to demonstrate the viability of implementing a computer vision solution to provide online information of the composition of material entering the plant, thus allowing the plant operators to adjust equipment settings and process parameters accordingly. Data is collected in the form of high resolution images captured every couple of seconds of material on the main feed conveyor belt into the Kao Diamond Mine processing plant. The modelling phase of the research is implemented in two stages. The first stage involves the implementation of a Mask Region-based Convolutional Neural Network (Mask R-CNN) model with a ResNet 101 CNN backbone for instance segmentation of individual rocks from each image. These individual rock images are extracted and used for the second phase of the modelling pipeline - utilizing an unsupervised clustering method known as Convolutional Deep Embedded Clustering with Data Augmentation (ConvDEC-DA). The clustering phase of this research provides a method to group feed material rocks into their respective types or facie using features developed from the auto-encoder portion of the ConvDEC-DA modelling. While this research focuses on the clustering of Kimberlite rocks according to their respective facie, similar implementations are possible for a wide range of mining and rock types.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

The process of mining and extracting precious minerals from the surface of the earth is often a complex and expensive one. An important step in this operation is the extraction of the targeted mineral from the rock ore body.

A typical mining process plant consists of a complex circuit of crushing and screening equipment, as well as density based separation equipment to extract the valuable minerals based on their increased density. Variations in the rock type and mineral composition can have a significant influence on the performance of each type of equipment and thus the overall effectiveness of the processing plant in general.

In this research, Convolutional Neural Networks (CNN's), auto-encoders and clustering algorithms, within the field of computer vision and unsupervised learning, are utilized to provide a tool for recognising specific rock types or facies entering the processing plant. By doing so, the solution aims to provide a means to adjust plant processing parameters and equipment in a proactive manner, as opposed to

the current reactive measures used.

The main aim of this research is to demonstrate how a CNN based clustering method can be effectively used to implement an unsupervised machine learning model for recognizing changes in incoming rock types at a mining plant, specifically a diamond mining operation.

## 1.2 Objectives and Significance

Currently, most mining plants make use of very basic and often inaccurate methods of capturing material feed type into the plant. This often consists of hourly recordings of the total number of trucks fetching material from each of the specified mining areas - each of which present a different ore blend to the processing plant, each with their own characteristics and diamond potential. This process is cumbersome and often inaccurate, with results only reaching the mining plant management team well after the material has been fed into the plant for processing.

This research will explore the use of CNNs, auto-encoders and clustering algorithms to identify changes in the type of rock entering a mining process plant on the feed conveyor. With the successful implementation of an online material monitoring system, mines will be able to adjust their process parameters and optimize equipment set points based on the properties of the feed material. By implementing an unsupervised clustering model, the solution aims to provide an easily adaptive model available to various mining process plants with a variety of rock and mineral types, avoiding laborious, expensive and time-consuming labelling of training data.

This solution would be incorporated as part of a mining operations' overall digitization strategy, which may include several other strategies such as predictive

maintenance and process automation.

The objectives of this research are to:

- Cluster images of rock ore material on a conveyor belt using CNN, auto-encoders and a clustering algorithm based on the rocks' unique attributes as a result of varying origin within the mining pit.
- Create a clustering model that is able to identify and cluster images according to normal operations versus abnormal situations such as breakdowns, resulting in an empty conveyor belt.

The significant contributions of this research is to:

- Provide an unsupervised clustering model that is able to identify and cluster images of incoming rock to a mining process plant according to the origin of the rock ore within the mining pit. The ability to receive online information related to the type and origin of the rock material will allow plant operators to adjust their equipment settings and process parameters. The informed and proactive changes will improve overall plant efficiencies related to mineral recovery, as well as power savings as a result of optimized equipment settings.
- Provide a tool to estimate the hourly blend ratios and percentage of rock ore fed into the plant from each area. Having an accurate blend ratio is an important part of a mining plants process control. The blend ratio is the basis for estimating a plants recovery factor, which is defined as the actual target mineral recovery versus the calculated total availability of that mineral within the rock type being processed. These calculations all feed into the overall profitability calculations for the mining operation, and form a critical component of day-to-day performance results.

## 1.3 Layout

This dissertation discusses the application of an unsupervised machine learning model which makes use of CNNs, auto-encoders and a clustering algorithm to estimate the rock ore type entering a diamond mine using images captured from the feed conveyor belt transporting material into the processing plant.

Chapter 2 provides an overview of diamond mining and introduces the problem within the context of an actual mining operation in Lesotho, used as the basis for this research. It also goes on to describe the different rock ore types and how they originate from the various facies within the mining pit.

Chapter 3 provides a review of literature relating to similar applications within the field of computer vision used to estimate material type in mining operations. It also discusses several novel methods not yet implemented in this type of application, which will be the focus of this research.

Chapter 4 describes the data set and how it was collected and managed. It describes both the hardware and software installation used to capture the images, as well as their storage and accessibility.

Chapter 5 presents the method for creating the unsupervised learning model, as well as several variations of the method which are explored further during the implementation phase.

Chapter 6 provides a description of the environment and setup used to build and implement the model and research. It describes the process of configuring and evaluating the model.

Chapter 7 provides the results from the clustering of the various rock types using the unsupervised learning methods.

Chapter 8 provides an overview conclusion and summary of the results obtained.

Chapter 9 provides recommendations for future work based on the results achieved.

# Chapter 2

# Diamond Mining

Considering that the primary purpose of this research is to develop a solution for the identification of different types of Kimberlite rocks entering the mining plant, a general understanding of Kimberlite geology and the diamond mining process is important. This chapter aims to provide an overview of how Kimberlite rocks are categorised according to origin, as well as to provide a concise overview of the mining process. A general understanding of the mining process is important to frame the context of the research and its potential impact within a diamond mining operation. This chapter also introduces the specific diamond mine from which the data set was collected, as well as detailing its specific Kimberlite facie or areas.

## 2.1 Definition

Mining is defined as the process of extracting useful minerals from the surface of the earth. In the case of pipe diamond mining, these precious stones are extracted from "pipes" or natural volcanic fissures which have risen from deep within the earth's crust (Skinner & Marsh 2004). The diamond bearing igneous

rock contained within these pipes is known as Kimberlite.

Kimberlite can be classified according to differing rock facie. Mitchell (2012) suggests that these different facie or Kimberlite types can be associated with variations in magmatic activity during its formation.

## 2.2 Kimberlite Facie

In order to understand the classification and grouping of Kimberlite rock types, it is important to discuss the concept of facie. Facie can be thought of as different classes or variants of Kimberlite rock with unique characteristics in terms of overall composition and appearance. Cas et al. (2009) suggests these classes are derived from the underlying process of formation, and are primarily identified through differences in textural elements such as crystal shape, crystal grain-size, degrees of rounding of the crystals and colour.

The variation in magmatic processes during the formation of a pipe leads to the variation in Kimberlite properties and thus facie types. These variations in formation have been proven to affect the grade and density distribution of diamonds within the rock type as discussed by Kjarsgaard (2007).

The variation in diamond grade and concentration across facie is a key driver of the requirement of mines to understand the total percentage contribution of the mine feed ore. Blends of multiple facie are adjusted and managed on an hourly basis in order to ensure the total plant recovery remains economically viable over both the short and long term.

An example of the importance of blend management is as follows: if a mining operation chose to only process the highest quality facie first, it would initially find itself in a very strong financial position, through the recovery of a large number of high quality diamonds. However, as this facie runs out, the mine

might be left with low quality facie which cannot be mined viably, especially as mining operations move deeper into the pipe, resulting in increased extraction costs. By blending the processed facie correctly, the lifespan of the operation can be optimized, as well as creating a financially stable operation without large variations in earnings. This is an important consideration in an industry where the cash-flow of the operation is critical to its success.

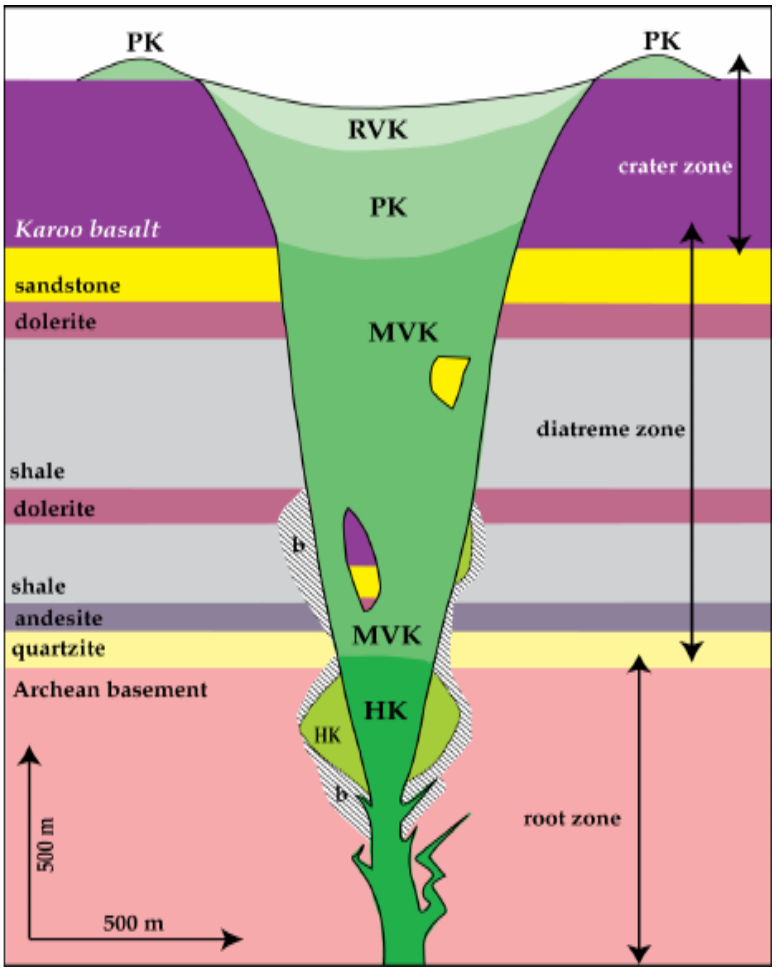


Figure 2.1: *A classic model of a South African Kimberlite pipe showing the various facie generated as well as typical surrounding rock types; pyroclastic kimberlite (PK), resedimented volcaniclastic kimberlite (RVK), massive volcaniclastic kimberlite (MVK) and hypabyssal kimberlite (HK), from Kjarsgaard (2007)*

Although not explored in any detail within this research, another important consideration when monitoring the Kimberlite facie ratio is to consider the potential inclusion of rock material surrounding the targeted Kimberlite pipe. The inclu-

sion of this material is know as dilution and generally comprises between one to two percent of the total material fed into the plant. When mining areas of the pipe closer to the crater, basalt and sandstone tend to be the typical rock types which dilute the Kimberlite ore feed as shown in Figure 2.1. Basalt is easily identifiable through its dark composition and additional work could be done to include the identification of this type of rock using computer vision, resulting in an online dilution tool. Dilution is an important metric for any mining operation to monitor, and should be minimized to avoid processing any rock ore which does not contain the targeted mineral.

## 2.3 Kao Mine

With a general overview of diamond mining and Kimberlite geology covered in this Chapter, it is also important to provide some background into operations at Kao Diamond Mine, the operation on which this research is based.

Kao Diamond Mine, owned by Namakwa Diamonds, is located in the mountainous northern regions of Lesotho. The mine was developed on the Kao Kimberlite pipe; the largest diamond bearing pipe in Lesotho. The total resource of the pipe is estimated at 173 million tonnes of ore, containing approximately 12.4 million carats of diamonds. This mine, along with the two other major diamond mines in the region contribute 5.7% to the country's overall GDP, as well as provide a key source of employment for the local population.

### 2.3.1 Kao Mine Facie

The Kao Kimberlite pipe, on which the research is based, consists of four major Kimberlite facie. The sketch shown as part of Figure 2.3, is taken from results of detailed geological research conducted by Skinner (2009). Each of the facie is

Figure 2.2: *Kao Diamond Mine, Lesotho (Skinner 2009)*

clearly marked on the sketch on the left, with a top view of the actual mining pit on which the sketch could be overlayed shown on the right.

The research proposes that the Kao Kimberlite pipe consists of four main bodies of Kimberlite rock, namely Fragmental Kimberlite (FK), Gritty Kimberlite (GK), Lower Quarry Kimberlite (LQ) and Quarry Kimberlite (QK). These are the four main facie of rock mined at Kao Mine, and would represent the target classes for the modelling conducted in this research. Each of these facie is characterized by its own set of unique properties related to mineral composition, colour and texture. In this section we explore these properties and characteristics in more detail.

Skinner (2009) suggests that the Fragmental Kimberlite facie can be considered as a massive volcaniclastic rock. This rock type is defined as one formed by fragments of material during a volcanic eruption. The irregular shape and outline of the magmaclusts suggests that these formations originated from large Kimberlite bodies, fragmented during the explosions which formed the rock. Magmaclusts

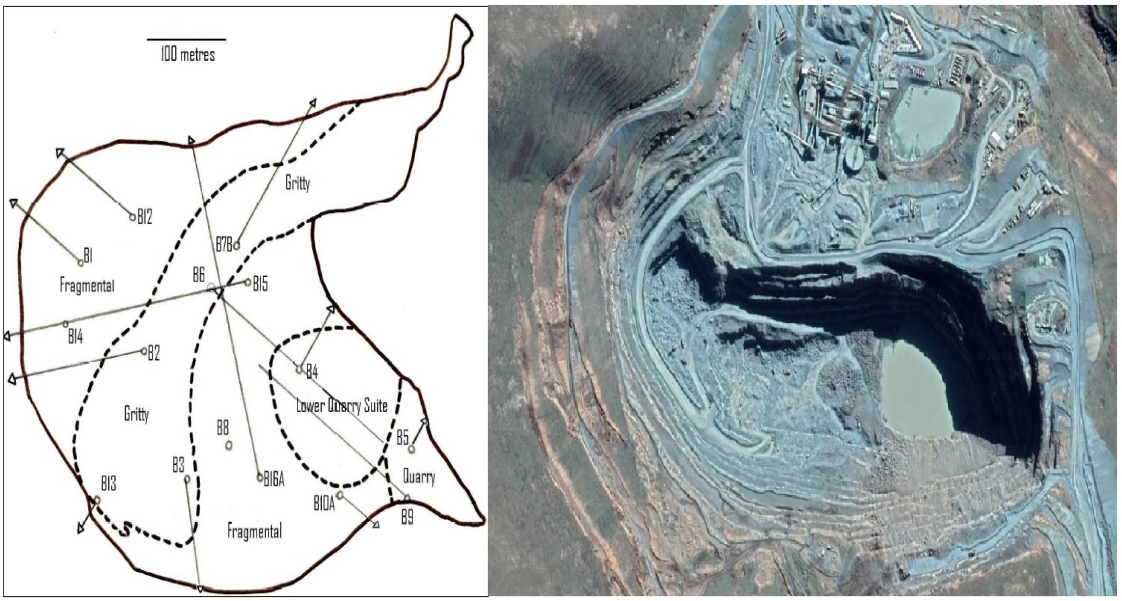


Figure 2.3: *Kao pipe Kimberlite facies overview. The three major facie include Fragmental Kimberlite (FK), Gritty Kimberlite (GK), Lower Quarry (LQ) and Quarry Kimberlite (QK). The lines sketched across the layout represent borehole drilled during the research campaign conducted by Skinner (2009).*

are defined as the fluidal-shaped bodies of Kimberlite formed by any process of magma disruption before solidification could occur, according to Smith et al. (2013).

The rock type consists of olivine grains (rock forming mineral typically green in colour), plus a smaller mass of opaques. The image in Figure 2.4 shows the crystalline structure and general colour characteristics of these rock types.

It is also important to note that diamond distribution within this type of rock tends to be clustered and not homogeneous in nature.

Gritty Kimberlite has more of a globular type crystalline structure when compared to other formations such as the Fragmental Kimberlite type. According to Skinner (2009), these Kimberlite types were formed by a changing Hypabyssal environment as opposed to explosive eruptions. The rock type is also classified as being of moderate to high interest when considering diamond potential. This

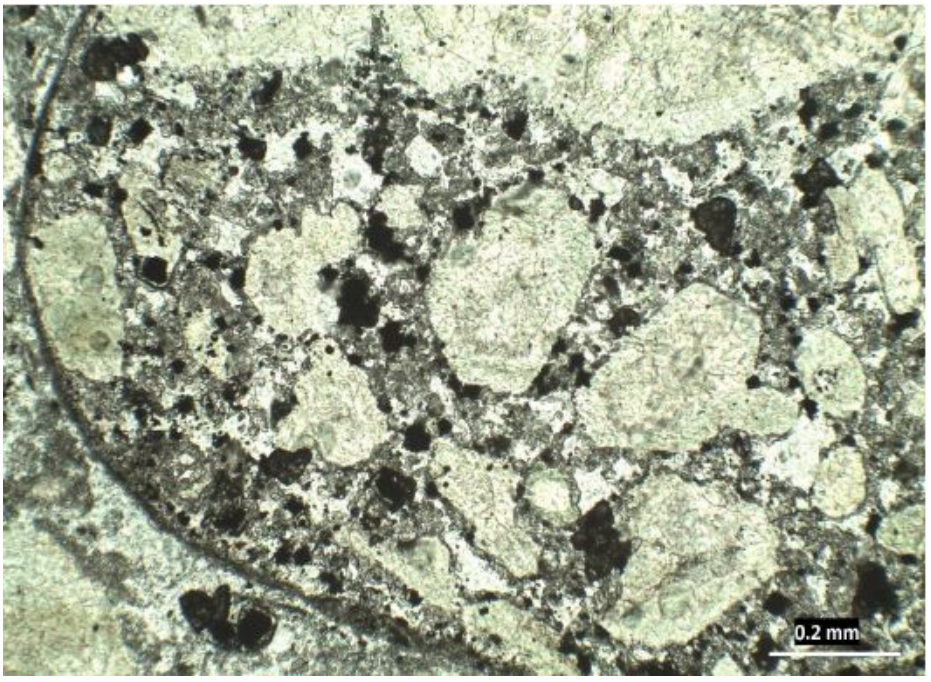


Figure 2.4: *Cross sectional view of Fragmental Kimberlite sample taken from the Kao pipe research conducted by Skinner (2009).*

measure relates specifically to the spacial density, size and stone quality of diamonds contained within that rock type.

Figure 2.5 shows a cross sectional view of the crystalline structure of Gritty Kimberlite. The structure of crystals is generally more rounded or globular than other facie from this pipe. The darker grey colour is interspersed with white clasts of altered sediments.

The Lower Quarry Kimberlite is considered to be relatively coarse grained. The mineralogy is not as advanced as the Gritty Kimberlite, an example of this are the globular crystalline features which are not as defined or developed as the Gritty type structure. The facie type has been rated as moderate interest related to diamond potential as defined by Skinner (2009).

The partially altered olivine crystal grains and surrounding finer ground-mass minerals are evident in the cross sectional view in Figure 2.6.

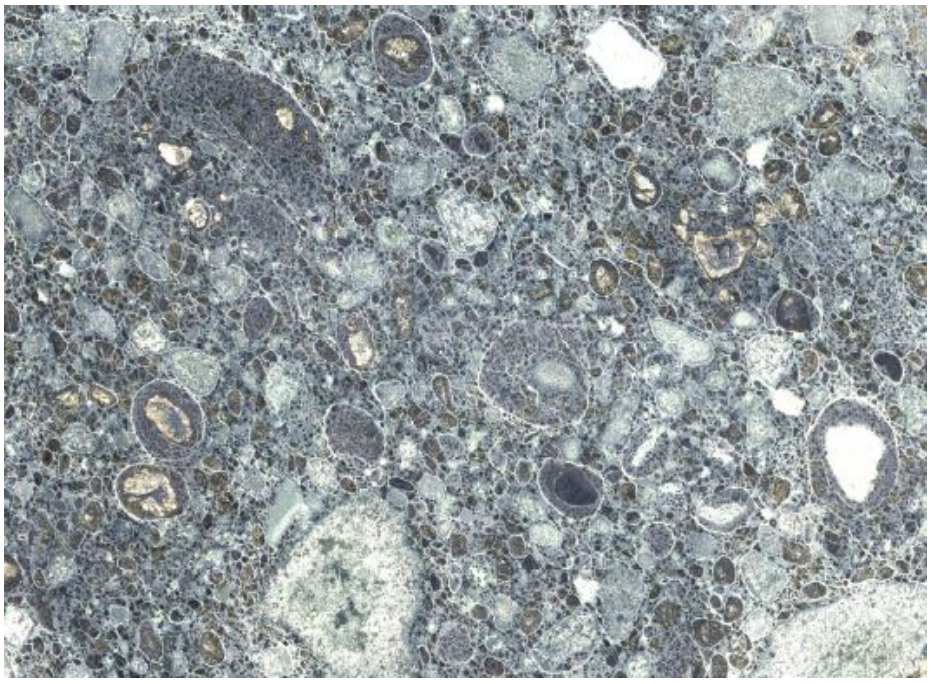

Figure 2.5: *Cross sectional view of Gritty Kimberlite sample taken from the Kao pipe research conducted by Skinner (2009).*

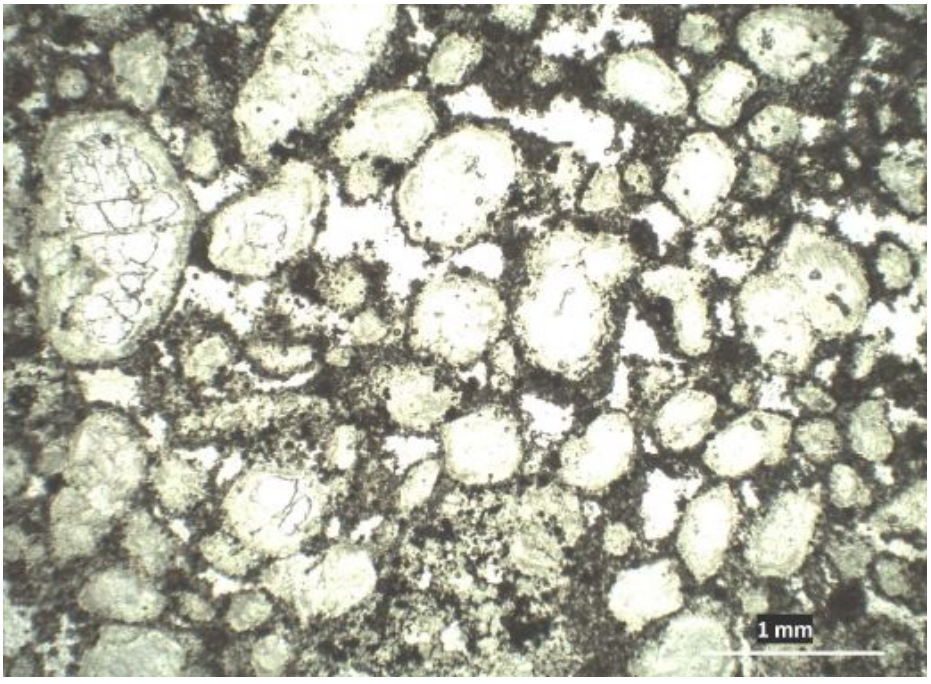


Figure 2.6: *Cross sectional view of Lower Quarry Kimberlite sample taken from the Kao pipe research conducted by Skinner (2009).*

Quarry Kimberlite is considered as a fragmental, massive volcaniclastic rock formed as a result of explosive eruptions (Skinner 2009). Significant differences between the facie type of the Fragmental and Gritty Kimberlite types exist. Some of the key differences include the smaller grain and globular crystalline structures, as well as the absence of fresh olivine grains from this facie type. This Kimberlite facie is considered as high interest in terms of diamond potential.

The cross section view in Figure 2.7 demonstrates this materials' more fine grained and lighter pale grey characteristics.

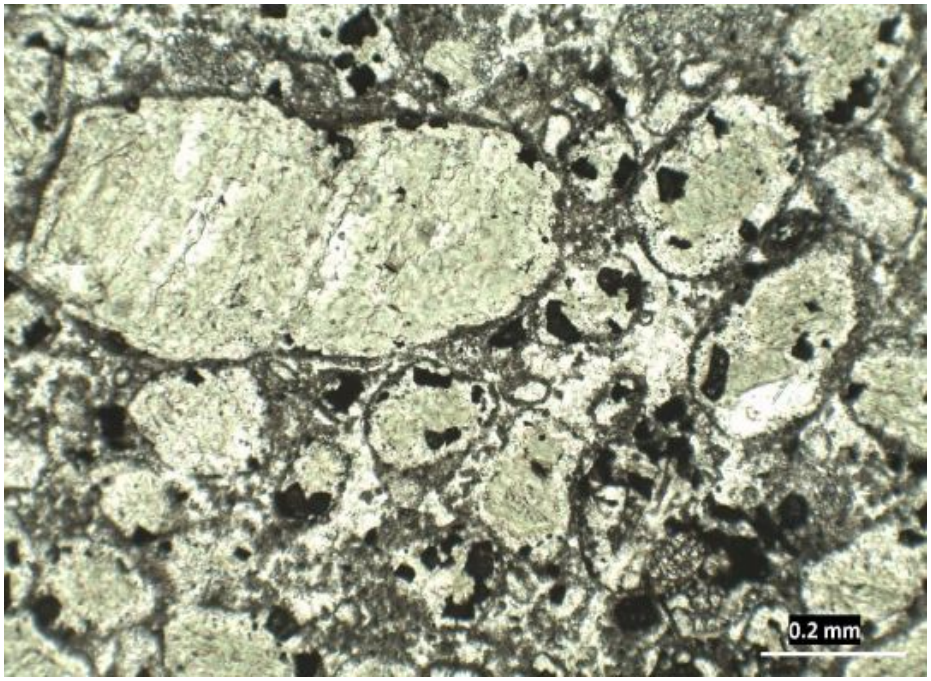


Figure 2.7: *Cross sectional view of Quarry Kimberlite sample taken from the Kao pipe research conducted by Skinner (2009).*

The high level overview and differences between facie mineralogy and physical characteristics should provide enough context to understand the key characteristics of each facie or rock type utilized, in order to identify respective material types entering the production plant on the feed conveyor.

Characteristics such as colour and texture are key components for building the model and should be kept in mind when considering model development in later

chapters.

In order to understand the impact of different facie types on the diamond mining process beyond a purely financial point of view, it is also important to have a basic understanding of the processing methods used to liberate and extract diamonds form the Kimberlite ore. This is covered briefly in the following section.

### 2.3.2 Kao Mine Processing Plant

This thesis has already touched on the importance of understanding the composition of ore being fed into the processing plant based on the concentration of diamonds within each facie and how that drives the financial aspect of the operation. Another important consideration is the varying physical characteristics of each facie type.

Properties such as hardness or clay content can vary from one facie to the next. It is thus important to understand the steps involved in processing Kimberlite ore to appreciate the relative impact these differences might have. By developing an online system for the estimation of facie type and feed ratios, it should be possible to adjust unit processes and equipment within the plant to account for known differences in physical properties across the facie. These preemptive changes should assist in optimising the overall performance of the processing plant, both from an energy usage and recovery point of view.

In order to appreciate the potential impact of this online solution, it is first important to have a basic understanding of the steps in diamond ore processing. The following five step summary is adapted from Chang (2018).

Step 1 - Crushing: The first step in ore processing is generally a crushing stage. Crushing takes the randomly sized ore feed and breaks it down into smaller more consistent sized rocks. Crushing is usually separated over several stages to grad-

ually break the material down to approximately 25 mm sized particles for further processing. This process helps liberate or free diamonds from the surrounding rock structure for extraction in later stages.

The settings of these crushers are generally dictated by the size and hardness of the incoming rock type, which can vary from one facie to another. Automatically adjusting these settings based on incoming facie type can provide a significant saving in power usage and equipment effectiveness.

Step 2 - Screening: A set of vibrating screens is generally found after each crushing stage. These screens help separate the rock out into its size fractions for further processing. The choice of correct screen panel and aperture size is critical for achieving optimal efficiency. These efficiencies can be impacted by the output of the crushing equipment, and with a more consistent crusher output, the screening configurations can be optimized.

Step 3 - Scrubbing: The process of scrubbing is used to wash the rock material, removing small particles and clay from the rocks. A scrubber unit usually consists of a large rotating drum. Material flows from one end to the other, with water added along the way. The product is screened on the other side to remove all small material and fines. The first three stages of the process prepare the material for the upcoming concentration stages.

Step 4 - Concentration: The heavy minerals and diamonds are then separated from the waste material based on density differences in a process called Dense Media Separation (DMS). In this process the ore is mixed with a ferrosilicon slurry and fed into a set of cyclones. In these units the denser material generally moves to the outer regions and downwards, while the light material is removed from the top. The heavy concentrate that flows out the bottom contains the diamonds and can be recovered in downstream processes.

Step 5 - Collection: The final stage is related to the recovery of the diamonds from the concentrate. Various forms of technology can be used to extract these diamonds, some of these include grease belts and X-ray sorting machines. The final concentrate is generally manually sorted at sorting tables under strict security surveillance.

Figure 2.8 provides a graphical overview of the main steps and equipment involved in the processing of diamond ore.

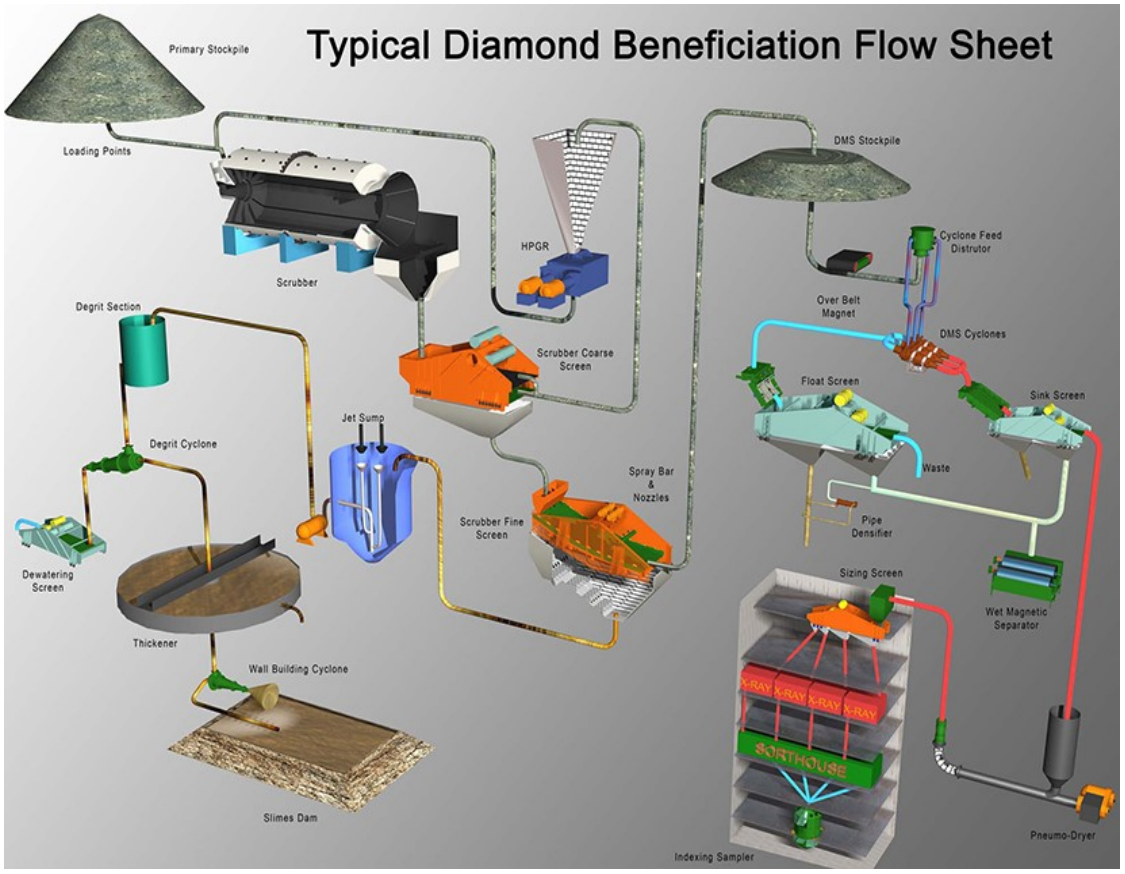


Figure 2.8: *Graphical overview of a typical process flow for diamond ore processing taken from Chang (2018).*

# Chapter 3

# Literature Review

The ultimate aim of this research is to develop a practical and industry relevant computer vision method for the classification of different rock ore bodies entering a mining plant via feed conveyors. With this aim in mind, this section aims to explore various supervised and unsupervised machine learning research papers which relate to similar work carried out, as well as to explore novel methods not yet utilized in this field. Each paper and method discussed holds some value in formulating a final approach used to tackle this research.

The use of machine learning and computer vision applications in particular are not new concepts within the mining industry. Throughout the 1990s and 2000s, several academic papers were published outlining various computer vision applications, with the majority providing an automated solution to the measurement of rock particle size distribution (PSD) on conveyor belts. At the time, this was an important problem to solve as the current methods required rock samples to be manually collected from the conveyor belts and put through a series of sieves.

Salinas et al. (2005) proposed a computer vision application to automatically estimate rock fragment distributions. Their solution centered around a Watershed

based segmentation technique. This segmentation algorithm, originally introduced in the early 1990s by Meyer (1994), acts by separating adjacent regions based on contrasting brightness within the image and is conceptually based on the notion of a landscape flooded by water. After segmentation, the individual regions were used as an estimate for rock volume and size.

Fast forward 20 years and significant advances in the field of computer vision within the mining industry have allowed for wide spread commercial implementation. Early versions of rock classification focused on the use of feature extraction and the application of Support Vector Machine (SVM) algorithms to the captured images.

Perez et al. (2015*a*) proposed a new method for online lithological classification of rocks on a conveyor belt. Each image extracted from the digital video was broken up into a set of sub-images and considered to be the main processing unit of the classification. A total of 210 images with 128 sub-images each (21,070 total sub images) were utilized as a training set, with a similar number used for the test set. Features were then extracted from the sub-images using Gabor filters (Gabor 1946). Following feature extraction, the feature vector for each of the sub-images was used as an input for the Support Vector Machine classifier. The method also employed the use of boundary detection utilizing the Watershed segmentation technique (Meyer 1994). It was then possible to assign each of the sub-images to a specific rock as defined by the boundary detection process. Voting amongst the sub-images which fall within a rock contour significantly improved the overall classification results.

More recently, Patel et al. (2017) developed an Iron ore classification model using data collected from Tensa mine located in State of Orissa, India. The method involved capturing image data of Iron ore rocks being transported along a conveyor belt. A total of 2200 images were captured, with 1466 used for training and 734

used for testing the model. The images were then pre-processed to improve their reliability using an adaptive median filter as proposed by Y. Zhao & Li (2007). This filtering acts to remove impulse noise (random variations in brightness and colour), while also reducing distortion of the images. A set of features were then extracted from the images which included nine colours and nine textures. The colour feature extraction focused on the red green blue (RGB) colour space primarily due to ease of availability for these types of images, while the texture features were extracted from the hue, saturation and intensity (HSI) space. An SVM algorithm was then used for the classification model. The use of an SVM for this type of application has several advantages, including the ability to map nonlinear separable problems into linear separable problems.

The past few years have seen significant advances in the field of Deep Learning and Artificial Neural Networks (ANN). These advancements in Deep Learning, fueled by increased accessibility to the computing power required to leverage this technology, have allowed for significant development of computer vision applications involving Convolutional Neural Networks. Many state of the art computer vision applications have been built on top of CNN architectures such as AlexNet developed by Krizhevsky et al. (2012*a*), VGGNet developed by Simonyan & Zisserman (2014) and GoogLeNet developed by Google (Szegedy et al. 2015).

With these developments in CNNs, Liu et al. (2020) utilized the Faster R-CNN framework to identify various rock types, with the primary application of on-site sampling in underground mines. The method utilized 1034 images across eight rock types. This framework is composed of a Region Proposal Network (RPN) and Fast R-CNN, both of which share a simplified VGG16 network. The VGG16 network is utilized as the underlying feature extraction tool, and the application's positive results highlight the applicability of this network type for rock type identification. The model achieved a 96% recognition accuracy on the

test set when identifying a single rock type image, and 80% for most rock types when considering a multi-rock hybrid image.

Another significant contribution to research in this field is related to the use of unmanned aerial vehicles (UAVs) to capture and process images of stockpiles by Schenk et al. (2019). The research utilizes an instance segmentation technique to identify individual rocks and calculate their respective sizes. This provides an automated method to calculate the overall PSD of rock stockpiles. The research paper provides an excellent example of the benefits of using the Mask R-CNN pixel wise instance segmentation model over other segmentation methods for rock stockpiles. This implementation provides one of the most similar use cases compared with this research, and provided key insight into selecting the best method for segmentation of individual rocks per image. The results indicated good performance of the model type for this application, and in general outperformed all other segmentation methods applied to date.

Ran et al. (2019) proposed a novel CNN network to classify six rock types, providing a much simplified architecture when compared to other more standard versions, such as VGGNet and AlexNet. The model, known as Rock Types deep CNN (RTCNN), consists of just two convolutional, max-pooling and fully connected layers. The model utilized a total of 2290 images across six rock types, with 60% used for training and 20% each for validation and testing. This data was further augmented through random cropping and flipping operations. Importantly, results seemed to suggest that the simplified model outperformed the more complex frameworks, even when transfer learning was utilized. This suggests that key colour and texture features required to classify rock types can be effectively learnt with smaller and less computationally expensive models. This is an important consideration when assessing the most practical solution to implement on a remote mining site.

One of the biggest drawbacks with all methods discussed up to this point is the requirement for a significant labelled data set. The inherent nature of neural network architecture, with their vast number of parameters, dictates that most models required a large training data set in order to avoid over-fitting. The labelling of rock images is both time consuming and potentially inaccurate. The recent advances in the field of unsupervised machine learning provides an opportunity to leverage vast amounts of easily captured images, without the need to manually label each image or rock type within an image.

Although there is very limited literature available for applications of these methods within the mining sector, several recent studies and academic papers have been produced implementing these unsupervised learning methods on more commonly available data sets.

Xie et al. (2016) introduced a method for clustering within the computer vision space. The primary aim of this method is to simultaneously solve for cluster assignment utilizing a reduced feature space. The method, known as Deep Embedded Clustering (Xie et al. 2016), makes use of parameterized non-linear mapping from the original image feature space using neural networks, which is in turn applied to a clustering objective to learn and optimize the mapping.

Deep Embedded Clustering makes use of a two stage process. First, the parameters are initialized through the implementation of a deep auto-encoder, constructed from an encoder and decoder part of an Artificial Neural Network. This method is typically used to learn efficient data codings with respect to the dimensionality of data in an unsupervised manner, as well as de-noising applications, allowing for dimensionality reduction and the learning of key features from the original feature space Vincent et al. (2010). The second stage is considered as the parameter optimization or clustering process and works by iterating between an auxiliary target distribution and minimizing the Kullback-Leibler (KL) diver-

gence to it. The model is then able to alternate between these two steps. It first computes the soft assignment of the embedded points as well as the cluster centroids, followed by the updating of the deep mapping and refinement of the centroids and assignments.

Because the Deep Embedded Method learns from the current high confidence assignments, it is important to ensure that at least a small percentage of the images within the data set are clearly distinguishable according to their relative features. Without a set of high confidence assignments, this method will struggle to form meaningful clusters. This is an important observation for the use in a mining ore application, where the differences in rock colour and texture are often very slight.

In an attempt to overcome the limitations imposed on Deep Embedded Clustering (DEC) by the requirement for a set of high confidence assignments, Ren et al. (2019*a*) propose the use of a semi-supervised deep embedding technique. Unlike DEC, this method makes use of prior knowledge to guide the learning process, achieved through the use of a set of both labelled and unlabelled images.

Yang et al. (2017) built on the concepts developed by Xie et al. (2016) through the implementation of simultaneous deep learning and clustering using k-means. This method attempts to optimize both the dimensionality and clustering tasks jointly, thereby creating a learned feature space which is optimized for clustering. While Deep Embedded Clustering also uses both Deep Neural Networks and Clustering, the loss function only contains clustering loss with no penalty on reconstruction loss as defined in this novel method.

Guo et al. (2017) developed a version of the method known as deep clustering with convolutional auto-encoders (DCEC), utilizing a CNN type architecture over the traditional dense network type within the staked auto-encoder (SAE). This up-

dated architecture provided significant improvements to the overall performance of the model type when considering image data. The method outperformed the original DEC method, as well as more standard approaches, such as k-means clustering, when tested on benchmark data sets such as USPS (9,298 grey-scale handwritten digit images of 16x16 pixels) and MNIST (70,000 handwritten grey-scale digits of 28x28 pixels).

Further improvements to the deep embedded clustering method were achieved by Guo et al. (2018) utilizing deep embedded clustering with data augmentation (DEC-DA), implementing a data augmentation pipeline as part of the training process during both the auto-encoder pre-training, as well and the clustering and feature space optimization training process. The results from this implementation showed superior performance for both convolutional, as well as dense network architectures, compared to all other previous implementation. This most recent implementation forms the basis of the method on which this research is focused.

This research will aim to follow a similar methodology by implementing a Deep Neural Network and clustering method, which optimizes both the dimensionality reduction and clustering process concurrently. Through the implementation of an unsupervised clustering method, the hope is that the implementation of the method for real world rock ore composition monitoring on mines can be achieved without the need for time consuming and costly data labelling, cited as one of the major barriers of entry for current supervised classification applications.

As a starting point, it is important to collect a representative sample of image data from the relevant mine using the correct hardware and software setup. This is covered in detail in the next chapter.

# Chapter 4

# Data

The image data for this research was collected during February of 2020. The quality of image data, and methodology used to capture it forms the basis for the research, and it is therefore important to discuss this process in detail. This section covers the data collection and prepossessing.

## 4.1 Data Collection

The data for this research was collected from Kao Diamond Mine, located in the mountainous region of North East Lesotho. The equipment used to capture the images was kindly provided on loan from Stone Three Mining, an existing computer vision technology provider in the mining sector. Stone Three collect image data from several mines across Southern Africa, with one of their primary allocations being an online particle size distribution capturing solution, leveraging laser cameras as opposed to colour cameras. For this specific application, a colour camera was selected due to the need to include rock colour as a potential feature.

As discussed in the Literature Review section, there are several examples both within academic research as well as commercial implementations of cameras which

have been mounted above conveyor belts to capture image or video data used to analyze the incoming material characteristics and types. The equipment setup, discussed in more detail in the next subsection, was based on similar research conducted by Galdames et al. (2017) and Galdames et al. (2019). In both of these examples, the researchers made use of a camera and light setup over a conveyor belt for image capturing.

### 4.1.1 Equipment and Installation

The installed equipment consisted of a single colour CCTV IP68 Gig camera, mounted within an industrial grade housing, along with a set of two Highmast 220v flood lights. The equipment was flown to Johannesburg, after which it was transported by road to the mine in Lesotho.

A week long site visit was conducted to complete the equipment installation. Equipment installation was conducted on a Wednesday, which is the weekly plant shut day for maintenance work.

During the first day of the site visit, several locations and conveyors were assessed as potential sites for the installation. The primary feed conveyor of the plant was selected as the most appropriate location. This location had existing infrastructure such as a remote distribution board, as well as a protective enclosure which housed existing equipment. The primary feed conveyor is also the most important material stream to monitor, as rocks are fed on the primary feed conveyor before being processed or split up into multiple streams within the processing plant.

The camera and light assembly were installed inside an existing housing built for a Stone Three laser camera, used for particle size distribution measurements. This housing prevents natural light and shadows from distorting the captured images. The completed installation is shown in Figure 4.1.

Figure 4.1: *Camera and floodlight installation at Kao Diamond Mine, above the primary feed conveyor.*

The existing network infrastructure installed by Stone Three was leveraged for this camera installation. Figure 4.2 shows the remote distribution board installed at the feed conveyor. The installation provided a LAN network connection, along with direct power source for the camera. The camera was connected to the Stone Three remote server, onto which all captured images were written for storage. Once on this server, it is possible to log in remotely and access the image repository. During the camera setup, Stone Three also assisted with camera focusing and adjustments by checking the live feed images and guiding the site based installation.

Once the camera is manually focused, the flood lights are connected up directly to the plants' power supply and kept on 24 hours a day to ensure uniform lighting of the material surface. The fully implemented solution is shown in Figure 4.3.

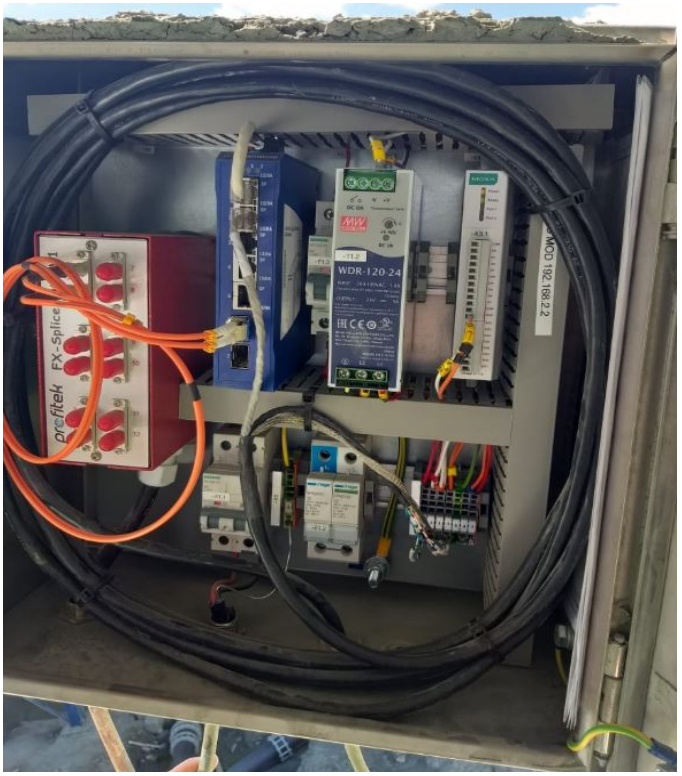


Figure 4.2: *Remote Distribution Board along with network connection used to connect the camera to the Stone Three remote server.*

Figure 4.3: *Camera and floodlight installation at Kao Diamond Mine, above the primary feed conveyor.*

### 4.1.2 Data Capturing Methodology

With the hardware installation fully implemented, and the camera linked to the Stone Three remote server via a local internet connection on site, the system is able to log images directly from the camera at set intervals. The system was set up to log images every 20 seconds for a week, collecting a total of approximately 25,000 images on the remote sever.

Ideally, images should be captured across a broad spectrum of possible environmental conditions, which at Kao Mine, might include wet and rainy conditions. During the data collecting campaign, rainy conditions were not experienced, which is an area for additional research in the future to assess the relative impact of wet material on the performance of the model pipeline. Due to the enclosure and flood lighting, image capturing is not affected by changes in natural lighting or time of day.

## 4.2 Data Description

The images captured onto the server have a resolution of 1000x760, in full colour. In order to provide the model pipeline with an optimal picture size, these images are cropped slightly to 992x662 during a prepossessing step. This new image dimension is fully divisible by two, and simplifies the implementation of the convolutional filter during the instance segmentation phase of the modelling pipeline.

For the purpose of the first phase of the modelling, a Python package known as Labelme.py (Wada 2016) is used to implement labelling of the individual rocks. This process is conducted on 120 images in total, providing a training and test set for the instance segmentation modelling. These images were selected randomly across the spectrum of available images captured during the image acquisition campaign. A broad spectrum of images were selected to provide a set of training

images that included rocks from all mining facies in a ratio that represents the average feed into the mining plant.

A screenshot of the images labelled using polygons within the Labelme.py package is shown in Figure 4.4. The labelled images generate a set of JSON files, each containing the unique identification number of the individual rocks, as well as their respective coordinates based on each point of the polygon.

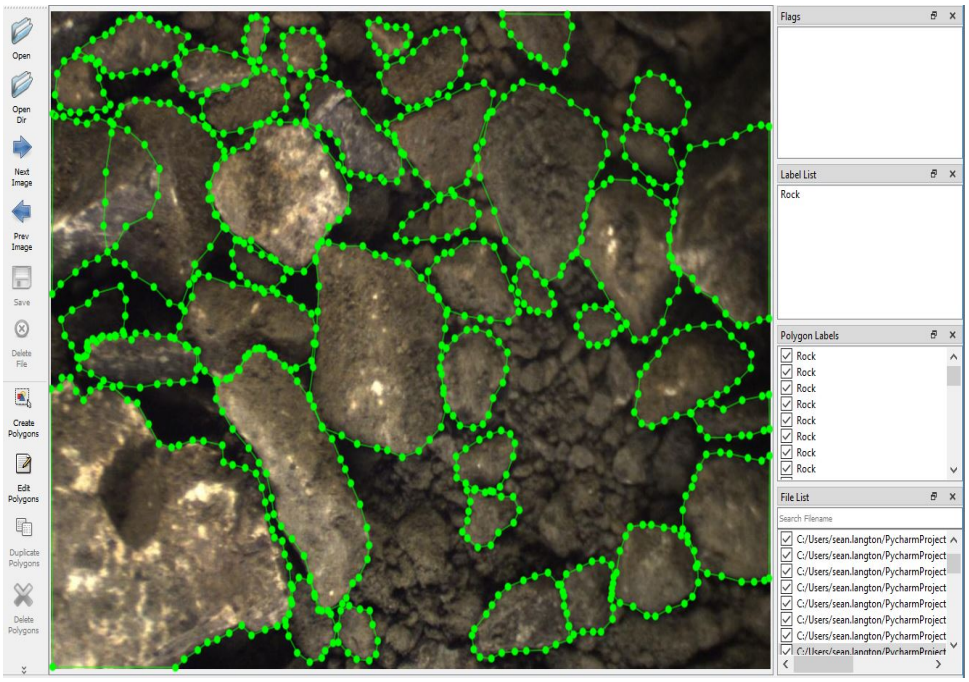


Figure 4.4: *992x662 images labelled using Labelme.py package (Wada 2016).*

Additional data processing of the output from the first stage of modelling is required. This output consists of individual rock images generated through the instance segmentation process. The details of this additional processing are covered as part of the Implementation chapter.

Once the data is successfully captured, pre-processed and labelled, it is then important to discuss the modelling pipeline and methods in more detail. This is covered in the next chapter.

# Chapter 5

# Method

This chapter provides a detailed overview of the method pipeline used to implement the rock clustering solution. The method can be categorized into two phases, namely the individual rock identification using instance segmentation, followed by the rock clustering phase using deep embedded clustering.

Before discussing these two phases in this chapter, a detailed overview of Convolutional Neural Network (CNN) is included, as this forms the backbone of the respective architectures used in both phases of the solution.

Ideally, the real-world solution should be as lightweight as possible to allow for practical implementation of a near live site based solution. The use of two phases and two separate models is not the preferred solution since it adds complexity and increased processing requirements to the implementation.

The phase one instance segmentation can be avoided completely if the rock material within each image is homogeneous, thus allowing the clustering model to be implemented based on an entire image as opposed to individual rocks within each image. This material assumption needs to be assessed on a case by case basis for real-world implementation, taking into account the material properties

of each mining site.

## 5.1 Method Design

An overview of the phased rock clustering solution is shown in Figure 5.1. The first phase of instance segmentation identifies individual rocks, while the second phase clusters each of the individual rocks, enabling the model to categorize the overall material type per image based on frequency of rock type occurrence.

### 5.1.1 Phase 1: Rock Segmentation

The first phase of the method used in this research involves identification and pixel-wise segmentation of individual rocks from the background or other material such as sand. The first step required the manual labelling of 120 images captured from a camera installation over the feed conveyor at Kao Mine as discussed in the previous chapter. The labelling focused on the larger and clearly visible rocks, excluding the small stones and sand which would not provide any significant information in the form of texture or colour, required for the next stage of processing.

The decision to label 120 images in total for the test and training dataset was made based on literature reviews of similar applications. In the case of Schenk et al. (2019), this research utilises both Mask R-CNN and transfer learning to generate an instance segmentation model for rock particle size distribution. A total of 1,000 individual rocks were labelled, as per the paper's suggestion, producing sufficiently good results. Given the low resolution of images used for this research, a decision was taken to increase the overall number of labelled rock instances to approximately 3,000.

The labelled training images are provided as an input to a Mask R-CNN model,

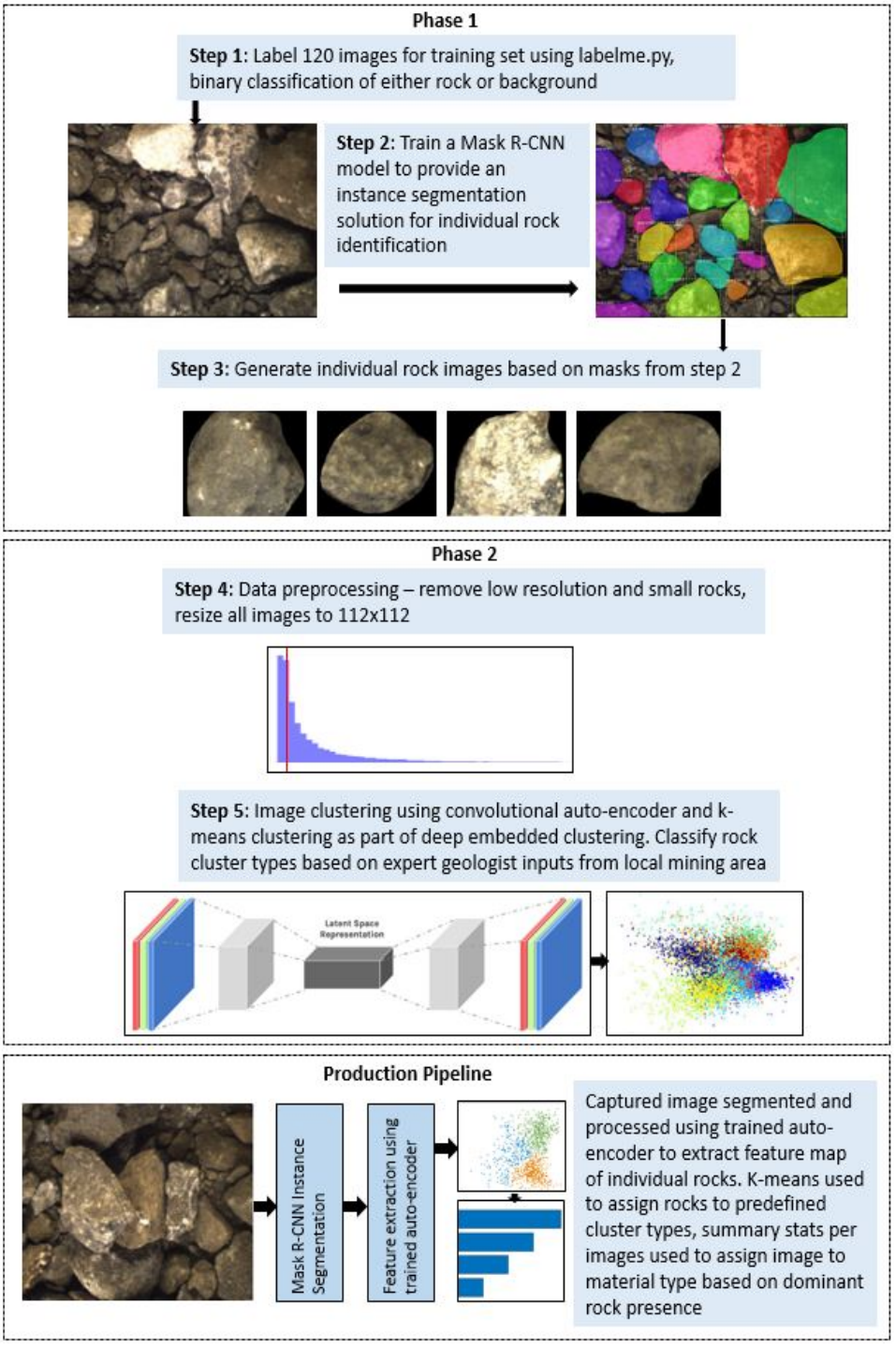


Figure 5.1: *Method design for image identification and rock clustering. The first phase (Phase 1) implements a Mask R-CNN model for individual rock segmentation, with the second phase (Phase 2) utilises these individual rock images in an unsupervised clustering method known as deep embedded clustering. This method makes use of k-means clustering based on the feature space derived through the auto-encoder training.*

utilized for instance segmentation. This model type is preferred over several other segmentation models due to its ability to provide a pixel wise mask, required in order to generate an output of individual rock images as shown in step three of Figure 5.1, as well as proven success in the instance segmentation of rock stockpiles (Schenk et al. 2019). The relatively low number of training images meant that transfer learning was utilized to train only the head of the model, utilizing pre-existing model weights from training on a standard dataset. The model utilized a Resnet 101 backbone due to its superior performance when compared to other standard architectures for this specific application. This process and the superior performance of the backbone type is discussed in more detail in the Implementation chapter.

Due to the complex nature of Mask R-CNN models, and specifically the Resnet 101 backbone, it is preferable to avoid the instance segmentation phase completely if the images captured from the conveyor belt show homogeneous rock types. This would provide a more light weight and robust solution for implementation on site, with "Phase 2" from Figure 5.1 being implemented directly on the initial images.

### 5.1.2 Phase 2: Rock Clustering and Identification

The second phase of the modelling process accepts a set of pre-processed individual rock images and utilizes the unsupervised learning method known as deep embedded clustering. The specific implementation of the method for this research includes data augmentation. This is a critical consideration to ensure the model does not learn oversimplified features of the input images such as shape or size.

The deep embedded convolutional clustering with data augmentation model (DCEC-DA) accepts resized individual rock images of 112x112 pixels each, while also capturing metadata related to their original image source. The model utilizes a convolutional auto-encoder (CAE) which is pre-trained on the image set to gen-

erate a set of key generative features. Further training incorporates a clustering loss function alongside the CAE loss function, which effectively improves the set of generated features to take into account the clustering ability of the overall model. This is discussed in more detail later in the Method section.

The output of the DCEC-DA model includes a set of rocks from each original image assigned to a cluster. These clusters of rocks, based on key generative features are then defined with the assistance of local expert's geological knowledge. The number of rocks per type per image can then be used to define the overall and predominant material type in each image, thus providing the mining operation with live information related to the composition of feed material to the processing plant.

The geological input for this thesis was kindly provided by John Ward, an expert on the Kimberlite formations found at Kao Mine and resident geologist. Johns' input was invaluable in assisting to confirm rock type clustering based on expected facie allocation. During the thesis, samples of rock images were sent to John to confirm or reject proposed facie assignments. An example of the allocation of images to respective facie type is seen in Figure 5.2.

A processing pipeline for new images would utilize the rock segmentation model, as well as the pre-trained CAE for feature extraction. These features are then used to assign each rock to the pre-defined clusters and rock type based on proximity or distance to the cluster centroid.

The remainder of this chapter focuses on a detailed overview of convolutional neural networks, a key component to both phases of the solution, as well as detailing the methods outlined in both Phase One and Phase Two as defined in Figure 5.1.

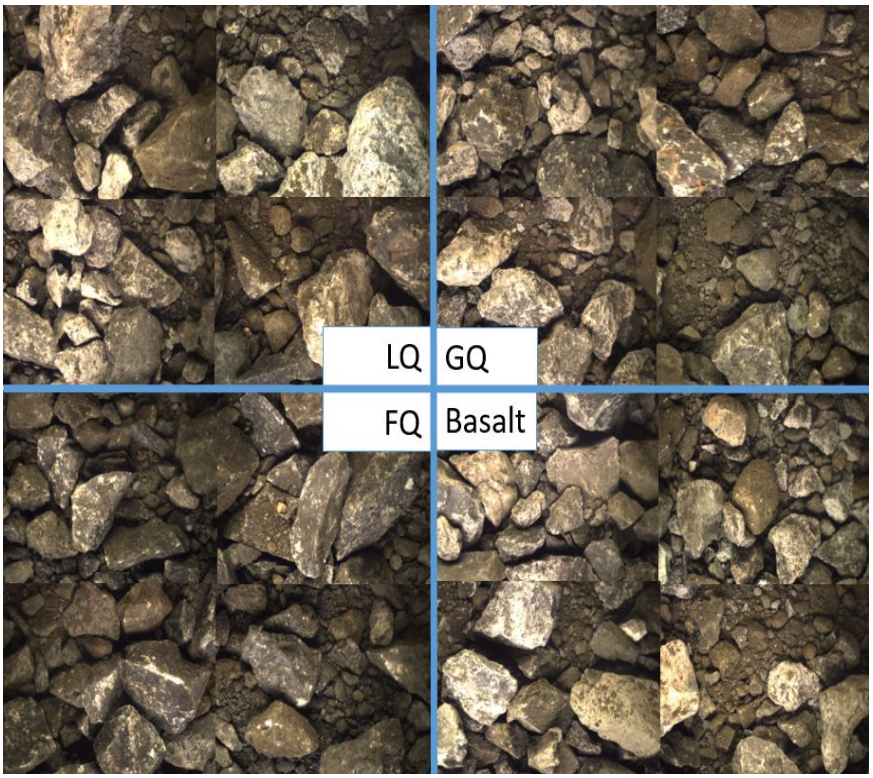


Figure 5.2: *Example of feed conveyor images categorized according to facie, with input from a Kimberlite geologist at Kao mine.*

## 5.2 Convolutional Neural Network

Convolutional Neural Networks (CNN) can be considered as a variant of the standard Artificial Neural Network (ANN) structure. The building blocks of these network types are neurons, which have learn-able weights and biases. Each of these neurons receives an input and translates it using a dot product and the user defined non-linearity function. The fundamental difference between ANN and CNN is related to the architecture of each type, which is outlined in detail within this chapter.

The major difference lies in the fact that CNN networks are specialized for processing data which has a grid like topography such as images, time series and audio data (Goodfellow et al. 2016). By employing the use of the convolutional mathematical operations, instead of the typical matrix multiplication operation

found in ANN networks "it is possible to write a convolutional operation as a matrix operation for an image with its rows or columns concatenated in a single row or column vector - defined as flattening", these networks are able to constrain their architecture in a more sensible way by arranging neurons in three dimensions (Stanford 2020). By doing so they leverage the convolutional process to allow the network to preserve spatial relationships between data such as those found between neighbouring pixels in an image.

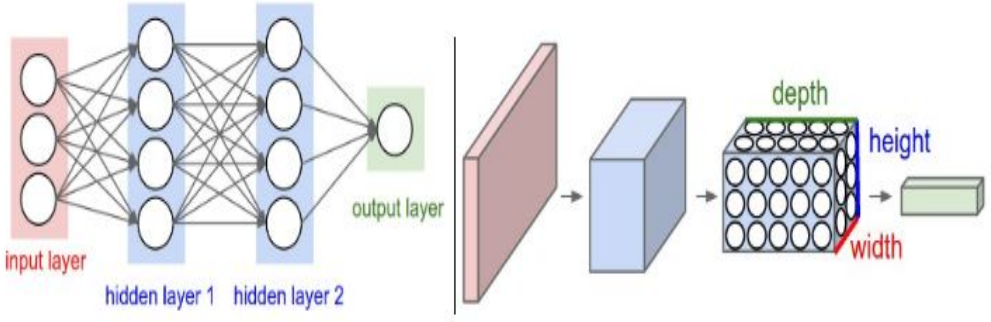


Figure 5.3: *Comparison between a 3-layer ANN network versus a CNN network. The CNN arranges neurons into a three dimensional structure (depth, width, height). The 3-dimensional structure along with the convolutional operations provide a significantly more optimized structure of processing topographical data such as images due to their inherent ability to extract features directly from images by considering spacial coherence, something a simple ANN cannot achieve. Sketch adapted from Stanford (2020) .*

In general, convolution can be thought of as a mathematical operation applied to two functions of real numbered ($\mathbb{R}$) arguments (Goodfellow et al. 2016). In general, the first function is referred to as the input, and the second function is known as the kernel or filter. In its most basic form, the output of a convolution can be defined as (Goodfellow et al. 2016):

$$s(t) = (x * w)(t) = \int x(a)w(t-a)da \tag{5.1}$$

Where the output ($s$) is a one-dimensional space, otherwise know as the feature

map and is a product of the convolution between input ($x$) and kernel ($w$) for integer values $t$ and $a$ respectively. This operation is generally represented by an asterisk (*). If the input and kernel are defined only for integer values of $t$, it is then possible to represent Eq.(5.1) as a discrete function (Goodfellow et al. 2016):

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{5.2}$$

Where the feature map ($s$) can be thought of as the sum of all convolutional operations between the input ($x$) and kernel ($w$) across all values of $a$ in the discrete domain space. The kernel function output itself varies according to the relative difference between the integer value $t$ and each value of $a$ within the domain.

For the application of this convolutional operation within CNNs, it is more practical to consider multidimensional applications, such as a two-dimensional image ($\boldsymbol{I}$). This two-dimensional input array is usually processed with a two-dimensional kernel ($\boldsymbol{F}$), which consists of an array of parameters adapted by the learning algorithm. These multi-dimensional input and kernel arrays are known as tensors, and the output of the convolutional operation in a two-dimensional space can be defined as (Goodfellow et al. 2016):

$$\boldsymbol{S}(i,j) = (\boldsymbol{I} * \boldsymbol{F})(i,j) = \sum_{m}\sum_{n} \boldsymbol{I}(m,n) * \boldsymbol{F}(i-m, j-n) \tag{5.3}$$

With $i$ and $j$ representing the convolutional output indexes ($\boldsymbol{S}$), and $m$ and $n$ representing the input indexes ($\boldsymbol{I}$).

Due to the binary nature of the convolutional operation, and its resulting commutative property, Eq.(5.3) can be equivalently written as (Goodfellow et al.

2016):

$$\boldsymbol{S}(i,j) = (\boldsymbol{I} * \boldsymbol{F})(i,j) = \sum_m \sum_n \boldsymbol{I}(i-m, i-n) * \boldsymbol{F}(m,n) \tag{5.4}$$

which has effectively flipped kernel ($\boldsymbol{F}$) relative to the input ($\boldsymbol{I}$). This modification provides a simplified and more computationally efficient formula for implementation within a machine learning application. This updated formula can be visualized as effectively sliding the input ($\boldsymbol{I}$) over the kernel ($\boldsymbol{F}$), which has inherited a set of input indexes $m$ and $n$ with a smaller range since the kernel ($\boldsymbol{F}$) would only ever be the same size or smaller than the input array ($\boldsymbol{I}$).

While Eq.(5.4) provides an improved formula for the convolutional operation, in practise, most machine learning libraries make use of a function called cross correlation. This function, defined as (Goodfellow et al. 2016):

$$\boldsymbol{S}(i,j) = (\boldsymbol{I} * \boldsymbol{F})(i,j) = \sum_m \sum_n \boldsymbol{I}(i+m, i+n) * \boldsymbol{F}(m,n) \tag{5.5}$$

is similar to the convolutional operation, but without flipping the kernel ($\boldsymbol{F}$). This type of operation is possible within the machine learning context, since the algorithm will learn the appropriate values of the kernels in their appropriate positions, regardless of whether or not the kernel has been flipped.

The resulting output from the cross-correlation Eq.(5.5) can be considered as a set of feature maps in the context of CNN. In the case of discrete convolutional operations, such as the processing of images using CNN, it is possible to consider convolution as the multiplication by a matrix, and thus a simple dot product over the entire volume of the filter (Aggarwal 2018).

Figure 5.4 illustrates the convolutional operation with the filter ($\boldsymbol{F}$) applied at two different positions on the input ($\boldsymbol{I}$). It also shows the complete output or

feature map ($\boldsymbol{S}$), a result from the dot product operation of the filter at each position on the input.

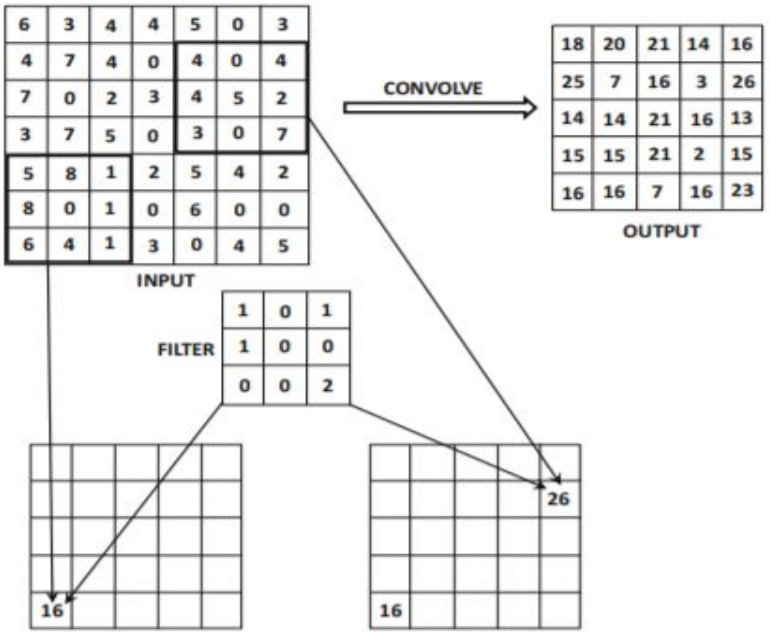


Figure 5.4: *An example of a convolutional operation between a two-dimensional input of 7x7 and filter of 3x3, generating an output or feature map of dimensions 5x5. The output has been restricted to areas where only the full kernel lies on the input matrix, adapted from (Aggarwal 2018) .*

The convolutional operation leverages four important concepts, namely; sparse interactions, parameter sharing, equivariant representations and the ability to work with inputs of variable sizes (Trivedi & Kondor 2017). This combination of attributes ensures that the implementation of a CNN type model provides a more efficient solution with vastly reduced parameters compared to a standard ANN for certain data types.

In order to quantify these concepts, it is important to compare the relative architectures of both the ANN and CNN type models. A traditional ANN type network utilizes a set of layers of fully connected neurons, meaning that each output interacts with the full set of inputs through matrix multiplication. On the other hand CNN models implement convolution in at least one layer of the

network. This type of operation utilizes sparse interactions, otherwise known as sparse connectivity or sparse weights. Sparse interactions are made possible through the use of a kernel that is typically several orders of magnitude smaller than the input. In the example of image data, each of these kernels can be passed over the input image pixels, detecting small meaningful features such as edges or corners that occupy a small fraction of the overall input pixels, thus reducing memory requirements and improving the overall performance of the model from a statistical perspective (Goodfellow et al. 2016).

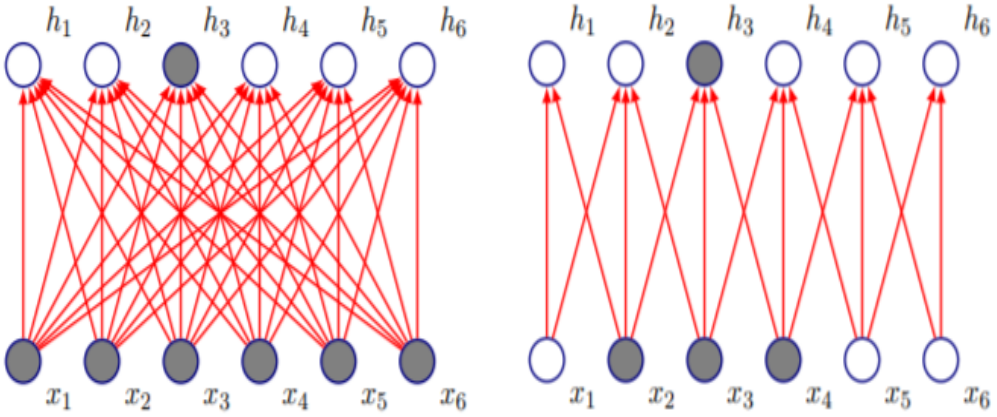


Figure 5.5: *An example of a fully connected network on the left hand side, making use of full matrix multiplication, while the network on the right hand side makes use of sparse connectivity through the implementation of a kernel of size three moved across each location of the input, adapted from Trivedi & Kondor (2017) .*

Parameter sharing is the second major attribute differentiating CNN type networks from fully connected ANNs. In a traditional network, each value of the weight matrix is used only once for calculating the output of that layer, whereas a convolutional neural network makes use of a kernel which is generally passed over every position of the input, effectively learning a single set of parameters across all locations. This concept of parameter sharing is also referred to as tied weights of a network, since the value of a weight applied to a certain location in the input is tied or linked to the value of the weight applied to all other locations

of that input (Goodfellow et al. 2016). This property lends itself to deal with scalability issues that might be experienced by a fully connected network when dealing with input data such as images with hundreds or thousands of pixels, requiring significantly less memory through a reduced total number of parameters (Abbas et al. 2020).

Equivariance to translation is a direct result of parameter sharing in convolutional neural networks (Ravanbakhsh et al. 2017). An example of this property is that if an input changes, it causes an equivalent change in the output. In the case of image data, the processing of these inputs using kernels creates a set of feature maps, and if certain features within the input image move, the resulting feature maps will cause those features to move by the same amount.

This property is extremely useful for utilizing the same function for a small feature across multiple locations on the input image. An example of this might include a facial recognition application, whereby a certain function is optimized to detect eye features. This function will be able to effectively detect these features at any position in the images, and detect multiple sets of these features across the image, which is a practical requirement when processing images which might have multiple faces in different areas of the image.

There are however certain transformations for which convolutional operations do not inherit the equivariance property, for image data this includes scaling and rotation. For these cases, alternative techniques can be considered to accommodate these transformations (Goodfellow et al. 2016).

Based on these fundamental concepts related to convolutional neural networks, it is then important to discuss the general structure and types of layers used to build such networks, namely; convolutional, pooling and fully connected layers.

### 5.2.1 Convolutional Neural Network Structure

A typical CNN architecture makes use of three general types of layers. These layers include convolutional, pooling and fully connected types. This section focuses on exploring each of these types of layers in detail. This network structure is implemented in both phases of the modelling pipeline in this research, and thus it is important to explore the concept in detail.

Figure 5.6 provides an overview of the complete network structure of a typical CNN. The network can be broken up into two general processes, namely feature extraction and classification.

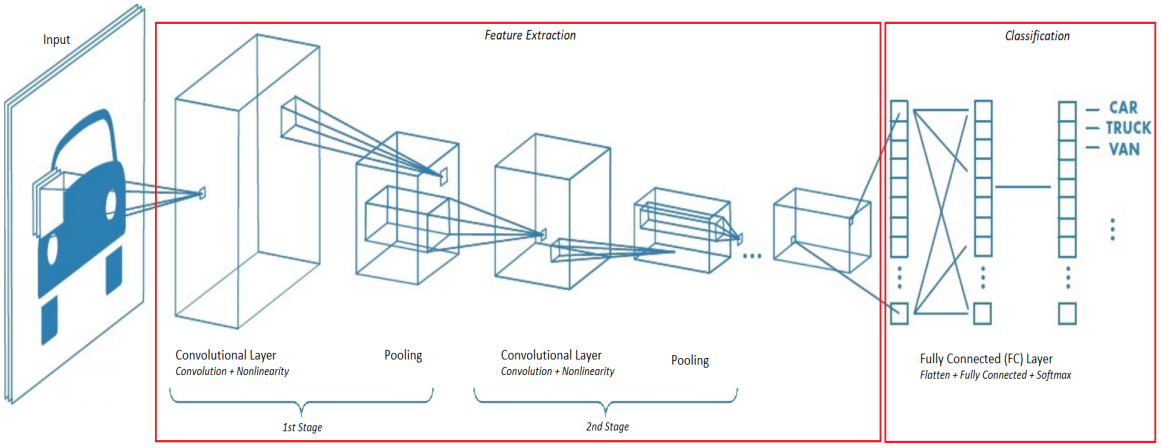


Figure 5.6: *An example of a Convolutional Neural Network structure, highlighting the feature extraction and classification phases as well as the type of layers utilized to construct such architectures, adapted from Saha (2018) .*

Feature extraction consists of a set of stages of convolutional and pooling layers. The number of stages generally depends on the complexity of the application and required number of features to be learnt; a basic architecture could be expected to have between one and three stages (LeCun et al. 2010). The convolutional layer, depending on its position within the network, will either accept images or features maps, following which the convolution operation is implemented by means of the set of filters or kernels. Part of this process incorporates the implementation of a non-linear activation function, discussed in more detail in the next

section. The second layer in each stage is known as pooling, and acts to reduce the spacial dimensions of the output while still preserving the most important feature information (Goodfellow et al. 2016).

The final set of feature maps, resulting from the feature extraction process, are flattened and fed as the input into a set of fully connected neural network layers, which act to perform the classification task (LeCun et al. 2010).

### Convolutional Layer

The convolutional layer acts by undertaking several convolutions simultaneously, producing a set of linear activations. As mentioned previously, the convolutional layer accepts either images or feature maps, depending on the network layer in question. The first layer of the network accepts the actual input data, generally in the form of images, while the next layers accept feature maps generated by the prior layer (Goodfellow et al. 2016). Hyper-parameters such as padding, stride and activation function are an important factor in defining the overall performance and characteristics of the convolutional layer.

The implementation of a convolutional layer can be explained more clearly using Figure 5.7.

In this case, an image of dimensions $n_W^{[l-1]} \times n_H^{[l-1]} \times n_C^{[l-1]}$ is considered as the input to the network, where $n_W^{[l-1]}$ and $n_H^{[l-1]}$ represent the width and height respectively while $n_C^{[l-1]}$ represents the depth or number of channels in the image. The layer index is defined by $l$, where $l-1$ represents the convolutional layer prior to the output. A typical example would be a three channel image with $RGB$ dimensions.

The convolution of the input image is then implemented using a set of $K$ filters or kernels. The filter height ($f_H$) and width ($f_H$) dimensions can be considered as one of the hyper-parameters for the convolutional layer. The depth ($f_C$) of the

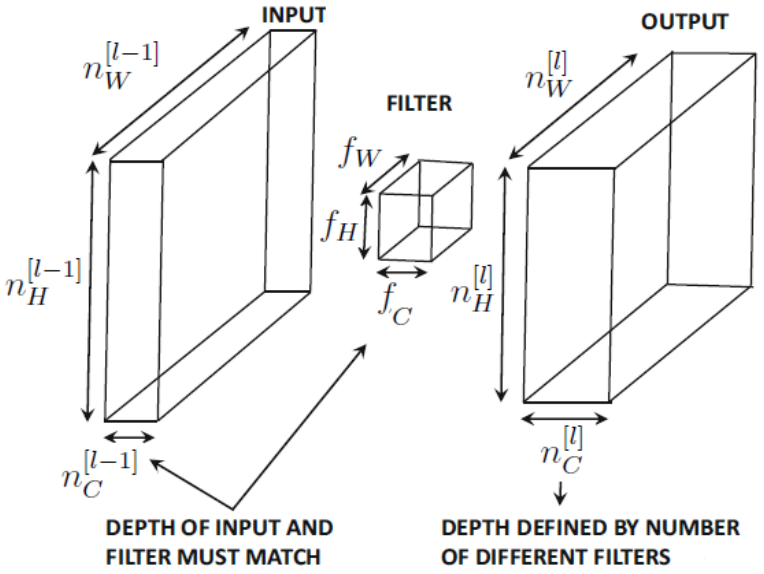


Figure 5.7: *Convolutional layer, adapted from Aggarwal (2018) .*

filter should be such that it exactly matches that of the input image ($n_C^{[l-1]} = f_C$).

The convolutional operation can then be thought of as the simple dot product over the entire volume of each filter, performed for each valid spacial position on the input image (Goodfellow et al. 2016). The output feature map has dimensions $n_W^{[l]}$ and $n_H^{[l]}$ representing the width and height respectively, while the depth $n_C^{[l]}$ is defined as the number of filters ($f_C$) applied to that image.

The elements in each filter are treated as model parameters, learnt during the training process. Each filter and its respective set of elements can be optimized to learn specific features from the input image or feature map to which it is applied. The learnt features become progressively more complex as the number of layers and overall complexity of the network architecture increases (LeCun et al. 2010).

The example provided in Figure 5.8 demonstrates how a specific filter has been defined to provide the ability to detect horizontal edges. The nature of the dot product sum means that as this filter passes over areas containing vertical edges

or homogeneous pixels, the filter output will be low (the sum of the top row of filter elements will effectively cancel out the sum of the bottom row of elements). However, as the element passes over horizontal edges, the difference in pixels between the upper and lower regions applied by the filter will create a significant output, being either positive or negative. The resulting feature map for this specific filter will thus provide information relating to all horizontal edges.

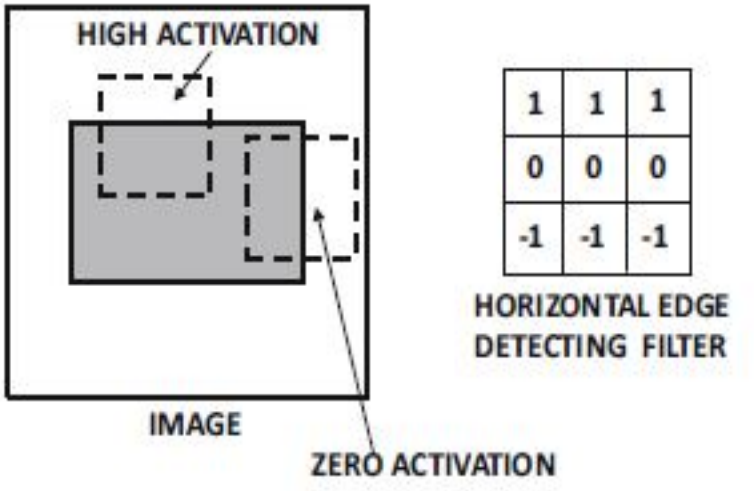


Figure 5.8: *Example of a filter for detecting horizontal edges, adapted from Aggarwal (2018) .*

The process of convolving on an image leads to an output feature map with reduced dimensions compared to the input, along with a potential loss in information as a result of the shrinkage. When considering unit strides during the convolution operation, the kernel effectively just slides across every position of the input (Dumoulin & Visin 2016). If a square input is considered ($n_H = n_W = i$), along with a square filter or kernel ($f_H = f_W = k$) it is then possible for the following relationship to be inferred for the square output dimension:

$$o = (i - k) + 1 \tag{5.6}$$

where the size of the output ($o$) will be equal to the number of steps or strides in each axis plus one (Dumoulin & Visin 2016).

In order to avoid this reduced set of output dimensions, a process known as zero padding is usually implemented (Goodfellow et al. 2016). Padding can be defined as the process of appending $p$ rows and $p$ columns of zeros to the input matrix (Kong & Lucey 2017). Padding provides a tool to avoid the potential loss of information that might occur as a result of the under-representation of the border pixels in the model (Goodfellow et al. 2016).

Assuming a square input matrix and kernel; if the input is padded with $p$ zeros the effective input dimensions for each size of the square matrix increase from $i$ to $i + 2p$. This relationship can then be expressed using:

$$o = (i - k) + 2p + 1 \tag{5.7}$$

where the output dimension ($o$) can be adjusted by changing the number of rows and columns of zeros ($p$) appended to the input image.

There are generally two instances of zero padding used extensively in convolutional layers due to their respective properties (Dumoulin & Visin 2016). The first instance type, commonly known as *half* padding, provides an output with the same size as the input ($o = i$). This property can be expressed as follows (Dumoulin & Visin 2016):

For any $i$ and for $k$ odd ($k = 2n + 1$, n $\in \mathbb{N}$), $s = 1$ and $p = [k/2] = n$,

$$\begin{aligned} o &= i + 2[k/2] - (k - 1), \\ &= i + 2n - 2n, \\ &= i \end{aligned} \tag{5.8}$$

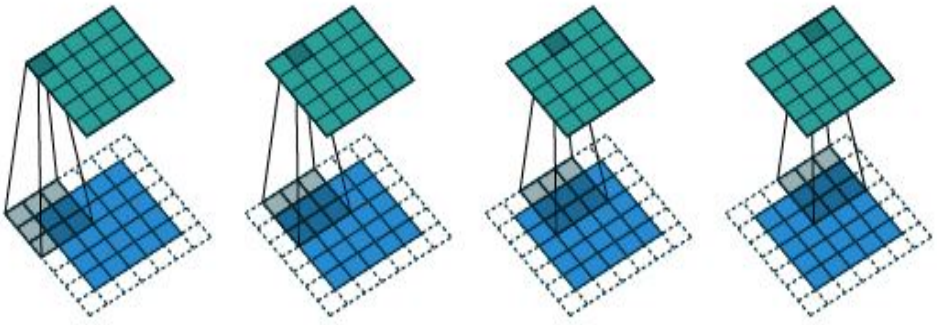

Figure 5.9: *Example of half padding - $3 \times 3$ kernel is convolved over a $5 \times 5$ input with unit strides, producing an output feature map with equivalent dimensions to the input image, adapted from Dumoulin & Visin (2016).*

The second commonly used padding type is known as *full* padding. By implementing this type of padding, it is possible to generate every possible partial superimposition of the kernel on the input image or feature map "the partial superimposition refers to the overlaying of he kernel onto every possible grid position of the input image" (Dumoulin & Visin 2016), and by doing so avoid potentially losing feature details along the edges which might otherwise be lost with less or no padding. It also allows the output feature map to have larger dimensions than the input.

This relationship can be described as follows:

For any $i$ and $k$, and for $p = k - 1$ and $s = 1$,

$$\begin{aligned} o &= i + 2(k-1) - (k-1), \\ &= i + (k-1) \end{aligned} \tag{5.9}$$

Along with the type and amount of padding, defining the stride of the convolution is another important hyper-parameter within the convolutional layer. The stride can be thought of as the resolution at which the kernel convolves across the input image or feature map. A unit stride is often implemented, meaning that the kernel

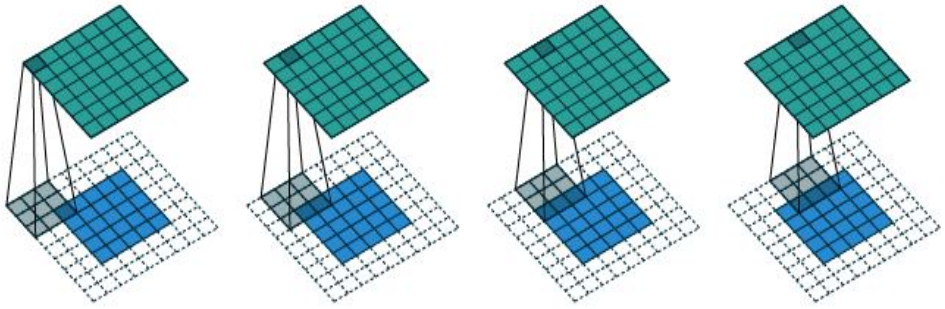

Figure 5.10: *Example of full padding - $3 \times 3$ kernel is convolved over a $5 \times 5$ input with unit strides, producing a $7 \times 7$ output feature map, adapted from Dumoulin & Visin (2016).*

"slides" across the input, convolving at each position along the input horizontal and vertical dimensions.

When considering a stride of $S_q$, the dimensions of the output feature map compared to the input is reduced by a factor of approximately $S_q$, and the area by $S_q^2$. This process effectively reduces the resolution of the output, along with an associated reduction in the number of model parameters to be learnt (Aggarwal 2018).

The activation function within a convolutional layer follows a similar process to that applied to a traditional neural network. In a traditional network, the activation function is combined with a linear transformation of a matrix of weights in order to produce the next layer of activations while assisting the network to learn more complex features in the input image or feature map. Similarly, in a convolutional layer, the activation function can be thought of as a non-linear operation performed on the product of the linear convolution operation (Aggarwal 2018).

The activation functions can be defined as follows:

$$\text{Sigmoid}: \quad \theta(x) = \frac{1}{(1+e^{-x})} = \frac{e^x}{e^x+1}, \tag{5.10}$$

$$\text{Tanh}: \quad \theta(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{5.11}$$

$$\text{ReLu}: \quad \theta(x) = \max(0, x) \tag{5.12}$$

Rectified Linear Unit (ReLu), shown in Eq.(5.12) has become one of the most popular activation functions in recent network designs. When compared to more traditional activation functions such as Sigmoid and Tanh, shown in Eq.(5.10) and Eq.(5.11) respectively, ReLu has several key advantages over its predecessors especially when considering convolutional based networks (Wang et al. 2020).

Functions such as Sigmoid and Tanh produce outputs ranging between 0 and 1, which creates the unwanted property known as vanishing gradients, created when the value of outputs multiplied across several layers during the back-propagation training process effectively shift the output to zero. ReLu does not suffer from this characteristic, and in addition provides a simpler and less computationally expensive solution as shown by Krizhevsky et al. (2012*b*). With this in mind it should also be noted that ReLu is not differentiable, and when active that portion of the network is just an over-determined linear system.

Recent studies and research have also provided several variations of the popular ReLu activation function. These attempt to address certain issues with this function such as "dying ReLu", causing some nodes to die and not learn anything as a result of all negative input values being assigned a value of zero (Wang et al. 2020).

### Pooling Layer

The pooling function is applied to the output of the convolutional layer with the primary purpose of reducing the spacial dimensions and complexity of the feature map to which it is applied, while still retaining the key feature properties and information. The pooling function shares some similarities with convolution; it applies a "window" or "filter", which moves across the input feature map with a resolution defined by the stride and window sizing. Unlike convolution however, this function acts on each input feature map independently, thus generating a set of output feature maps equal in number to the input feature maps (Aggarwal 2018).

The actual pooling function can also vary depending on the requirements of the specific application or data set. The two most common types of pooling are know as Max Pooling and Average Pooling (Lee et al. 2017). Max Pooling simply takes the maximum value within the pooling window at each position on the input feature map, while Average Pooling calculates the average value for each position. Apart from reducing the spacial complexity of feature maps, the implementation of pooling can also result in a degree of translational invariance. This property is what allows the CNN type architecture to classify images based on objects which may have very different relative locations (Aggarwal 2018).

### Fully Connected Layers

The fully connected layers are the final structural element of a standard Convolutional Neural Network architecture. The layers take the output of the feature extraction layer, as shown in Figure 5.7, and vectorizes the three dimensional array. This fully connected layer is responsible for the classification process and functions in exactly the same way a traditional feed forward network would (Aggarwal 2018), taking the vectorized feature map as an input, and providing a set

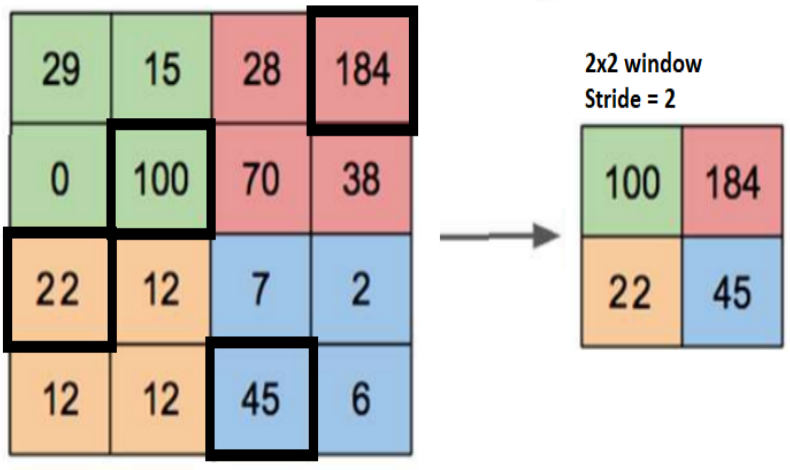


Figure 5.11: *Illustration of Max Pooling using a $2 \times 2$ sliding window with a stride of 2, applied to a $4 \times 4$ input feature map, resulting in a $2 \times 2$ output feature map. As illustrated by the darker outlined blocks, this technique extracts the largest value from each position of the window, adapted from Yani et al. (2019).*

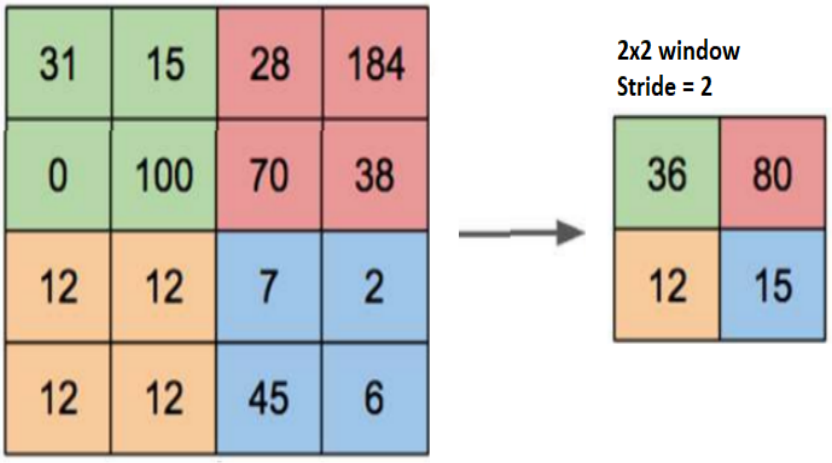


Figure 5.12: *Illustration of Average Pooling using a $2 \times 2$ sliding window with a stride of 2, applied to a $4 \times 4$ input feature map, resulting in a $2 \times 2$ output feature map. At each position of the window, the average value of the inputs is extracted, adapted from Yani et al. (2019).*

of predictions as an output. Due to the densely connected nature of the fully connected layer, the majority of model parameters lie within this layer, and careful consideration should be given to the size and number of layers as this plays a significant role in the overall complexity of the model and therefore the amount of training required.

The output layer may make use of functions such as softmax or linear activation, depending on the specific application (classification or regression). Using the case of image classification, a typical output layer will make use of the Softmax activation function which is generally used as the output of a classifier; with a number of neurons equivalent to the number of classes being predicted in the classification application.

The Softmax function represents the probability distribution of a given output over $n$ different classes, and because the output is required to lie between 0 and 1 as a probability per class, the log-likelihood is often used since it is more suited for the gradient based optimization process (Goodfellow et al. 2016).

A linear layer is first required to predict the log probabilities as follows:

$$z = \mathbf{W}h + \mathbf{b} \tag{5.13}$$

where ($\mathbf{W}$) is a matrix of weights and ($\mathbf{b}$) a vector of biases, while the the Softmax function is then used to exponentiate and normalize the required output $z$, creating predictions between 0 and 1, and summing to 1 across the $n$ classes. The Softmax function can be defined as follows:

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum \exp(z_i)} \tag{5.14}$$

Which can be represented in a log-likelihood format as follows:

$$\log \ \text{softmax}(z)_i = z_i - \log \sum \exp(z_i) \tag{5.15}$$

By representing the equation in a log-likelihood format, it ensures that the term $z_i$ contributes directly the cost function, and ensures that the term cannot saturate, which occurs when the function outputs values close to the asymptotic ends, damaging the overall learning capacity of the network (Goodfellow et al. 2016).

### 5.2.2 Convolutional Neural Network Learning Algorithm

The learning or training process of a CNN model involves the iterative optimization of a set of parameters, as well as the optimization of a set of hyper-parameters which are generally defined before training. The model parameters are calculated by iteratively minimizing the overall loss function using a gradient descent method. This method works by calculating the value of the loss function partial derivative with respect to the other parameters within the network, after which the results are fed back and the required parameter updates made in order to decrease the total loss, known as back propagation.

The algorithm known as backpropagation provides the gradients used in gradient descent. Following a random initialization of the model parameters, the algorithm implements the following steps: feed forward computation, backpropogation and updating the weights or parameters (Rojas 1996). The Algorithm 1 illustrates this process more clearly (Rojas 1996).

#### Feed Forward Computation

The first step of the backpropagation algorithm is the feed forward computation. This process, sometimes known as the "forward pass", presents an input vector to the network. The set of activations from each layer of the network are passed to the next, with the calculation defined as follows:

**Algorithm 1:** Backpropogation Algorithm

**Input :** Training Data

**Output:** Optimized model parameters

1 Randomly initialize parameters

2 **while** *iteration* $\geq$ *threshold or* $\Delta$ *error* $\leq$ *error threshold* **do**

3 Forward propagation

4 Backward propagation

5 Update model parameters

6 **end**

$$z^{(l)} = (\mathbf{W}a^{(l-1)} + \mathbf{b}) \tag{5.16}$$

$$a^{(l)} = \sigma(z^{(l)}) \tag{5.17}$$

where the linear activation of a layer $z^{(l)}$ in Eq.(5.16) is defined as the product of the activation value from the previous layer $a^{(l-1)}$ and the weight matrix $\mathbf{W}$. This product is then summed with the bias value $\mathbf{b}$. This is all wrapped in the Sigmoid activation function defined as $\sigma$, providing the output activation function $z^{(l)}$ from Eq.(5.17).

This forward computation continues through the layers of the network until it reaches the output layer, where a prediction is generated. These predictions can then be compared directly to the actual target values of the training data, using what is known as a cost or loss function. In the case of classification applications, the loss function is generally defined as cross entropy under the inference framework of maximum likelihood. The log cross entropy loss is defined as follows:

$$\mathcal{L}_{log} = -\sum_{j} y^{(j)} log\, \sigma(\mathbf{o})^{(j)} \tag{5.18}$$

Where $y^{(j)}$ is the actual class output, $\mathbf{o}$ is the network output and $\sigma$ is the Softmax activation function required to provide a probability estimate (Janocha & Czarnecki 2017).

### Backpropagation

Backpropagation is an efficient method for calculating the gradients through the network layers. As the name suggests, it starts from the output layer and works backwards through the network, computing the gradients and allowing for the weight and bias parameters of each layer to be adjusted in order to minimize the overall loss function.

This method relies on partial derivatives to compute the relationship between changes in various parameters within the network and the effect on the overall loss function. The following formula provides an example of the partial derivative of the loss function with respect to the weight parameter for the final layer (Nielsen 2015):

$$\frac{\partial \mathcal{L}}{\partial w^{(l)}} = \frac{\partial \mathcal{L}}{\partial a^{(l)}} \frac{\partial a^{(l)}}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial w^{(l)}} = 2\left(a^{(l)} - y\right) \sigma^{'}\left(z^{(l)}\right) a^{(l-1)} \tag{5.19}$$

where $\frac{\partial \mathcal{L}}{\partial w^{(l)}}$ represents the partial derivative of the loss function with respect to the weight in the last layer ($L$). In other words, it computes the relative impact of the change in weight on the overall loss function of the model, while keeping the other parameters constant. Although $w^{(l)}$ is not directly found in the cost function, it is possible to first assess the impact on the linear activation as defined in Eq.(5.16) as $\frac{\partial z^{(l)}}{\partial w^{(l)}}$, after which the change of the linear activation relative to the sigmoid activation function from Eq.(5.17) as $\frac{\partial a^{(l)}}{\partial z^{(l)}}$ can be measured. Finally

this activation function output can be measured against the change in overall loss function as $\frac{\partial \mathcal{L}}{\partial w^{(l)}}$.

Similar to the partial derivative function defined for weights in Eq.(5.19), the following equations provide a similar definition for the bias and activation parameters of the model (Nielsen 2015):

$$\frac{\partial \mathcal{L}}{\partial b^{(l)}} = \frac{\partial \mathcal{L}}{\partial a^{(l)}} \frac{\partial a^{(l)}}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial b^{(l)}} \tag{5.20}$$

$$\frac{\partial \mathcal{L}}{\partial a^{(l-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(l)}} \frac{\partial a^{(l)}}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial a^{(l-1)}} \tag{5.21}$$

where $b^{(l)}$ represents the bias term in the last layer while $a^{(l)}$ and $a^{(l-1)}$ represent the activations in the last and second last layers respectively.

### Model Parameter Update

The final step in the backpropagation process is to update the weight and bias parameters by moving backwards through the network.

The backpropogation process effectively calculates the relative gradients for each change in weight and bias parameters (Rojas 1996). The model parameter update process uses this information to make adjustments to these parameters, moving backwards from the output of the network to reduce the overall loss value. The process of updating the model parameters in the opposite direction to the gradients defined using partial derivatives is known as gradient descent, and is commonly implemented in neural network algorithms.

Using each partial derivative result as defined in Eq.(5.19) and Eq.(5.20), a gradient vector can be defined with the number of dimensions equivalent to the total number of weights and biases as follows:

$$-\nabla C(w_1, b_1, ..., w_n, b_n) = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_1} \\ \frac{\partial \mathcal{L}}{\partial b^1} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial w_n} \\ \frac{\partial \mathcal{L}}{\partial b^n} \end{bmatrix} \tag{5.22}$$

In order to calculate the results derived by the gradient vector, three commonly used gradient descent variants are considered. Each of these variants differ in terms of the amount of data used to calculate the gradient function. The amount of data used for these updates is generally a trade-off between overall accuracy of the parameter updates, versus the time taken to perform the calculation (Ruder 2016).

The first of the gradient descent variants mentioned earlier in this chapter is known as the batch gradient descent. The algorithm calculates the loss function with respect to the parameters of the complete training set (Ruder 2016). By using the complete data set for calculations, this option tends to be slow and impractical for large data sets.

Another variant, known as mini-batch gradient descent, is the most commonly implemented version. The method utilizes subsets or batches of $n$ data from the total training data. By doing so the overall variance of parameter updates decreases significantly, while still providing a fast solution with stable convergence. Most modern deep learning libraries and implementations make use of this gradient descent variant, with batch sizes typically ranging between 50 to 256 depending on the specific application and overall training data set size (Ruder 2016).

The following simplistic formulae describe the process of updating the weights and biases in a certain layer using the results of the gradient vector calculated through the partial derivatives, and controlled by the magnitude of the learning

rate. In general, the gradient vector would have been calculated using a mini-batch gradient decent method:

$$w^{(l)*} = w^{(l)} - \text{learning rate} \times \frac{\partial \mathcal{L}}{\partial w^{(l)}} \tag{5.23}$$

$$b^{(l)*} = b^{(l)} - \text{learning rate} \times \frac{\partial \mathcal{L}}{\partial b^{(l)}} \tag{5.24}$$

where $w^{(l)}$ and $b^{(l)}$ represent the weight and and bias parameters in layer $l$ respectively.

In each case of the gradient descent algorithm, a learning rate is used to control the rate at which the parameters are updated. A very small learning rate would in turn mean that each update based on the gradient results would itself be small, and lead to a model which slowly converges towards a (local) minimum. The converse is also true, with a large learning rate resulting in faster convergence towards a (local) minimum. The risk however, with large learning rates, is that it causes the loss function to fluctuate around, or potentially diverge from the (local) minimum (Ruder 2016). It is for this reason that the learning rate is considered to be an important hyper-parameter in the design of a neural network architecture. Most modern architectures also provide the ability to adjust the learning rate during the process of training, allowing the model to initially converge quickly on the (local) minimum, and reduce this rate as it approaches this minimum to improve accuracy and avoid fluctuation and potential divergence.

While CNN type architectures can be used in a variety of relational data applications such as various computer vision and audio data tasks, this research will focus specifically on instance segmentation and auto encoders, used for object detection and unsupervised image clustering respectively.

With a conceptual and theoretical overview of the general CNN architecture complete, it is important to explore the specific applications used for this research, starting with instance segmentation in the next section.

### 5.2.3 Image Segmentation

This research aims to provide a solution for the clustering and eventual classification of rocks according to their facie origin. This approach should assist mining process plants effectively monitor the ratio in which these rock types enter the plant.

The initial approach to this application was the direct implementation of an unsupervised clustering method on the original camera images. This was based on the assumption that the vast majority of rocks contained within a single image belonged to the same class or facie. As the research developed, and through consultation with the local geology team, it became evident that the problem was not as straight forward, and each image in fact contained a variety of rock types. A revised approach was required, in which individual rocks within an image are first identified and then extracted, after which the clustering method is applied, and the results summarized.

As a first step, the solution needs to identify and segment the individual rocks within an image, to allow for further processing and clustering of each rock. Image segmentation can be grouped into three general classes. Object detection identifies all objects within an image, outputting the region and class type of each object. Semantic segmentation assigns classes to specific pixels, grouping each different class type, while instance segmentation groups pixels of each unique instance within a class, and can be considered as the most granular form of segmentation.

This first step requires the implementation of an instance segmentation model, which generates a pixel wise mask for each unique rock identified in an image. The method implemented for this research is known as Mask R-CNN, and is explored in detail in the following section.

Several other methods for image segmentation were considered, including simplistic region-based segmentation such as threshold segmentation and edge detection, both of which use pixel values to identify sharp contrasts between objects and backgrounds. More advanced methods such as Faster R-CNN were also investigated. The requirement for pixel wise segmentation, along with the state of the art results on multiple benchmark image data-sets, meant that Mask R-CNN was selected as the preferred segmentation method for this research.

### Mask Region-based Convolutional Neural Network (Mask R-CNN)

Mask R-CNN method was developed as an extension of the Faster R-CNN model which is used for semantic segmentation. By incorporating a branch for predicting segmentation masks on each of the Regions of Interest (RoI), in parallel with the existing branches for classification and bounding box regression, this method is able to provide a relatively simple solution to implement instance segmentation (He et al. 2017). The three outputs from a Mask R-CNN model include a pixel wise instance mask, a bounding box and class type prediction.

As shown in Figure 5.13, the input image is first passed through a backbone CNN architecture, which can be optimized based on the specific application or use case. The two most common architectures are Resnet 50 and Resnet 101, with the latter being utilized for this research due to its superior performance for this application. The output from the CNN backbone is a set of low and high level feature maps used to identify features such as edges, texture and shapes.

These outputs are then fed into the Regional Proposal Network (RPN), which

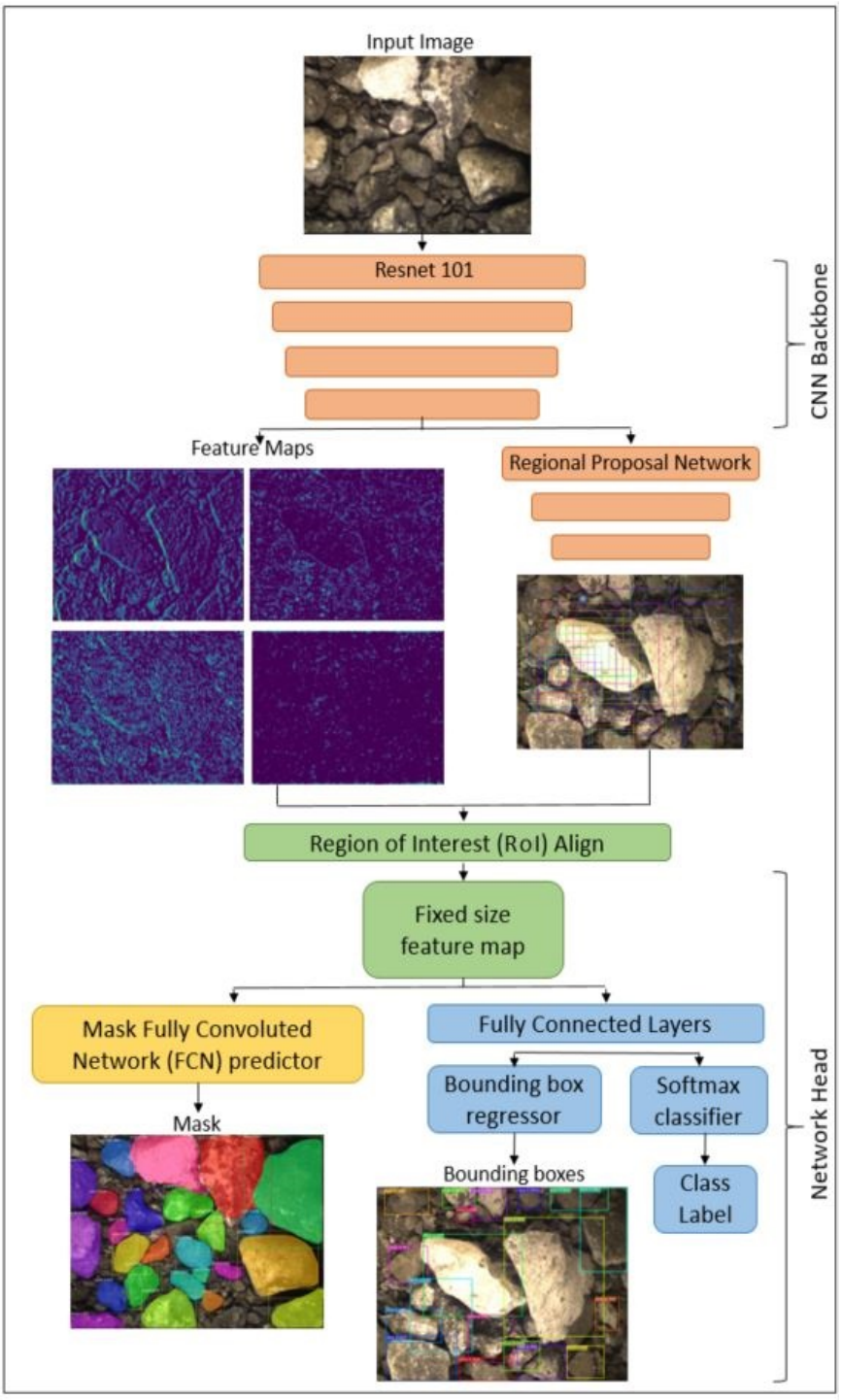


Figure 5.13: *The Mask R-CNN model architecture utilized for this research, including a Resnet 101 CNN backbone. The outputs include pixel wise masks, bounding boxes and class type predictions.*

scans the feature maps, identifying regions which are most likely to contain a targeted object otherwise known as Region of Interest (RoI) or proposals. A set of overlapping boxes of varying size and aspect ratio are generated (He et al. 2017). The boxes are known as anchors, and can be visualised as shown in the example rock image from Figure 5.13.

The RoI Align operation is then applied to the set of feature maps and anchors. It generates a fixed size feature map with reduced levels of misalignment between the feature maps and RoI's defined in the RPN network when compared to operations making use of quantization and binning. This feature map is then fed into a fully convolutional network (FCN), used to generate pixel wise predictions for objects masks. The feature map is also introduced into a set of fully connected layers. These are used to predict the bounding boxes for each object through the use of a bounding box regressor, as well as predict the class label for each object using a Softmax classifier.

This model is optimized according to a parallel multi-task log loss of classification ($\mathcal{L}_{\text{class}}$), localization ($\mathcal{L}_{\text{box}}$) and segmentation mask ($\mathcal{L}_{\text{mask}}$) as follows (He et al. 2017):

$$\mathcal{L} = \mathcal{L}_{\text{class}} + \mathcal{L}_{\text{box}} + \mathcal{L}_{\text{mask}} \tag{5.25}$$

The parallel nature of the loss function differentiates this model type from other classification methods which generally depend on the masks for the classification process. By dissociating the class and mask predictions, superior instance segmentation results have been achieved (He et al. 2017).

The inclusion of a pixel-to-pixel alignment operation, commonly known as RoIAlign is a critical component, required to allow for the more detailed processing of the spatial layout of an object required for pixel wise masks (He et al. 2017).

The RoIAlign operation is implemented in favour of the generally used RoIPool operation. The RoIPool operation performs a series of quantization and binning procedures on the RoI feature map. This process results in misalignments in the form of translations between the RoI and the extracted feature map, which introduce inaccuracies when considering pixel wise masking. RoIAlign removes the need for quantization for data pooling. RoiAlign instead makes use of bi-linear interpolation (He et al. 2017). This process divides the RoI region up into a set of blocks determined by the size of the RoI as well as the size of the pooling layer. Within each of these newly specified blocks, four sampling points are selected. The values at each of these points is calculated by considering the relative distance of that point to its closest neighbouring cells (in this case pixels). The values of its surrounding cells are incorporated into the calculation in a proportion relative to the proximity to the point considered. This process provides an alternative to quantization, which is associated with a loss in information.

In this research, a successfully trained Mask R-CNN model is able to accurately segment each rock object in an image, utilizing both bounding boxes and pixel wise masking. By doing so, it is then possible to generate a set of new images, each containing just a single rock, which can then be input into the next phase of the modelling pipeline. This next phase will utilize an unsupervised method to cluster the sets of rock images, and eventually provide a method by which to analyze the number of rocks belonging to various types per image.

The second phase clustering method, known as Deep Embedded Clustering (DEC), is discussed in more detail in the next section.

### 5.2.4 Deep Embedded Clustering

Deep Embedded Clustering (DEC) provides an unsupervised method based on deep neural network architectures, which simultaneously learns reduced dimen-

sion feature spaces while also providing cluster assignment on these learnt features.

Unlike more conventional clustering algorithms such as K-Means and Hierarchical clustering, which generally leverage linear distances between data-points in the original feature space, DEC provides a non-linear mapping of the original feature space to a lower dimensional space (Xie et al. 2016).

Several previous research papers within the field of rock classification have relied on the ability of the researcher to develop a set of user defined features, such as colour and texture, on which to train a classification model (Perez et al. 2015*b*) (Patel et al. 2017). A significant advantage of utilizing a method incorporating a learnt feature space is the automatic generation of these features which requires no external input or domain expertise, a key consideration for this application.

The process of pre-training the auto-encoder, followed by the optimization of the clustering based on the learnt feature space can be visualized in Figure 5.14.

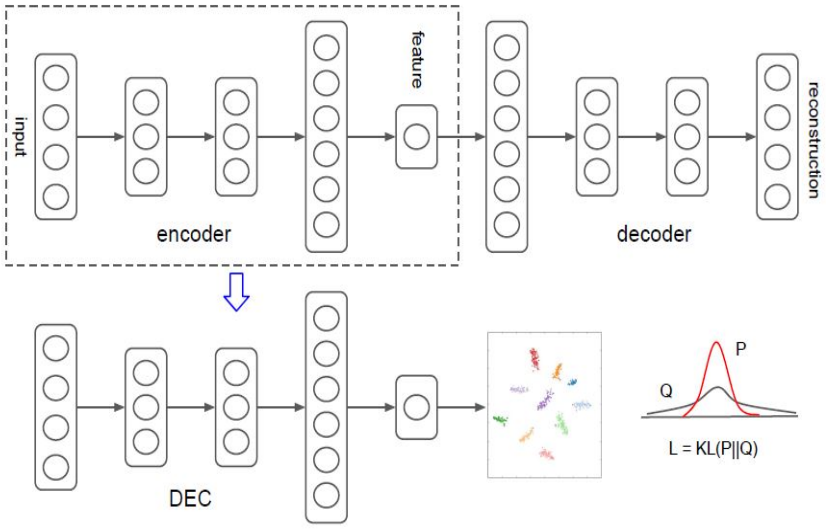


Figure 5.14: *Two stage optimization process of the DEC method; training of an auto-encoder, followed by the optimization of a clustering method utilizing the learnt feature space and KL distribution, adapted from Xie et al. (2016).*

The implementation of DEC in the research conducted by (Xie et al. 2016) provides a joint optimization of deep embedding and clustering. The deep embedding and feature space generation is implemented using a fully connected Autoencoder, while clustering on this learnt feature space is optimized by minimizing the Kullback-Liebler (KL) divergence between data distribution and embedded distribution (Xie et al. 2016).

Clustering using KL divergence and DEC alternates between two steps. First each embedded data point is soft assigned to a centroid cluster, after which the deep mapping is updated and the centroids adjusted based on the high confidence assignments (Xie et al. 2016). This process is iterated until a convergence target is met.

The probability of each embedded point $z_i$ belonging to cluster $j$ can be represented as follows (Van der Maaten & Hinton 2008):

$$q_{ij} = \frac{(1 + ||z_i - \mu_j||^2)^{-1}}{\sum_j (1 + ||z_i - \mu_j||^2)^{-1}} \tag{5.26}$$

Where the clustering layer maintain centers $\mu_j$ as trainable weights (Guo et al. 2017).

The loss function for KL divergence minimization can then be defined as follows (Xie et al. 2016):

$$\mathcal{L} = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{5.27}$$

Where the loss $\mathcal{L}$ is defined as the loss between the soft assignments $q_{ij}$ and auxiliary distribution $p_{ij}$ for sample $i$ in cluster $j$. The target distribution is defined in order to help strengthen assignments and node purity. It also acts to emphasize data points assigned with high confidence, as well as normalize the

loss contribution of each centroid, and by doing so prevents large clusters from distorting the reduced dimension feature space (Xie et al. 2016).

The joint optimization of both the feature space embedding and cluster centroid assignment can be defined as follows for KL Divergence (Xie et al. 2016):

$$\frac{\partial \mathcal{L}}{\partial z_i} = \frac{\alpha+1}{\alpha} \sum_j \left(1 + \frac{||z_i - \mu_j||^2}{\alpha}\right)^{-1} (p_{ij} - q_{ij})(z_i - \mu_j) \tag{5.28}$$

$$\frac{\partial \mathcal{L}}{\partial \mu_j} = -\frac{\alpha+1}{\alpha} \sum_i \left(1 + \frac{||z_i - \mu_j||^2}{\alpha}\right)^{-1} (p_{ij} - q_{ij})(z_i - \mu_j) \tag{5.29}$$

Where $z_i$ and $\mu_j$ represent the feature space embedding and cluster centroid respectively for each data point. The parameter $\alpha$ is defined as the degrees of freedom of the distribution. Due to the unsupervised nature of this method, optimizing these parameters is not feasible, and for the sake of the research these parameters are set to one, as recommended by Xie et al. (2016). The probability of assigning data-point $i$ to cluster $j$ is defined as $q_{ij}$, while the $p_{ij}$ is computed as the frequency at which the datapoint is assigned to a specific cluster.

Since the development of the DEC method in 2016, several research papers have been released which provide application specific improvements to the original method. DEC with convolutional auto-encoders was proposed by Guo et al. (2017), otherwise known as Deep Convolutional Embedded Clustering (DCEC), which effectively replaces the dense neural network auto-encoder with a CNN type architecture. The Convolutional Auto-encoder (CAE) proved to provide a superior solution by leveraging spacial relationships inherent in image data. Another recent implementation by Ren et al. (2019*b*), provided a semi-supervised deep embedded clustering architecture. This method included a set of pairwise constraints introduced to the feature space based on a labelled subset of data. By including a labeled subset, the method was able to learn high probability

assignments quickly, as well as utilize both the semi-supervised loss as well as KL divergence loss.

A recent research paper by Guo et al. (2018), on which this research implementation is based, is known as deep embedded clustering with data augmentation (DEC-DA). This method provides a state of the art solution based on benchmark image data sets, by incorporating robust image augmentation as well as a convolutional auto-encoder architecture. By implementing data augmentation during the unsupervised feature learning process, the model is able to provide a set of more generalized features. This is a critically important consideration for this research, which without augmentation might produce oversimplified features such as individual rock shapes or orientations. It is important to differentiate between pre-processing of the images to create a larger dataset using data augmentation versus this methods use of batch by batch augmentation during training.

The DEC-DA method actually replaces the original training image with a randomly augmented version for each iteration or epoch during training, therefore the model never sees an identical version of the original image during training, allowing for much better generalization and avoiding over simplified feature selection. This is in contrast to generating more training data before inputting into the model, which effectively just increases the total data available, but does not effectively avoid extraction of oversimplified features, which would tend to happen as the model sees the same image for each iteration of training.

The overall loss function for the DEC-DA model during fine-tuning can be defined as follows (Guo et al. 2018):

$$\mathcal{L} = \alpha\mathcal{L}_r + \beta\mathcal{L}_c \tag{5.30}$$

Where $\mathcal{L}_r$ is defined as the reconstruction loss, otherwise known as the auto-

encoders ability to reconstruct the original image for this application and $(\alpha, \beta)$ are coefficients to balance the loss functions. The $\mathcal{L}_c$ is defined as the clustering loss, derived using either $k$-means or KL divergence (defined in Eq.5.28 and Eq.5.29).

The clustering loss can be further generalized as:

$$\mathcal{L} = dist(y, t) \tag{5.31}$$

Where $y$ and $t$ are the actual and target distributions respectively, while the loss is defined as the distance between them. The target distributions are defined using the non-augmented data, while the actual outputs are derived from the augmented data. In order to understand this, it is important to consider the two stages of this model type separately.

The DEC-DA algorithm can be summarised using pseudo code as defined in Algorithm 2.

**Algorithm 2:** Deep Embedded Clustering using Data Augmentation

**Input :** Training Data X; Augmentation; Number of Clusters K

**Output:** Cluster Assignment

```
Pretrain the autoencoder using augmented data;
Initialize cluster centers, s, and target distribution t;
while Stopping criteria not met do
    Update network weights using Eq 5.30, with t fixed;
    Update s and t with fixed network weights;
end
```

The initial stage of pre-training the auto-encoder is implemented using the augmented data set. By doing so it allows the unsupervised method to learn gener-

alised features, avoiding an oversimplified feature space which might include rock shape and size when considering features relevant to this research.

The second stage involves concurrently optimizing the clustering loss, as well as the reconstruction loss, by using the learnt feature space derived during the pre-training stage. The optimization of the clustering loss can be thought of as supervised type problem, where the actual output $y$ is generated using the augmented data, while the target $t$ is defined using the original data and can be thought of as the centre of the cluster to which an input has been assigned. In much the same way that evaluating a supervised learning problem on the test set is done without augmenting that test set, the clustering performance in this case is evaluated using the target distribution of the features extracted from the original data. This process can be visualized as shown in Figure 5.15.

The stopping criteria used during the training process was based on max-gradient, with the limit being set at an $\epsilon$ of 0,001. This stopping criteria is defined from a composite of the class, mask and RPN losses, creating an overall validation log-loss on which the max-gradient is applied. The results of which are discussed in detail in Chapter 7, with a trace plot provided in Figure 7.1.

Having discussed the model types required in each phase of this research, it is then important to clearly define how these models are used and implemented in this research. The next section will outline the required environment setup and data pre-processing, followed by a detailed discussion of the results.

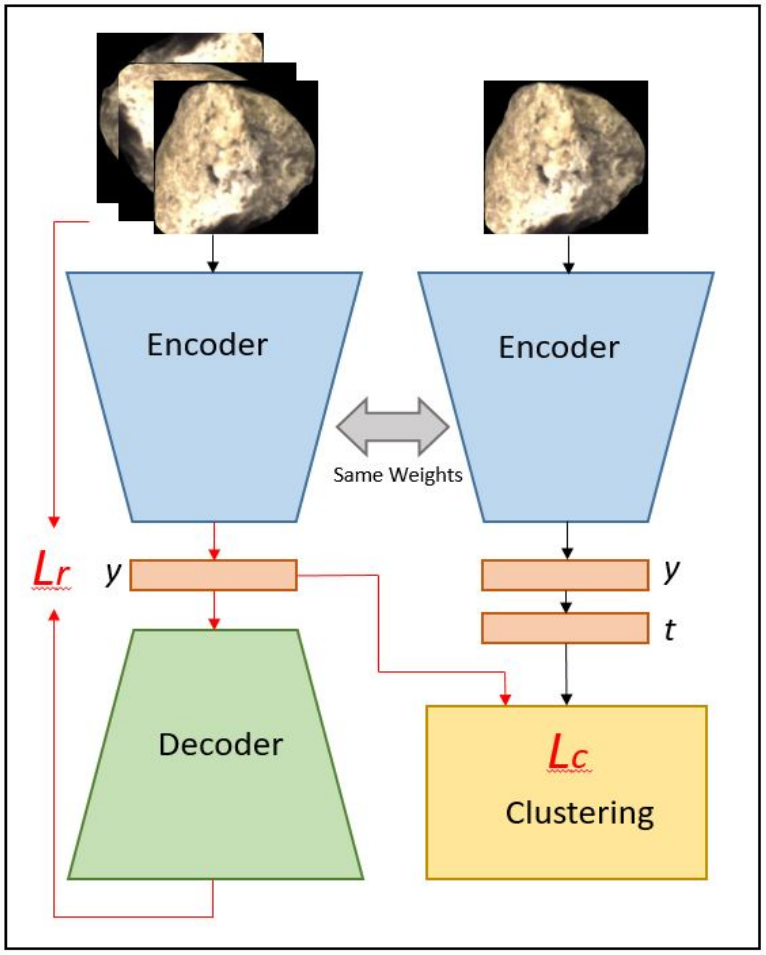


Figure 5.15: *DEC-DA model overview sketch showing the generation of the target distribution (t) using set of features from original data, along with predicted outputs (y) making use of the set of features generated using augmentation. The red lines present the path of iterative weight updates during fine tuning, based on the defined loss functions, adapted from Guo et al. (2018).*

# Chapter 6

# Implementation

This chapter focuses on providing a detailed overview of the practical implementation of the modelling pipeline. Both phases of the computer vision models are discussed, including the environment setup, data pre-processing and model evaluation metrics.

Before the implementation of each phase of modelling can be discussed, it is important to define the general computing environment setup for this research, as described in the next subsection.

## 6.1 Environment Set-up

While the image training data was labelled on a local machine using the Labelme.py Python package, the subsequent modeling for Phase 1 and 2 was conducted on the Google Colab platform, which provides a free cloud based application on which Python notebooks can be run and implemented. The platform provides free access to a GPU resource, and is thus a popular choice for data processing and machine learning. The specifications of which are provided in Table 6.1.

Both phases of modelling conducted as part of this research were implemented in Python programming language (Van Rossum & Drake Jr 1995).

Table 6.1: Google Colab Cloud Platform computing environment specifications.

| Environment Setup | |
|---|---|
| Specification | Description |
| Cloud Service | Google Colab |
| CPU Clock | 2.3 GHz |
| GPU | NVIDIA Tesla K80 |
| CUDA cores | 2496 |
| GPU Memory | 12GB |

## 6.2 Rock Instance Segmentation Phase

As discussed in previous chapters, due to the presence of several rock types from different facies in each image, a first phase of modelling is required to identify individual rocks. This phase, known as instance segmentation, is described in this section. The data preparation and evaluation metrics for this modelling phase are covered in detail.

### 6.2.1 Data Preparation

The collection and storage of data from Kao Diamond Mine is covered in detail in Chapter 4. A total of 25,000 images were captured and stored on a remote server managed by Stone Three Mining. The image data was pulled directly from this server for data pre-processing and modelling phases.

A total of 120 images, with resolution 992x662 pixels, were labelled using the Labelme.py Python package. Each image generally contained between 30 to 40

labelled rock instances, resulting in a total of approximately 4,200 labelled rock instances of various facie type and shape. The test and training split is summarized in Table 6.2. Images were randomly selected across the full range of time stamps for the complete data set. By doing so, the training data should provide a representative sample of all plant feed material typical for this mining operation. This step is important to ensure the model generalizes well on unseen images, once put into production.

Table 6.2: Instance segmentation dataset

| Dataset | Training | Testing |
|---|---|---|
| Conveyor Feed Images | 100 (84%) | 20 (16%) |

The estimated number of training images required for this modelling phase was based on findings in other research papers, most notably a similar research paper using Mask R-CNN and labelled rock instances to calculated the particle size distribution for rock stockpiles in mining applications (Schenk et al. 2019). This research suggested 1,000 individually labelled rock fragments with data augmentation.

With the training and test data set prepared, the computer vision model for instance segmentation is configured and implemented, as discussed in the next subsection.

### 6.2.2 Model Configuration and Training

As with most machine learning models and computer vision applications, several hyper-parameters and decisions relating to the overall model architecture and configuration are required. Each of these user defined inputs contributes to the overall performance and accuracy of the resulting model.

The Mask R-CNN model, used for the instance segmentation phase of this research, is relatively complex, with millions of trainable parameters. The number of trainable parameters in any deep neural network generally requires a very large training set in order to avoid over-fitting the model. Due to the time and resource constraints of this research, collecting and labelling a massive data set is not a viable option. A method called transfer learning was thus implemented. A set of pre-trained weights for the Mask R-CNN model is used, generated through the training of the model backbone on several thousand images from the Microsoft Common Objects in Context (MS COCO) data set (Lin et al. 2014). The model head is then subsequently trained using the limited data set of 100 rock images. By using this process, the model is able to leverage generalized features learnt from the large scale MS COCO data set, after which application specific complex features of the rock image data set can be learnt as a fine-tuning and final training step.

The use of transfer learning is common practise within computer vision applications and provides the ability to leverage features learnt from a large scale database, while having relatively few training images in the actual data set (Li et al. 2020). There are several large scale training data sets freely available, and a second commonly utilized data set for transfer learning is known as ImageNet (Deng et al. 2009), which contains approximately 14 million labelled images across 22,000 categories.

Implementing transfer learning, using the MS COCO dataset, showed improved results based on test log-loss, when compared to implementation using equivalent model hyper-parameters and the ImageNet database, discussed in more detail in the results section.

A second major model configuration decision is the selection of an optimal backbone architecture or network. This backbone network refers to the convolutional

network architecture of the Mask R-CNN model. The two most common backbone architectures for this model type include the Residual Networks (ResNet) 50 and 101 (He et al. 2016). For the purpose of this research, both of these architectures were implemented based on pre-trained MS COCO weights. The ResNet 101 architecture was found to perform better, and therefore selected as the final solution, discussed in detail in the Results section. It is important to note however, that the high level of complexity of this solution, with 101 layers of deep convolutional network, might not provide a practical solution for implementation at a mining plant. The large size of the model would make live computer vision applications very challenging. In this case, the smaller ResNet model might be a preferred solution for the final implementation. There would need to be a trade-off between acceptable model accuracy versus model size and available computing resources.

Table 6.3: Model configuration and architecture for instance segmentation phase.

| Mask R-CNN Model | |
|---|---|
| Architecture | Description |
| Backbone network | Resnet 101 |
| Number of Classes | 1 |
| RPN Proposal Threshold | 0.7 |
| Minimum Detection Confidence | 0.7 |
| Learning Rate | 0.001 |
| Learning Momentum | 0.9 |

### 6.2.3 Evaluation Metrics

The evaluation of the rock instance segmentation phase involves the consideration of three parallel loss functions, namely classification, localization and segmenta-

tion. The combination of these three functions provides a single loss output function known as the log-loss. In addition to the log-loss, mean Average Precision (mAP) is used to assess the relative performance of the Mask R-CNN image segmentation model.

### Log Loss

Log Loss, otherwise known as multitask loss can be defined as follows (He et al. 2017):

$$\mathcal{L} = \mathcal{L}_{\text{class}} + \mathcal{L}_{\text{box}} + \mathcal{L}_{\text{mask}} \tag{6.1}$$

This loss function is jointly applied to each RoI in the training process. For each RoI, a target class $u$ is defined, along with a ground truth bounding box regression target $v$, and by using the multi task loss function, this model type can be jointly trained for classification and bounding box regression (Girshick 2015).

Classification Loss, represented as $\mathcal{L}_{\text{class}}$, can be defined as follows (He et al. 2017):

$$\mathcal{L}_{\text{class}}(p, u) = -\log p_u \tag{6.2}$$

Representing the cross entropy or log loss for the true class $u$.

The bounding box loss ($\mathcal{L}_{\text{box}}$) is defined as a tuple over the true bounding box regression targets for $u$ and $v$ (Girshick 2015). This loss can be further defined as the difference between the predicted tuple of class $u$, defined as $t^u = (t^u_x, t^u_y, t^u_w, t^u_h)$, and the actual bounding box regression target of $v = (v_x, v_y, v_w, v_h)$. Figure 6.1 provides an example of a typical bounding box in the context of the rock segmentation data set. The figure shows the typical origin point $(0, 0)$, which is always defined in the top left corner of the image when using programming

languages such as Python. The centre of the box has coordinates $(x, y)$, while the height and width of the bounding box can be expressed using $h$ and $w$ respectively.

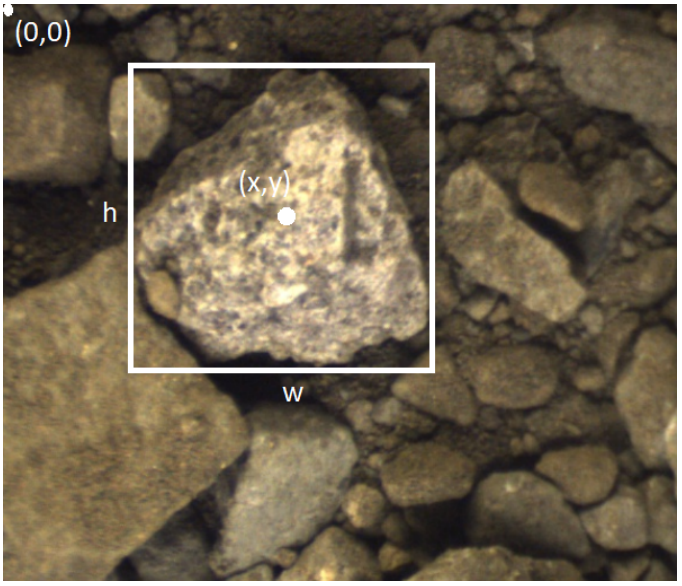


Figure 6.1: *Example of a bounding anchor box with dimensions $(h, w)$, and an anchor position of $(x, y)$*

Finally, the Mask loss ($\mathcal{L}_{\text{mask}}$) can be defined as the average binary cross-entropy loss (He et al. 2017). The Mask R-CNN model is structured in such a way as to inherently decouple the mask generation from class prediction, and by doing so allowing masks for every class to be generated without competition amongst classes. This effectively means that a binary mask of dimensions $m \times m$ is generated for each class in total number of classes $k$, thus generating an output mask layer of size $k \times m^2$ for each RoI.

### Mean Average Precision

Mean Average Precision (mAP) is a similarity metric, and in the context of object detection, can be thought of as a measure of the accuracy of each detection. Average Precision computes an output in the range of 0 to 1, representing the average precision value for the defined recall value.

The concept of precision and recall can be defined as follows (Zhu 2004);

$$Precision = \frac{TP}{TP + FP} \tag{6.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{6.4}$$

where precision computes the ratio of true positive (TP) occurrences versus the total number of positive predictions, both true and false (FP), while recall computes the ratio of true positive cases to actual true cases, which include true positive as well as false negative (FN). In other words, the average precision measures how accurate your predictions are (precision) based on the actual positive cases identified by the model (recall) (Zhu 2004).

High values of mAP on the validation and test set suggest the model is performing well in terms of making detections for that given recall threshold value.

Before the Average Precision can be determined, it is first necessary to define how a prediction is considered as a true positive case. This requires the concept of Intersection over Union (IoU) which can be though of as the relative overlapping of the predicted bounding box on top of the ground truth bounding box in the case of instance segmentation (Rezatofighi et al. 2019), as shown in Figure 6.2. Values towards 1 indicate higher detection accuracy, and good overlap between the ground truth and prediction bounding boxes (Rezatofighi et al. 2019).

With the implementation of the instance segmentation phase covered, it is then important to define equivalent implementation metrics for the second phase of image clustering.

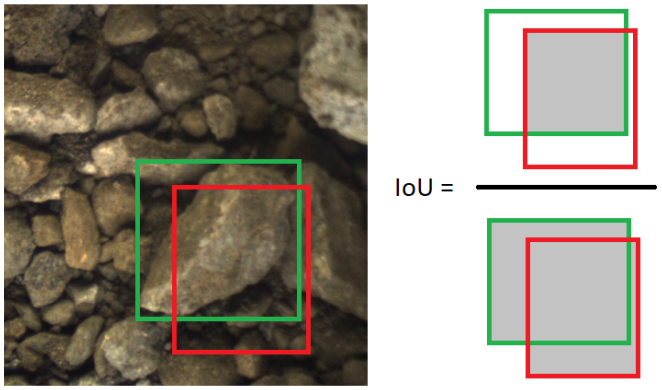


Figure 6.2: *IoU where the numerator is defined as the intersection between the ground truth and predicted bounding boxes, while the denominator is defined as the union of the two areas.*

## 6.3 Rock Clustering Phase

This section details the data preparation and evaluation metrics used for the second phase of modelling. The data input for this phase being the product or output of the previous segmentation phase, and thus completing the full modelling pipeline.

### 6.3.1 Data Preparation

The first phase segmentation model was applied to 2,000 unlabelled rock images captured across a representative time frame. The images included mining operations at different times of the day, as well as the processing of various mining facie. The output of this first phase included approximately 50,000 individual rock images. Each input image contained between 20 to 30 identifiable rocks, producing the same number of output images, with each rock having a black background applied to it through a masking process.

An important consideration for the individual rock data set is the inherent variability in terms of the size or number of pixels, as well as the shape of the rocks.

In order to provide a training data set for the second phase that utilized rocks with enough textural and colour information, a threshold of image size in pixels is defined to ensure low resolution rock images are not used to train the second phase model. This threshold was set at 20,000 pixels per image.

Figure 6.3 shows the distribution of the output image size in terms of number of pixels. The rock images shown as part of this graphic give an indication of range of image quality based on the total number of pixels per image, ranging from low to high.

Once selected, the subset of images are resized to 168x168 pixels. This pre-processing step is important to ensure images with equivalent dimensions are introduced into the DEC clustering model. The first input layer of the model has a pre-defined number of neurons set as a model parameter, which matches the common input dimensions of the images as a "matrix" of pixels - in this case 28,224 pixels.

At this point it is important to recognize the potential negative implications of this resizing step. Although rock shape should not play an important part in the clustering process, due to rotational data augmentation, the potential distortion as a result of resizing images could have a negative impact on the quality of image data related to attributes such as rock texture. This method was selected over cropping, as the potential for even greater losses of information for the larger images would be expected using cropping, which would require that all images are cropped to match the smallest image in the data set.

With the data pre-processing complete, the clustering modelling is configured and implemented as discussed in the next section.

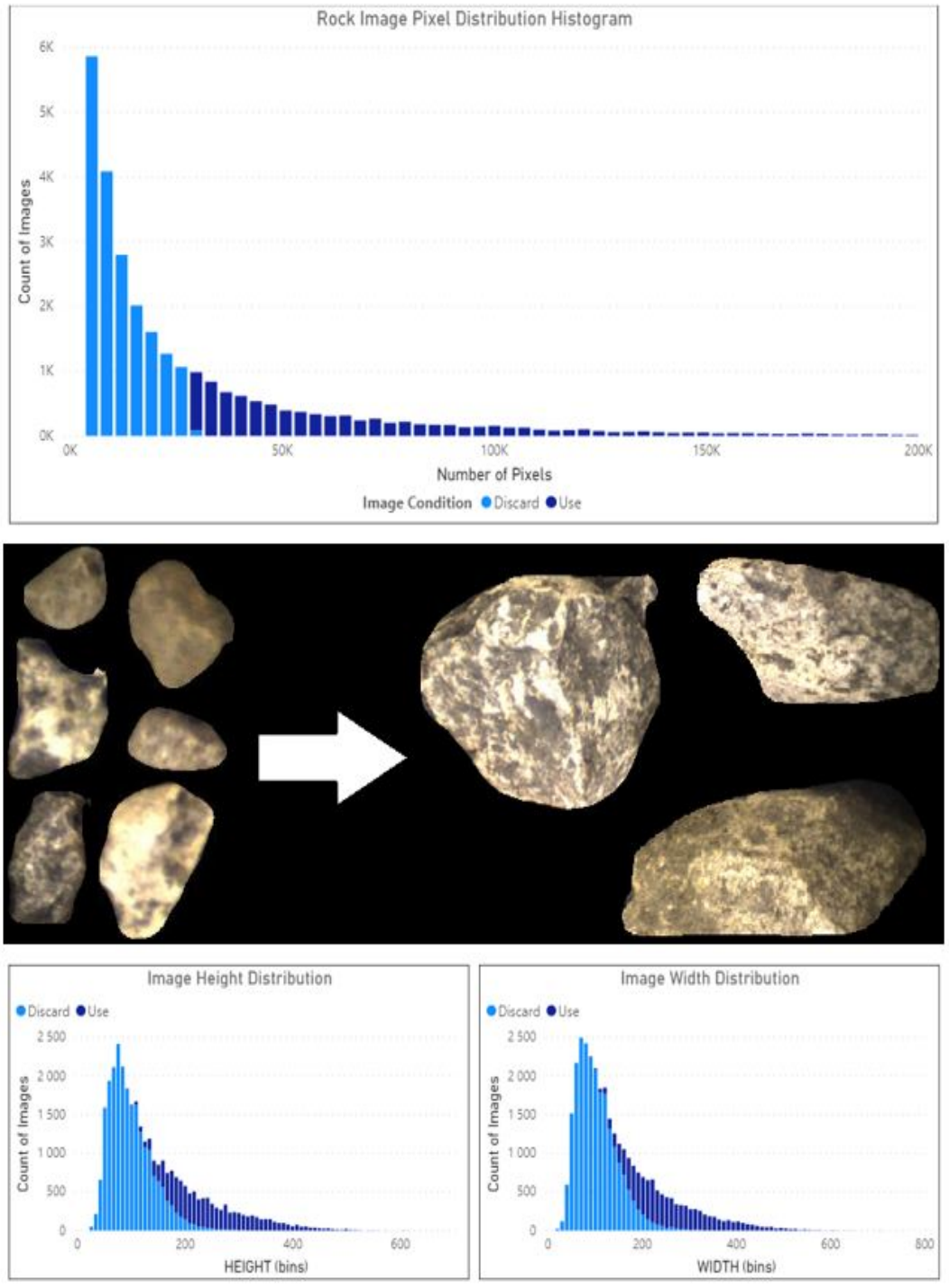


Figure 6.3: *Image distribution info-graphic. The top distribution shows frequency of total number of pixels per image, as well as which images were utilized for the next stage of modelling. The bottom distributions show the frequency of height and widths respectively, as well as which images were used for the next stage of processing. The rock images indicate visually the relative range from low pixel to high pixel resolution.*

### 6.3.2 Model Configuration and Training

As briefly discussed in the previous Method chapter, the clustering algorithm implemented for this phase of modelling is an unsupervised learning model, known as Deep Embedded Clustering with Data Augmentation (DEC-DA). Guo et al. (2018) developed this method as an advancement of previous deep embedded clustering methods, with the major improvement being the inclusion of data augmentation.

This unsupervised method consists of two clearly defined steps; a pre-training of the auto-encoder to generate an initial feature space, followed by further optimization of the space by constraining the feature space with a clustering loss. As with many supervised learning algorithms, there are several hyper-parameters and general architecture decisions that need to be defined for the model, the challenge in this case is that unsupervised models do not have a "labelled" target, and therefore it is difficult to define the relative performance of different model configurations.

Most of these hyper-parameter and architecture configuration decisions are thus based on literature, which demonstrated the model performance using labelled image data sets such as MNIST (LeCun 1998), for which clustering performance could be compared to actual labels. Guo et al. (2018) provided several examples of model performance, along with model configurations.

As a first step, the general architecture of the model is defined. The convolutional auto-encoder was selected since the application utilizes image data. The IDEC Guo et al. (2018) clustering model was selected, which represents the improved version of the initial clustering model initially developed by Guo et al. (2017) and defined as DEC. Finally, data augmentation was included to assist with generalization and clustering performance. This model architecture is defined as

ConvIDEC-DA (Guo et al. 2018).

Table 6.4 provides a summary of the model configuration for the clustering phase of the implementation.

Table 6.4: Model configuration and architecture for clustering phase.

| ConvIDEC-DA Model | |
|---|---|
| Architecture | Description |
| Type | Autoencoder |
| Model Details | ConvIDEC-DA |
| Clusters | 4 |
| Optimizer | SGD |
| Learning Rate | 0.01 |
| Momentum | 0.9 |
| Activation Functions | ReLU |
| Pre-training Epochs | 200 |
| Training Stopping Criteria | 0.001 |
| Clustering Method | K-means |

With the configuration of the model defined, it is then important to discuss and define the metrics on which the model performance can be evaluated.

### 6.3.3 Evaluation Metrics

The clustering phase of the project implementation is defined as an unsupervised learning model, which is built using the individual rocks data set - which is unlabelled. This scenario makes it difficult to define clear metrics with which to evaluate the performance of the model, since the outcomes cannot be compared to any known or labelled "target" values.

In order to provide some indication of performance, a set of evaluation metrics, used to determine relative clustering performance and optimal number of clusters, is utilized.

Ideally, future research should incorporate the manual labelling of the individual rocks to provide a labelled data set on which clustering performance can be fully evaluated. According to feedback from the resident expert geologist, the difficulty in identifying individual rocks for this specific application would have made it impractical - "Unfortunately, the rock pictures are really difficult to determine kimberlite type – to be honest. So you are not alone in that respect. A combination of the light reflecting off the paler parts of the rocks – mainly calcite linings by the look of things – and the mucky nature of the material makes it difficult to discern even the basic texture of the kimberlites." - John Ward (Resident Geologist, Kao Mine).

### Elbow Method

The Elbow Method is a heuristic used to determine the optimal number of clusters. The method defines the variance explained as a function of the number of clusters, the result of which is a line graph which should show progressively more variance explained as the number of clusters increases (Syakur et al. 2018). There should generally be an "elbow" where the rate of variance explained becomes less as more clusters are added, which can be considered as the optimal number of clusters. This elbow is not always clearly defined, thereby making this method subjective in many situations (Syakur et al. 2018).

### Silhouette Method

This method is also used to determine the optimal number of clusters. It does so by computing 'Silhouette coefficients' for each data point. These coefficients

compare relative similarity of each data point to others within the same clusters, as well as across clusters (Rousseeuw 1987). The results of this are typically a set of graphical visualizations, which provide insight into the optimal number of clusters (Rousseeuw 1987).

Figure 6.4 provides a summary of the elements required for the Silhouette coefficient calculation. The coefficient can be defined as (Rousseeuw 1987):

$$S(i) = \frac{c(i) - a(i)}{max\{a(i), c(i)\}} \tag{6.5}$$

Where $S(i)$ is defined as the dissimilarity coefficient for point $i$ in cluster $A$, and $a(i)$ is the average dissimilarity of $i$ relative to all other points within the same cluster $A$. The nearest neighbour cluster or object is calculated as $C$ in this example, which then allows us to calculate the average dissimilarity of $i$ relative to this cluster, defined as $c(i)$. The nearest neighbour cluster is used since this is the next best possibility for assigning the point to another cluster.

Finally, the average of $S(i)$ is calculated for each cluster, and used as an evaluation metric for the overall performance of the clustering model. Values closer to one indicate a good fit, with low intra cluster dissimilarity, and high inter cluster dissimilarity, whereas low coefficients closer to zero tend to indicate high intra cluster dissimilarity and no substantial clustering structure (Rousseeuw 1987).

In the next chapter, the results of both phases of modelling are discussed and evaluated based on the metrics defined within this chapter.

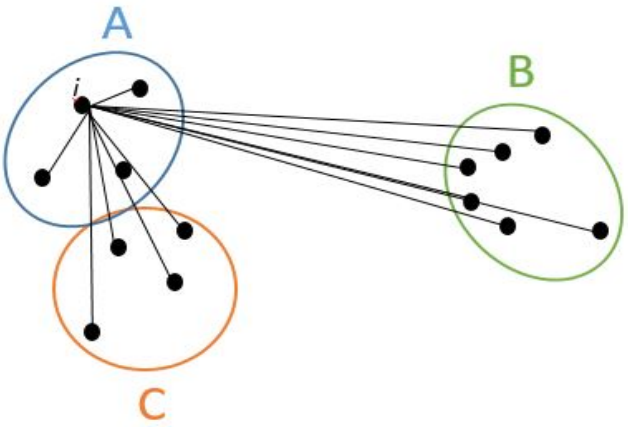


Figure 6.4: *Sketch showing elements required in Silhouette Coefficient calculation.*

# Chapter 7

# Results

This chapter provides a detailed overview of the results achieved for Phase 1, individual rock segmentation, and Phase 2, rock clustering methods, which together provide the complete solution pipeline. The methods used for each phase are evaluated based on the model performance metrics as defined in Chapter 6.

The results of the complete solution are also assessed within the context of the specific application for potential implementation on a mining plant.

## 7.1 Phase 1 Results

The Phase 1 instance segmentation method utilized the Matterport implementation of Mask R-CNN (Abdulla 2017). This method was selected as it provides a state of the art solution for several custom image segmentation applications, including a similar rock segmentation project using drone images to assess size distribution in rock stockpiles (Schenk et al. 2019).

Several iterations of the Mask R-CNN model were developed using different hyperparameters, backbone architecture and transfer learning methods. After analyzing the results for each of these, the best performing model was selected. This

section provides an overview of the results for this iterative development process.

The model that performed the best was that which effectively identified and segmented individual rocks in each image. The good performance of Phase 1 was important to ensure good quality individual rock images were available for the next stage of modelling. The best performing model achieved a log-loss of 0.737 on the test set, which represents the sum of the bounding box, class label and mask loss (Girshick 2015). Table 7.1 provides a summary of the model performance; indicating a bounding box loss of 0.123, a class label loss of 0.254 and a mask loss of 0.172. The relatively low loss values for each of these categories suggests the model is effective for this application.

The rock instance model achieved a mAP of 0.701 at an $IOU$ of 0.5, with anything over 0.6 considered to be a good performing model (He et al. 2017). A benchmark for performance is considered a mAP of 0.623 as scored by the Mask R-CNN model with a ResNeXt-101-FPN backbone (He et al. 2017).

Table 7.1: Performance results for Phase 1 Instance Segmentation Model.

| Mask R-CNN Instance Segmentation Model | |
|---|---|
| Metric | Results |
| Bounding Box Loss | 0.123 |
| Class Label Loss | 0.254 |
| Mask Loss | 0.172 |
| mAP @ IOU=0.5 | 0.701 |

While the results described in Table 7.1 can be considered to be the best performing model, multiple experiments were run using a combination of different hyper-parameters, architectures and total number of training and test images.

Table 7.2 summarizes the key experimental setups tested for the Mask R-CNN

Phase 1 modelling.

Table 7.2: Mask R-CNN Experiments

| Mask R-CNN Experiments | | | |
|---|---|---|---|
| Run Number | Backbone | Transfer Learning | Dataset |
| 1 | ResNet 101 | COCO | No Data Augmentation, 20 training, 5 test images. |
| 2 | ResNet 101 | COCO | Custom Data Augmentation, 40 training, 5 test images. |
| 3 | ResNet 50 | COCO | Full Data Augmentation, 100 training, 20 test images. |
| 4 | ResNet 101 | ImageNet | Full Data Augmentation, 100 training, 20 test images. |
| 5 | ResNet 50 | COCO | Full Data Augmentation, 100 training, 20 test images. |

The training and validation log-loss results of each experimental run defined in Table 7.2 can be visualized in Figure 7.1.

One of the most important variables to understand from a practicality point of view was the total number of training images required to sufficiently train the model. Due to the time-intensive nature of image labelling, these results may play a considerable part in determining the overall feasibility of the solution on mining plants. Experiment runs 1 and 2 used 20 and 40 labelled training images respectively, whereas runs 3, 4 and 5 each used 100 training images. From the validation log-loss results described in Figure 7.1, it is evident that increasing the number of training images has a significant impact on the overall performance of the model. The validation set is automatically generated using 10% of the training

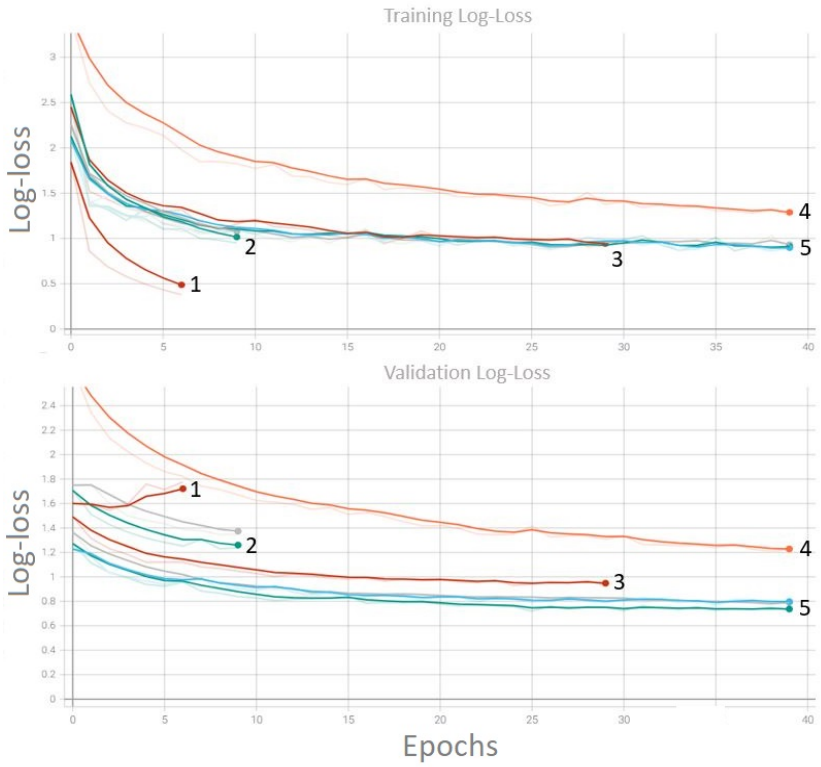


Figure 7.1: *Log-loss results for training (top) and validation (bottom) data sets.*

data during the Mask R-CNN model training and testing process (Abdulla 2017). Since it takes on average 5-10 minutes to label each image, a trade-off should be conducted to determine the relative performance gain versus the incremental labour cost of increasing the training data set.

The stopping criteria for the model was defined using max-gradient, and set at an $\epsilon$ of 0,001 based on the Log-loss result. This value is commonly used across various applications of Mask-RCNN model implementation, and provides a good baseline in terms of trading off model complexity versus accuracy (Abdulla 2017).

In order to reduce the required number of training images, a robust data augmentation pipeline is required. Experiment 1 shows a clear example of the model over-fitting; indicated by the steep improvement in training log-loss compared to the poor validation log-loss results. This is as a result of a lack of data augmen-

tation, coupled with a small total number of training images.

The process of transfer learning was used to train the vast majority of the network on a generic data set, followed by training the head of the network on the custom data set, which also proved to be an important metric to explore. The COCO (Lin et al. 2014) data set was found to perform significantly better than the ImagNet equivalent (Deng et al. 2009), used in Run 4.

The model backbone architecture also played a significant role in the overall performance of the model. The larger and more complex ResNet 101 (Abdulla 2017) architecture performed better than the smaller ResNet 50 option (Abdulla 2017), used for Run 3. This result is supported by several other studies, including the rock size analysis of stock piles using drone images (Schenk et al. 2019).

A summary of results based on log-loss and mAP is shown in Table 7.3. These results support the initial findings both in terms of log-loss and mAP at $IOU = 0.5$.

Table 7.3: Mask R-CNN Experiment Results Summary

| Mask R-CNN Experiment Results Summary | | |
|---|---|---|
| Run Number | Log-loss | mAP @ IOU = 0.5 |
| 1 | 1.721 | 0.355 |
| 2 | 1.260 | 0.627 |
| 3 | 0.948 | 0.651 |
| 4 | 1.229 | 0.584 |
| 5 | 0.737 | 0.701 |

Figure 7.2 provides a nice example of the relative performance of three different models used for rock instance segmentation on a validation image. The mAP scores for each model correlate well to the visual results observed.

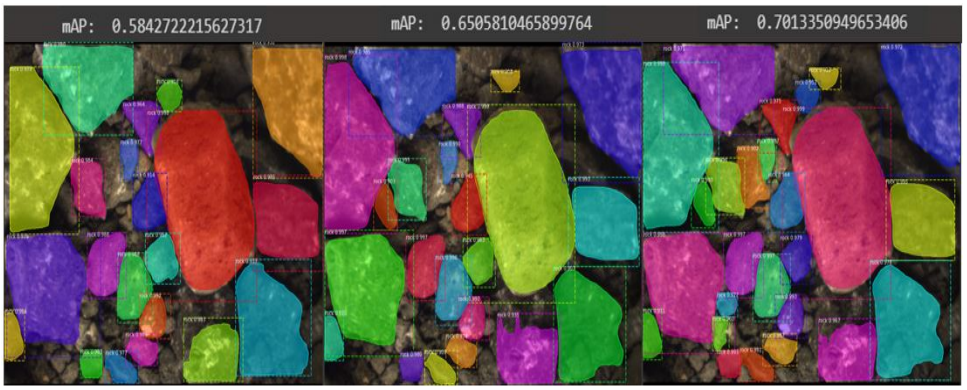


Figure 7.2: *Visual comparison of masking results for Experiment Run 3 (middle), 4 (left) and 5 (right) on a validation image.*

With the results for the Phase 1 instance segmentation modelling complete, it is then important to assess the Phase 2 clustering results in the next section. The overall performance of the solution can then be assessed based on this.

## 7.2 Phase 2 Results

The second phase of modelling consisted of the unsupervised method Deep Embedded Clustering (DEC) (Xie et al. 2016). Several variations of this method exist, most of which were explored in this research. The primary focus was on the implementation of the method most suited for image data, and considered state of the art in terms of DEC currently - Convolutional Deep Embedded Clustering with Data Augmentation (ConvDEC-DA) (Guo et al. 2018).

Effectively quantifying the performance of unsupervised methods is challenging, considering there are no labelled training data to provide actual targets against which to measure the model performance. Considering the method makes use of a k-means clustering algorithm component, it was possible to use methods such as the Elbow (Syakur et al. 2018) and the Silhoutte (Rousseeuw 1987) method, discussed in Chapter 6, to determine the optimal number of clusters. Apart from these indicators, the analysis of the model performance would be fairly subjective

and involve visually assessing examples of the rock images assigned to each cluster to determine if the clustering did indeed group rocks into facie types.

Unfortunately the clustering phase of the pipeline did not produce the expected results, ideally the output should have generated a set of well defined clusters, each associated with a different rock facie. In reality, a single elongated cluster was generated; this was the case for several variations of clustering algorithms investigated.

The results can be visualized in Figure 7.3, showing a principal component analysis (PCA) biplot of the resulting clustering, represented in a two-dimensional space. These PCA plots are used extensively for data exploration and dimensionality reduction, usually as a first step before implementing supervised learning models (Wold et al. 1987). For this specific application each of the 10 original dimensions generated in the model are mapped to the PCA biplot. The 10 dimensions or features have been generated through the training of the auto-encoder and k-means clustering part of the ConvDEC-DA model. This output data set from the ConvDEC-DA model should ideally be optimised by this type model to provide optimal clustering features. The percentage of total variance explained by the two dimensional plot is significant, with the horizontal and vertical axes representing 82% and 16% respectively.

The ConvDEC-DA model provides an output mapping of each image to a reduced 10 dimensional space. This output, as represented in Figure 7.3, shows no significant clustering. These findings are supported by analysis of the outputs from the Silhouette method, discussed in more detail in Chapter 6. The results are shown in Figure 7.4.

The results indicate that using two clusters was the optimal solution, producing an average silhouette width of just less than 0.60. In general, silhouette width

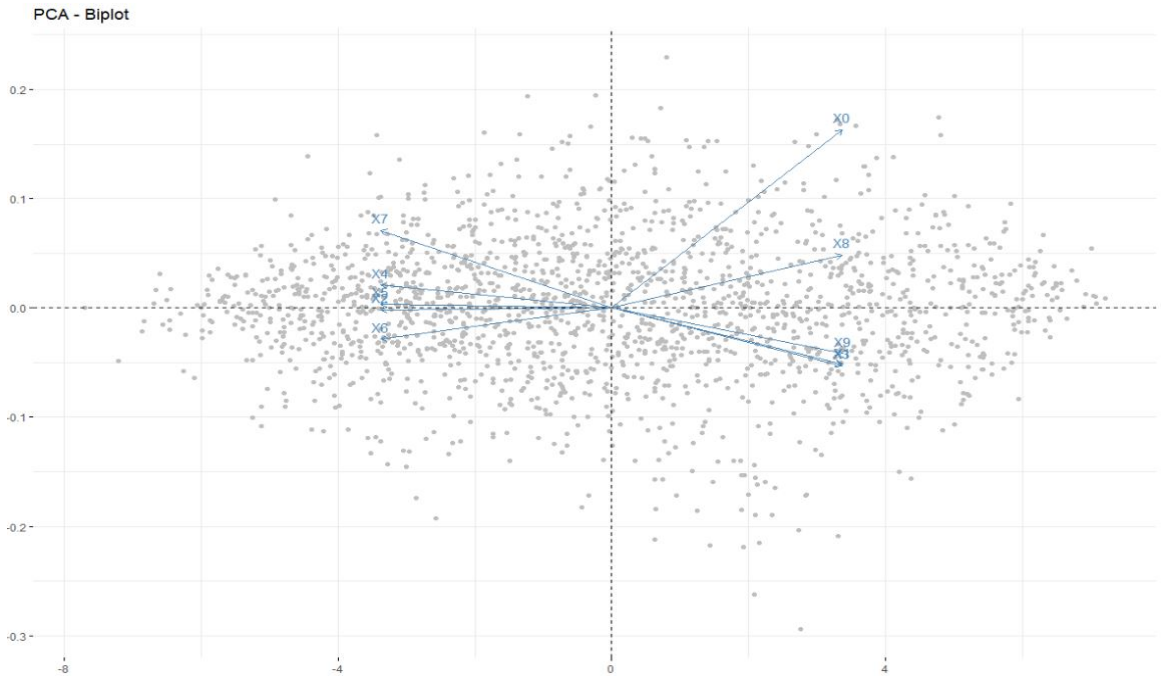


Figure 7.3: *Principal component analysis (PCA) biplot showing clustering results output based on the ConvDEC-DA modelling.*

values ranging between 0.51 - 0.70 can be considered to have reasonable structure within the clustering (Rousseeuw 1987). This indicates that, although there is no visible clustering when represented within a two dimensional PCA Biplot, there does exist some degree of structure to the clustering.

In addition to the Silhouette method, the Elbow method was utilized to assess the optimal number of clusters. The results of which can be visualized in Figure 7.5. The Elbow method also seems to indicate that two clusters provide the optimal solution for this data set.

With these results in mind, it was also important to explore the clustering results in terms of actual rock characteristics.

By investigating sample points from the clustering results and PCA Biplot visual, it was possible to further explore the context of these results. Figure 7.6 shows a few examples of specific data points selected and their associated input images.

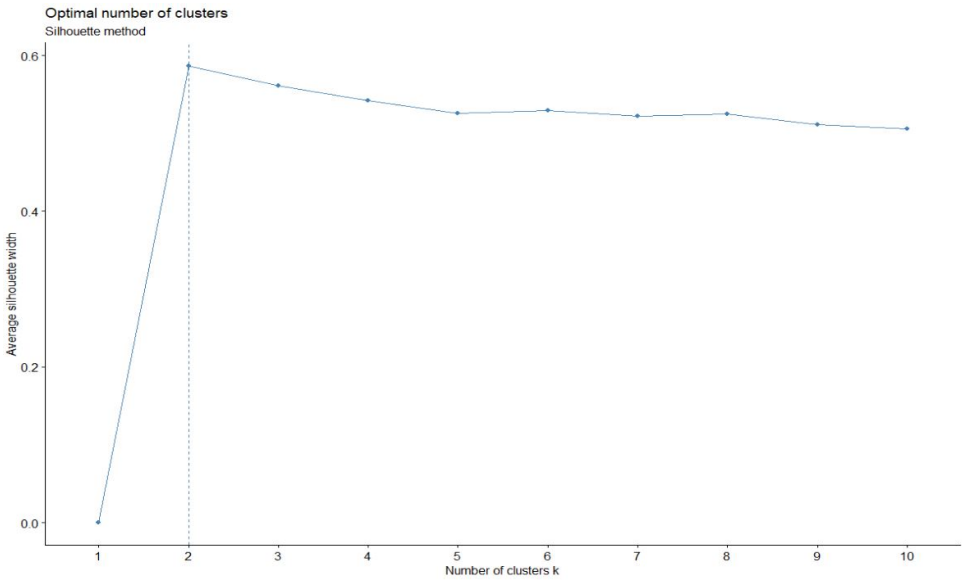


Figure 7.4: *Silhouette method results for ConvDEC-DA clustering model.*

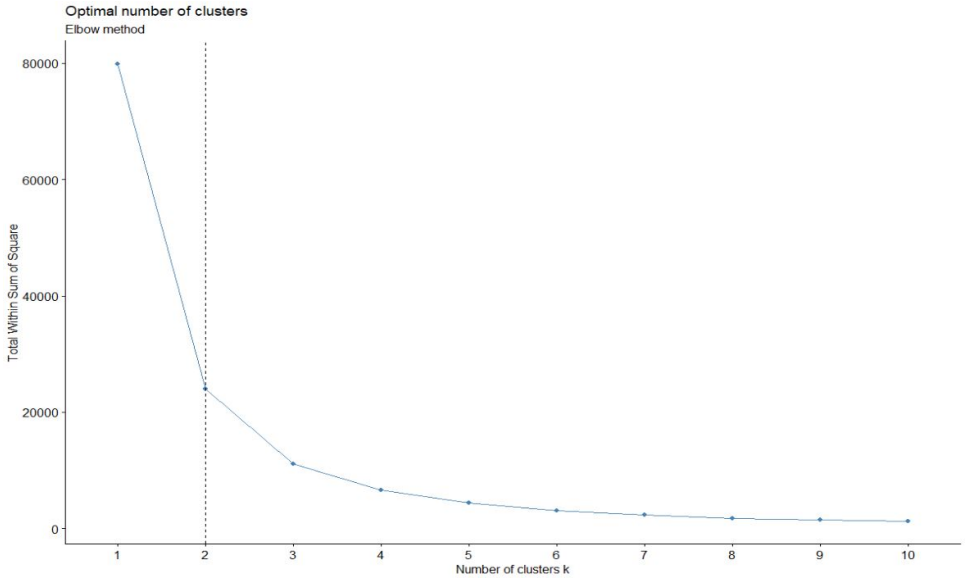


Figure 7.5: *Elbow method results for ConvDEC-DA clustering model.*

Although well defined clusters are not evident, there does appear to be some degree of grouping and ordering of images according to their respective colour and texture properties. The colour and texture of the samples does seem to range across a spectrum from facie FK through the GQ and LQ, which is very similar to Basalt.

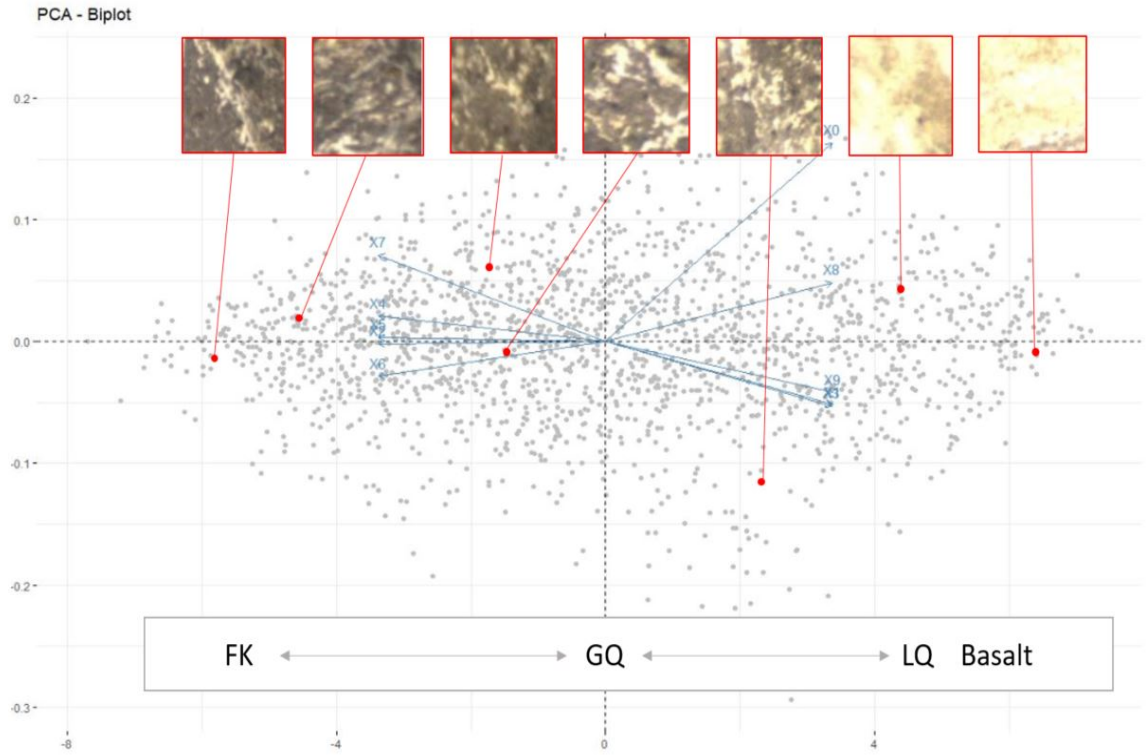


Figure 7.6: *Visualization of rock clustering results based on a sample of rock images for ConvDEC-DA clustering model.*

The results of this sampling analysis provides insight into one of the potential reasons for a lack of clear grouping or cluster formations. While certain individual rock samples can visibly be categorized into a specific facie, the majority of samples are not clearly defined. Several images appear to have colour characteristics and textures which could fall on the spectrum between two facie types, and as such, the clustering model will not be able to clearly separate individual clustering for a large random sample of rock images.

Another limitation of the clustering effectiveness could be the relatively low reso-

lution of images when analyzing on individual rock level. This might be improved by installing a camera with a higher resolution camera instead. An important finding in other similar research, including that conducted by Iyakwari et al. (2017), found that Near Infrared (NIR) range cameras were far more effective at identifying different mineral composites when compared to standard spectrum colour cameras.

While the overall solution, consisting of the instance segmentation and clustering phases, provided some important insights and was able to successfully group certain rock images to a certain degree, there is still work to be done to further refine the solution to make it suitable for implementation on a mining plant. The clustering phase especially requires further attention, which will be discussed further in the recommendations section in Chapter 9.

# Chapter 8

# Conclusion

This chapter aims to provide an overview of the complete implementation of this research, as well as to summarize the key findings and results.

This research presents an approach which utilizes computer vision to assess the feed material entering a mining plant, by clustering individual rocks into their respective facie types, thereby providing the mining operation with a near live and automated approach to monitoring feed input by facie type.

The method involved a two phased approach or model pipeline. The first of which used images of the feed conveyor belt material to which an instance segmentation model was applied to extract individual rock images with no background. The instance segmentation model consisted of a Mask R-CNN solution, making use of transfer learning on the COCO data set, as well as a ResNet 101 type architecture. The model performed well on the data set, with a mAP of 0.701 at an $IOU$ threshold of 0.5 and a log-loss of 0.737.

The second phase of modelling leveraged an unsupervised clustering method known as Convolutional Deep Embedded Clustering with Data Augmentation (ConvDEC-DA). This involved using the previously segmented individual rock

images to train a model with an auto-encoder type structure while also optimizing a k-means clustering algorithm loss function. The result of which was a set of features per image, optimized both according to the auto-encoder and clustering loss function (Guo et al. 2018).

Analysis of the second phase results was limited due to it being an unsupervised problem with no labelled data for reference. The Silhouette method was applied and suggested the optimal number of clusters to be two, with an average silhouette width just below 0.6, suggesting a degree of structure in the clustering results (Rousseeuw 1987). This result was further confirmed using the Elbow method, which also suggested an optimum number of clusters of two, based on the plotted total within cluster of squares per number of potential clusters.

On analysis of the results it was found that the clustering phase of the model did not produce well defined clusters, which can be attributed to a number of factors, although primarily relating to the quality of the available data. The image resolution of the individual rocks was found to be relatively low, this coupled with the dirty feed material into the plant often obscured detail related to rock colour and texture, resulting in a reduced performance model. The rock types were also often found to be indistinguishable in terms of facie origin to the naked eye, which makes applying a computer vision model challenging.

# Chapter 9

# Recommendations and Further Studies

During the implementation of this research and analysis of results, several areas for future improvement were identified. Starting with the image data; since all pictures of material were taken above the main plant feed conveyor, many rocks were to shown to be dirty, with large amounts of sand and mud in each image. The size variance between rocks was also seen to be significant. All of these factors contributed to increased complexity and dilution of the core colour and texture features required to successfully cluster specific rock types.

Future research should consider installing the camera above a conveyor with material having gone through the first stage of crushing and wet screening. This material would then be washed and contain more uniform rocks.

Other research on production level ore sorters have been known to effectively use NIR camera technology, which provides superior features compared to a standard Reg-Green-Blue (RGB) type camera. Should a budget be available, future research should include the use of NIR colour spectrum, in addition to RGB,

providing a better set of features on which to train the model.

While the first phase of the model pipeline provided good results, the time-intensive process of manually labelling individual rocks on each training image is a significant drawback of implementing this solution as a potential commercial application in mining plants. The first phase was only required since the feed material was often found to have multiple facies per image - assuming material is less mixed, one could consider a purely unsupervised clustering approach without the need to segment individual rocks for processing.

The instance segmentation model type was a Mask R-CNN, with a ResNet 101 backbone architecture. This model is complex, and its size might make it prohibitive to implement in real word solutions, including the deployment of models to edge devices which require light weight and fast architectures. Several alternative and lighter architectures do exist, and further research should weigh up the trade-off between complexity and performance to select the best solution.

The lack of defined clusters in the second phase of modelling could be attributed to the low quality of training data, as well as the general rock characteristics and lack of clearly differentiating features between facie. Future research could explore similar applications on mines processing other rock types, which may provide more well defined clusters.

Ideally a set of labelled rock images should be used to compare clustering results to the pre-defined labels to validate the accuracy and effectiveness of the clustering solution. Although impractical for large scale adoption in industry, this step is important for validation of the concept.

The use of Google Colab as an open source computing platform was invaluable during this research. Access to GPU processing capacity is critical to efficiently train deep learning models, and it was an important aspect of this research.

# Bibliography


Abbas, A., Khan, S. U. & Zomaya, A. Y. (2020), *Fog Computing: Theory and Practice*, John Wiley & Sons.

Abdulla, W. (2017), 'Mask r-cnn for object detection and instance segmentation on keras and tensorflow', `https://github.com/matterport/Mask_RCNN`, last accessed on 08/03/2021.

Aggarwal, C. C. (2018), 'Neural networks and deep learning', *Springer* **10**, 978–3.

Cas, R. A. F., Porritt, L., Pittari, A. & Hayman, P. C. (2009), 'A practical guide to terminology for kimberlite facies: A systematic progression from descriptive to genetic, including a pocket guide', *Lithos* **112**, 183–190.

Chang, Y.-C. (2018), 'The 5 main steps in processing diamond ore'.
**URL:** *http://multotec.ca/en/blog/the-5-main-steps-in-processing-diamond-ore/1/15, last accessed on 12/06/2020*

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. & Fei-Fei, L. (2009), Imagenet: A large-scale hierarchical image database, *in* '2009 IEEE conference on computer vision and pattern recognition', Ieee, pp. 248–255.

Dumoulin, V. & Visin, F. (2016), 'A guide to convolution arithmetic for deep learning', *arXiv preprint arXiv:1603.07285* .

Gabor, D. (1946), 'Theory of communication. part 1: The analysis of information', *Journal of the Institution of Electrical Engineers - Part III: Radio and Communication Engineering* **93**, 429–441(12).
**URL:** *https://digital-library.theiet.org/content/journals/10.1049/ji-3-2.1946.0074, last accessed on 14/05/2020*

Galdames, F. J., Perez, C. A., Estévez, P. A. & Adams, M. (2017), 'Classification of rock lithology by laser range 3d and color images', *International Journal of Mineral Processing* **160**, 47–57.

Galdames, F. J., Perez, C. A., Estévez, P. A. & Adams, M. (2019), 'Rock lithological classification by hyperspectral, range 3d and color images', *Chemometrics and Intelligent Laboratory Systems* **189**, 138–148.

Girshick, R. (2015), Fast r-cnn, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 1440–1448.

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep Learning*, MIT Press. `http://www.deeplearningbook.org`, last accessed 19/08/2020.

Guo, X., Liu, X., Zhu, E. & Yin, J. (2017), Deep clustering with convolutional autoencoders, *in* 'International conference on neural information processing', Springer, pp. 373–382.

Guo, X., Zhu, E., Liu, X. & Yin, J. (2018), Deep embedded clustering with data augmentation, *in* 'Asian conference on machine learning', pp. 550–565.

He, K., Gkioxari, G., Dollár, P. & Girshick, R. (2017), Mask r-cnn, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 2961–2969.

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 770–778.

Iyakwari, S., Glass, H. J. & Obrike, S. E. (2017), 'Discerning mineral association in the near infrared region for ore sorting', *International Journal of Mineral Processing* **166**, 24–28.

Janocha, K. & Czarnecki, W. M. (2017), 'On loss functions for deep neural networks in classification', *arXiv preprint arXiv:1702.05659* .

Kjarsgaard, B. (2007), 'Kimberlite pipe models: Significance for exploration', *Proceedings of Exploration 07: Fifth Decennial International Conference on Mineral Exploration* .

Kong, C. & Lucey, S. (2017), 'Take it in your stride: Do we need striding in cnns?', *arXiv preprint arXiv:1712.02502* .

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012*a*), Imagenet classification with deep convolutional neural networks, *in* F. Pereira, C. J. C. Burges, L. Bottou & K. Q. Weinberger, eds, 'Advances in Neural Information Processing Systems 25', Curran Associates, Inc., pp. 1097–1105.
**URL:** *http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf, last accessed on 24/05/2020*

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012*b*), 'Imagenet classification with deep convolutional neural networks', pp. 1097–1105.

LeCun, Y. (1998), 'The mnist database of handwritten digits', *http://yann. lecun. com/exdb/mnist/* .

LeCun, Y., Kavukcuoglu, K. & Farabet, C. (2010), Convolutional networks and applications in vision, *in* 'Proceedings of 2010 IEEE international symposium on circuits and systems', IEEE, pp. 253–256.

Lee, C.-Y., Gallagher, P. & Tu, Z. (2017), 'Generalizing pooling functions in cnns:

Mixed, gated, and tree', *IEEE transactions on pattern analysis and machine intelligence* **40**(4), 863–875.

Li, X., Grandvalet, Y., Davoine, F., Cheng, J., Cui, Y., Zhang, H., Belongie, S., Tsai, Y.-H. & Yang, M.-H. (2020), 'Transfer learning in computer vision tasks: Remember where you come from', *Image and Vision Computing* **93**, 103853.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. & Zitnick, C. L. (2014), Microsoft coco: Common objects in context, *in* 'European conference on computer vision', Springer, pp. 740–755.

Liu, X., Wang, H., Jing, H., Shao, A. & Wang, L. (2020), 'Research on intelligent identification of rock types based on faster r-cnn method', *IEEE Access* **8**, 21804–21812.

Meyer, F. (1994), 'Topographical distance and watershed lines', *Signal Processing* **38**(1), 113–125.

Mitchell, R. H. (2012), *Kimberlites, orangeites, and related rocks*, Springer Science & Business Media.

Nielsen, M. A. (2015), *Neural Networks and Deep Learning*, Determination Press.

Patel, A. K., Chatterjee, S. & Gorai, A. K. (2017), 'Development of machine vision-based ore classification model using support vector machine (svm) algorithm', *Arabian Journal of Geosciences* **10**(5), 107.

Perez, C. A., Saravia, J. A., Navarro, C. F., Schulz, D. A., Aravena, C. M. & Galdames, F. J. (2015*a*), 'Rock lithological classification using multi-scale gabor features from sub-images, and voting with rock contour information'.
**URL:** *http://www.sciencedirect.com/science/article/pii/S0301751615300326, last accessed on 02/05/2020*

Perez, C. A., Saravia, J. A., Navarro, C. F., Schulz, D. A., Aravena, C. M. & Galdames, F. J. (2015*b*), 'Rock lithological classification using multi-scale gabor features from sub-images, and voting with rock contour information', *International Journal of Mineral Processing* **144**, 56 – 64.
**URL:** *http://www.sciencedirect.com/science/article/pii/S0301751615300326, last accessed on 02/05/2020*

Ran, X., Xue, L., Zhang, Y., Liu, Z., Sang, X. & He, J. (2019), 'Rock classification from field image patches analyzed using a deep convolutional neural network', *Mathematics* **7**(8), 755.

Ravanbakhsh, S., Schneider, J. & Poczos, B. (2017), Equivariance through parameter-sharing, *in* 'International Conference on Machine Learning', PMLR, pp. 2892–2901.

Ren, Y., Hu, K., Dai, X., Pan, L., Hoi, S. C. & Xu, Z. (2019*a*), 'Semi-supervised deep embedded clustering', *Neurocomputing* **325**, 121–130.

Ren, Y., Hu, K., Dai, X., Pan, L., Hoi, S. C. & Xu, Z. (2019*b*), 'Semi-supervised deep embedded clustering', *Neurocomputing* **325**, 121–130.

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I. & Savarese, S. (2019), Generalized intersection over union: A metric and a loss for bounding box regression, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 658–666.

Rojas, R. (1996), The backpropagation algorithm, *in* 'Neural networks', Springer, pp. 149–182.

Rousseeuw, P. J. (1987), 'Silhouettes: a graphical aid to the interpretation and validation of cluster analysis', *Journal of computational and applied mathematics* **20**, 53–65.

Ruder, S. (2016), 'An overview of gradient descent optimization algorithms', *arXiv preprint arXiv:1609.04747* .

Saha, S. (2018), 'A comprehensive guide to convolutional neural networks — the eli5 way'.

**URL:** *https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, last accessed on 24/05/2020*

Salinas, R., Raff, U. & Farfan, C. (2005), 'Automated estimation of rock fragment distributions using computer vision and its application in mining', *IEE proceedings.* **152**(1).

Schenk, F., Tscharf, A., Mayer, G. & Fraundorfer, F. (2019), 'Automatic muck pile characterization from uav images.', *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences* **4**.

Simonyan, K. & Zisserman, A. (2014), 'Very deep convolutional networks for large-scale image recognition', *CoRR* **abs/1409.1556**.

Skinner, E. (2009), 'The geology of the kao kimberlites, lesotho', *Geology Department, Rhodes University, Grahamstown 6140, South Africa* .

Skinner, E. & Marsh, J. (2004), 'Distinct kimberlite pipe classes with contrasting eruption processes', *Lithos* **76**(1-4), 183–200.

Smith, B. S., Nowicki, T., Russell, J., Webb, K., Mitchell, R., Hetman, C., Harder, M., Skinner, E. & Robey, J. A. (2013), Kimberlite terminology and classification, *in* 'Proceedings of 10th International Kimberlite Conference', Springer, pp. 1–17.

Stanford (2020), 'Cs231n: Convolutional neural networks for visual recognition'.

**URL:** *http://http://cs231n.stanford.edu/, last accessed on 21/06/2020*

Syakur, M., Khotimah, B., Rochman, E. & Satoto, B. (2018), Integration k-means clustering method and elbow method for identification of the best customer profile cluster, *in* 'IOP Conference Series: Materials Science and Engineering', Vol. 336, IOP Publishing, p. 012017.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015), 'Going deeper with convolutions', pp. 1–9.

Trivedi, S. & Kondor, R. (2017), 'Lecture 7, convolutional neural networks, cmsc 35246: Deep learning'.
**URL:** *https://ttic.uchicago.edu/ shubhendu/Pages/Files/Lecture7*$_p$*auses.pdf*

Van der Maaten, L. & Hinton, G. (2008), 'Visualizing data using t-sne.', *Journal of machine learning research* **9**(11).

Van Rossum, G. & Drake Jr, F. L. (1995), *Python tutorial*, Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. & Manzagol, P.-A. (2010), 'Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion', *Journal of machine learning research* **11**(Dec), 3371–3408.

Wada, K. (2016), 'labelme: Image Polygonal Annotation with Python', `https://github.com/wkentaro/labelme`, last accessed on 02/11/2020.

Wang, Y., Li, Y., Song, Y. & Rong, X. (2020), 'The influence of the activation function in a convolution neural network model of facial expression recognition', *Applied Sciences* **10**(5), 1897.

Wold, S., Esbensen, K. & Geladi, P. (1987), 'Principal component analysis', *Chemometrics and intelligent laboratory systems* **2**(1-3), 37–52.

Xie, J., Girshick, R. & Farhadi, A. (2016), Unsupervised deep embedding for clustering analysis, *in* 'International conference on machine learning', pp. 478–487.

Y. Zhao, D. L. & Li, Z. (2007), 'Performance enhancement and analysis of an adaptive median filter', *2007 Second International Conference on Communications and Networking in China, Shanghai, 2007* pp. 651–653.

Yang, B., Fu, X., Sidiropoulos, N. D. & Hong, M. (2017), Towards k-means-friendly spaces: Simultaneous deep learning and clustering, *in* 'Proceedings of the 34th International Conference on Machine Learning-Volume 70', JMLR. org, pp. 3861–3870.

Yani, M. et al. (2019), Application of transfer learning using convolutional neural network method for early detection of terry's nail, *in* 'Journal of Physics: Conference Series', Vol. 1201, IOP Publishing, p. 012052.

Zhu, M. (2004), 'Recall, precision and average precision', *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo* **2**(30), 6.