White Maize Classification and Grading

Shane Erasmus
23549777

Report submitted in partial fulfilment of the requirements of the module Project (E) 448 for the degree Baccalaureus in Engineering in the Department of Electrical and Electronic Engineering at Stellenbosch University.

Supervisor: Rensu P. Theart
Department of Electrical and Electronic Engineering

October 2023

Acknowledgements

I would first like to thank Johan Olivier, Arlo Steyn, and Werner at Timber King for their help in building and testing the system. I would also like to thank Wiana and Ben from SAGL for providing knowledge and insights about the agriculture industry and for supplying maize samples that were key to the development of the system. Finally, Rensu Theart for his supervision and assistance.

Stellenbosch
UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

Plagiaatverklaring / *Plagiarism Declaration*

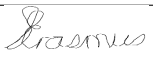
1. Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
   Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.

2. Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
   I agree that plagiarism is a punishable offence because it constitutes theft.

3. Ek verstaan ook dat direkte vertalings plagiaat is.
   I also understand that direct translations are plagiarism.

4. Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelikse aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
   Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism

5. Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
   I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

23549777 Studentenommer / *Student number*	 Handtekening / *Signature*
S Erasmus Voorletters en van / *Initials and surname*	27/10/2023 Datum / *Date*

Abstract

This report presents a innovative approach to maize sample grading. This is done by building a image capturing system coupled with image processing technologies and deep learning algorithms. The core portion of the report revolves around overcoming many of the limitations and difficulties in developing such a system and optimizing its performance. The developed system is finally tested on unseen batches of maize and compared with other technologies. The results gathered offer valuable insights and promise it's potential for developing further solutions that could, some day, be utilized in the industry.

Contents

List of Figures

List of Tables

Nomenclature

Variables and functions

Ω	Ohms
A	Amperes
DIS	Discolored kernels
DoG	Difference in Gaussian blur
EUR	Euros
HD	Heat Damage
HI	Hyperspectral Imaging
ID	Insect Damage
LED	Light Emitting Diodes
MI	Multispectral Imaging
MSE	Mean Squared Error
nm	Nanometers
PCA	Principal Component Analysis
PLS-DA	Partial Least Square Discriminant Analysis
ReLU	Rectified Linear Unit
RMSProp	Root Mean Squared Propagation
SBC	Single-board Computer
V	Volts
W	Watts
WV	Wavebands

Acronyms and abbreviations

CNN	Convolutional Neural Network
HDMI	High-Definition Multimedia Interface
NoIR	No Infrared Filter
RSA	Republic of South Africa
SAGL	South African Grain Laboratory
USB	Universal Serial Bus
WM	White Maize
YM	Yellow Maize

Chapter 1

Introduction

1.1. Background

Maize is a staple food all over the world and accounts for a significant share of South Africa's agricultural exports [3]. It is used to produce syrups, alcohols and breakfast cereals. It remains the most important feed for livestock in the country and for many of the people forms part of their staple diet. Attaining the highest quality in the industry is essential, as it ensures the most favourable market value for grain. Additionally knowing the quality of the maize is also beneficial for farmers, since this tells them how to properly store and preserve the batch.

The typical grading system to determine defective kernels is as follows. According to the South African department of agriculture [1], a 150g sample, or about 380 kernels is taken from the maize consignment and this sample is examined by hand by a qualified inspector. The inspector will sum the portions of each type of defect and then assign an overall grade to the batch. The traditional methods are slow and prone to inaccuracies. The project goal is thus to test alternative ways in which to grade defective maize kernels.

1.2. Problem Statement

This project aims to address the problem of having potential inconsistencies in maize grading by building a camera system that captures images of both sides of the grain and then, by using a trained model, automatically grades the batches with a high degree of accuracy. With computer-vision one would essentially like to remove the risk for induced human error.

1.3. Objectives

In order to solve the above mentioned problem the system should be able to perform the following objectives:

1. Having a camera system that will be able to take a top and bottom images of a 150g samples with sufficient quality to distinguish defects.
2. Perform image processing to properly separate each of the kernels.
3. Use the labeled kernels to train a machine learning model.
4. Test the system with the labelled test set to determine the robustness of the system.
5. Optimize the system so that the process is relatively fast. So that multiple batches can be analyzed quickly.
6. Test and compare results with other methods of grading technologies.
7. Use graded batches that are independently sourced to test the model on.

1.4. Project Scope

Due to limited time and resources, the project has some degree of constraints. These constraints are as follows:

- Maize kernels have many defects and unfortunately not all of the defects are accounted for. This project examines a portion of defects which are relatively difficult to distinguish by mere sight.
- Foreign material found in batches will not be evaluated, but rather removed from the batches. Likewise small kernels and shavings below a certain size will also be discarded.
- The top and bottom side of the kernel will be examined. And thus not the entire shape of the kernel since the edges wont be taken into account.
- Chemical properties and interiors of kernel not be examined. Moisture will also not be accounted for.
- Due to the scarcity of some of the defects the training set will be limited in size.
- In this project we are not aiming to develop a fully automated system. For this reason we use only one camera and the design considerations of the camera system is not optimized for portability.

- Due to the limited time of the project other types of cameras such as NoIR, hyperspectral etc. will not be tested.

1.5. Structure of Report

- **Chapter 1 Introduction**
  Overview of the topic and problem at hand, as well as project objectives and a brief explanation of the solution design.
- **Chapter 2 Literature review**
  Outlining fundamental concepts of the project with an explanation of other existing solutions to problem. Additionally some software knowledge required in developing a solution to the problem.
- **Chapter 3 Hardware design**
  How the system hardware is designed and why certain choices have been chosen above others in development.
- **Chapter 4 Software and System Design**
  How the software operates and the specific details of each process that runs.
- **Chapter 5 Results**
  Results in performance of algorithm, run time, training sets and testing batches.
- **Chapter 6 Conclusion**
  Conclusion and overview of the system.

Chapter 2

Literature review

Chapter presents a literature review, outlining the fundamental concepts of the project. It begins by giving an overview of how maize is normally classified in the industry, including the types of defects that will be dealt with. The chapter then dives into previous technologies and studies related to this classification problem. Finally some key software background concepts are briefly explained.

2.1. Grading of maize batches

According to the South African Department of Agriculture, white maize is categorized into 3 classes: WM1, WM2, WM3, with WM1 being the most pure and WM3 the least [1]. To grade the maize, a percentage of deviations, discoloring, and foreign materials in a sample is assessed according to the RSA grading regulations. The following table is a snippet from the Department of Agriculture report which shows the percentages used to place the sample of maize into each of the above mentioned categories. The defect category which results in the lowest grade will be taken as the overall grade for the batch. When proportions exceed the WM3 limit the batch gets labelled as "OTHER" meaning no grade can be given.

Table 2.1: Regulations relating to the Grading, Packing and Marking of Maize intended for sale in the Republic of South Africa as published in the Government Gazette No. 32190, Regulation No. R. 473 of 8 May 2009. [1]

Proportions allowed per deviation (%)		**Grade**		
		WM1	**WM2**	**WM3**
#1	Foreign matter	(0 - 0.3]	(0.3 - 0.5]	(0.5 - 0.75]
#2	Defective maize kernels, above and below the 6.35 mm round-hole sieve	(0 - 7]	(7 - 13]	(13 - 30]
#3	Other colour maize kernels	(0 - 3]	(3 - 6]	(6 - 10]
#4	Deviations referred to in items 1, 2, 3 collectively: Provided that the deviations are individually within the specified limits	(0-8]	(8-16]	(16-30]

Although foreign material does affect the grade of the maize, for the scope of the project we will be dealing and examining point 2, 3 and 4 only. Table 2.2 illustrates the types of defective kernels one can expect to find during the grading process.

Table 2.2: Defective kernel types according to the SAGL grading Report [2]

Defective kernel types:
Heat Damaged
Frost / Water Damaged
Sprouted
Fusarium (Fungal infection)
Pest Damage
Immature
Soiled (smoke, fire, etc.)
Matter which can pass through 6.35mm sieve

Depending on seasons and harvests, it is quite difficult to obtain enough samples of every single defect type. This year in particular, water and frost damage, as well as soiled, sprouted and immature kernels were particularly hard to find. The deviations chosen were those which were believed being quite difficult to see with naked eye compared to others. Thanks to the SAGL we managed to obtain enough samples of the following kernels and deviations which we will be used throughout the project.

Table 2.3: Available categories provided by SAGL

Available categories		**Approximate sample size (# kernels)**
(a)	White Maize	1566
(b)	Discolored White Maize	1481
(c)	Yellow Maize	1500
(d)	Insect Damaged White Maize	969
(e)	Heat Damaged White Maize	1157
(f)	Fusarium	786
(g)	Severe Fusarium	1257

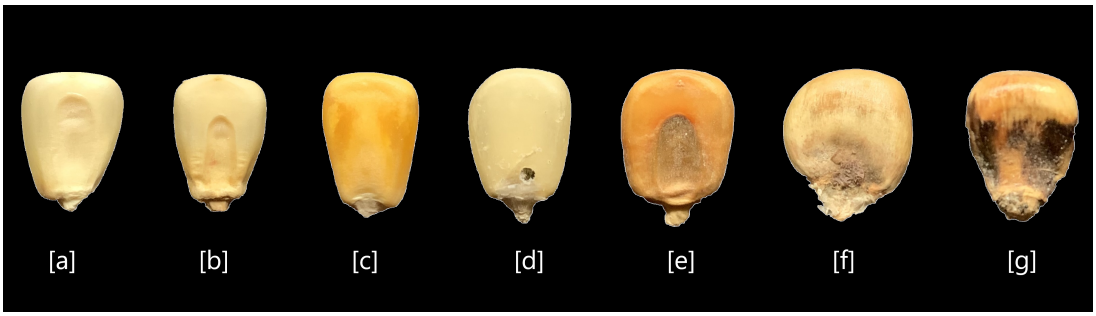


Figure 2.1: Visual representation of each maize category

2.2. Previous related studies

There have been a few approaches in solving this problem of classification, with the two main approaches being, using multi-spectral and hyper-spectral cameras. This allows one to detect all wavelengths of light in a specific range. These methods also involved using principal component analysis coupled with partial least squares discriminant analysis to classify the kernels. The main concept behind PCA is that it is primarily used for dimensional reduction. PLS-DA is a supervised learning model that maximizes the separation between closely related classes. It does this by finding a a linear regression model in the feature space.

The main difference between hyper-spectral and multi-spectral imaging comes down to wavelength resolution. Multi-spectral collects spectral data in a few specified bands, whereas hyper-spectral imaging collects a more detailed continuous spectrum using hundreds of narrow bands. The studies using these systems are documented in [4] and [5] respectively. These studies were conducted on behalf of the Department of Food Science at Stellenbosch University.

The main issue with these solutions is that the price of these cameras are quite high. The typical cost depends on the exact configuration setup which is quite difficult to obtain. However with a response from a employee at HySpex we know that, typical solutions with added equipment are in the range of 45 000 EUR to 65 000 EUR. This is quite expensive even by industry standards. Processing images across a wide spectrum is also computationally intensive. Additionally setting up the cameras and taking images is a lengthy process.

In the multi-spectral case [4] the camera used 19 wavelengths ranging from 375 to 970 nm. In the hyper-spectral study, [5] a comparison was made using the full spectrum model (953 – 2517 nm) and three other camera configurations. These configurations entail reducing the number of spectral channels of the original 288 wavebands into 48, 21 and 13 windows respectively. The accuracies of the two studies, for each of the wavebands are shown below in Table 2.4. As one can see, as the wavebands decrease, so does the relevant

classification accuracy. These accuracies listed will be used as a benchmark to compare with the results obtained using our trained model. Following below is the Table 2.5 summarizing the benchmark accuracies that will be used to test our model against.

Table 2.4: Accuracies obtained in previous studies, where (HI - hyperspectural imaging), (MI multispectural imaging), (WV - waveband),

Class	Accuracy (%)				
	HI (288 WV)	HI (48 WV)	HI (21 WV)	HI (13 WV)	MI
Sound white maize	**88.3**	**76.7**	**78.3**	**63.3**	-
Pinked white maize	**83.3**	**75.0**	**78.3**	**80.0**	**91.43**
Yellow maize	**75.0**	**60.0**	**56.7**	**48.3**	**100**
Defects (average)	**93.3**	**86.7**	**83.2**	**81.0**	**92.86**
- Screenings	93.3	78.3	78.3	73.3	100
- Fusarium	100	88.3	86.7	83.3	88.57
- Diplodia	90.0	90.0	71.7	78.3	82.86
- Heat	95.0	76.7	76.7	61.7	100
- Water	90.0	88.3	81.7	76.7	97.14
- Frost	96.7	90.0	86.7	85.0	-
- Pest	95.8	93.3	82.5	88.3	88.57
- Sprouted	86.7	90.0	86.7	93.3	-
- Immature	92.6	85.2	98.1	88.9	-
Foreign materials (average)	**100**	**96.66**	**95.66**	**94.32**	**100**
- Soy	100	100	100	100	100
- Sorghum	100	98.3	100	98.3	100
- Sunflower	100	100	100	100	100
- Wheat	100	85.0	83.3	83.3	100
- Plant	100	100	95.0	90.0	100

Table 2.5: Benchmark accuracy from previous studies

Available categories	**Benchmark Accuracy %**	
	Hyperspectral Camera	**Multispectral Camera**
Sound White Maize	88.3	-
Discolored White Maize	-	-
Yellow Maize	75.0	100
Insect Damaged	95.8	88.57
Heat Damaged	95.0	100
Fusarium	100	88.57

2.3. Background concepts

This section is aimed to introduce the reader to some key concepts that will be used in the project. The three sections that will be discussed are image registration, image segmentation and convolutional neural networks. All which play important roles in developing a solution to the problem.

2.3.1. Image registration

Image registration is a process which uses feature matching to align two images of the same object that are very similar, but not necessarily identical. The use of this technique is later shown in Chapter 4 where it is used to align top and bottom images of maize kernels. While we will explore the main concepts of image registration, we will not dive deeply into the exact details of each step. The process is briefly explained in the following steps, where [6] is used as a reference source:

Feature Detection:

The first step involves feature detection of binary images, for which we use the SIFT (Scale-Invariant Feature Transform). Given a input image, a few copies of the original image is made, each with a certain amount of Gaussian blur and scaling. These images can be thought of as being stacked on top of each other, such as a "pyramid". At each level of the pyramid we calculate the difference in Gaussian blur (DoG). This is done by subtracting the top image from the bottom, or essentially a blurred version of the image from a slightly more blurred one. What this does is highlights certain key points in a image where spots changes significantly. Once the extreme key points have been found in both images they then need to go through a filtering process called key point localization. This involves removing elements like edge points which are sensitive to DoG and overly bright points.

Orientation Identification

The next step involves determining the orientation of each of the key points. This is done looking at the magnitude of the gradients of the surrounding points. Each pixel's direction is calculated according to this region and is then placed in one of the bins of a orientation histogram. The histogram consists of 36 bins covering 360 degrees. When all the directions of the surrounding area is calculated then the highest bin will be considered as the main "orientation" of the key point. The final orientation is achieved by fitting a function to the three bins closest to the peak.

Key point Matching:

After finding key points, the next step is to find matches from those in the source and target image. A distance metric between each pair is calculated using Euclidean distance. The goal is to find the key point that most closely corresponds to its corresponding key-point in the other image. The matches found are then filtered, choosing only good matches (closest ones). Using these good matches and the orientations the source image is then finally warped to fit onto the target image.

Figure 2.2: Example of two satellite images being aligned using image registration. Reproduced from [7].

2.3.2. Image segmentation

Color thresholding

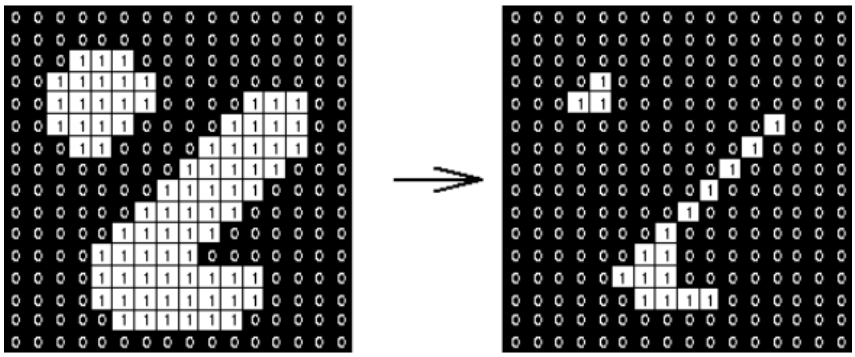
Color thresholding is a simple concept that is used to initially separate the "light" kernels from the darker background in our system. It essentially segments an image based on the color intensities of the pixels. Thresholding can be applied to different types of color spaces such as RGB, HSV, or Lab. In our case the color spaced used is HSV and after thresholding the script creates a binarized image.

Eroding

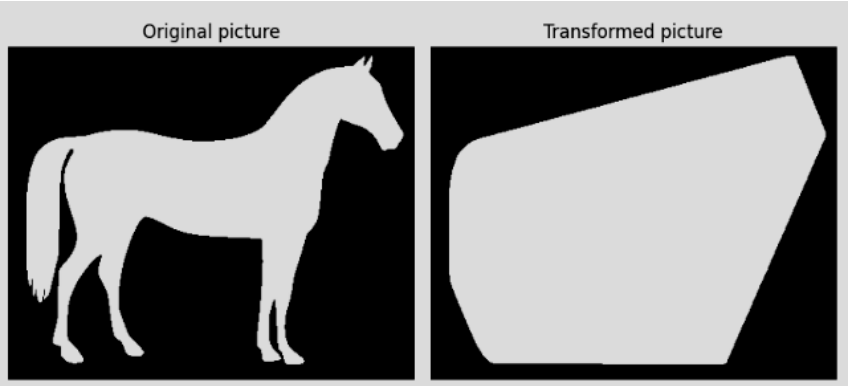
Erosion is used to strip away extrusions of a binary image. This aids in image extraction processes. This is achieved by choosing a square kernel of size say $P \times P$. This kernel is filled with 1's and is swept from the left to the right over the image for each row of pixels. For each pixel the kernel sweeps over, the program checks if the P adjacent pixels match with the kernel. If this is the case then the evaluated pixel is set to 1. Otherwise the pixel is set to 0. Changing the size of the value P will affect how much of the image is essentially striped away. One needs to be careful not to make the size of the kernel too large as then you lose too much information on your image. An example of how this is done is shown in Figure 2.3a.

Convex hull algorithm

After applying the segmentation algorithm, we are left with a mask image. Sometimes some kernels are cut through the middle by the algorithm and not properly segmented. To remedy this, the convex hull algorithm is used to store the entirety of the shape of the kernel. The way in which this works is that the algorithm tries to create the smallest convex polygon that surrounds a white structure in the binarized image. In the case of kernels this works well since we assume that the shape will almost always be convex. A visual example is shown in Figure 2.3b



(a) Erosion using a 3x3 kernel. Image reproduced from [8]



(b) Convex hulling. Image reproduced from [9]

Figure 2.3: Visual examples of erosion and convex hulling on binary images

Watershedding

Watershedding segmentation is a technique that is used to separate objects in an image. Simple contour detection works well when trying to separate objects that are relatively well distanced from each other. However in this case, when taking pictures of hundreds of grain kernels, there is bound to be kernels that are touching each other, sometimes even overlapping. The best way to properly separate and extract these types of images is by means of the watershedding algorithm.

The basic principle of watershedding is to take a gray scale image and then consider the image as topological surface. This means that high pixel values will be seen as “high

surface" regions and low values as "low surface" regions. If one were to imagine 'flooding' the image with water, where the water sprouts from each of the low lying points, then where the water meets from different sources, a "dam wall" is built. At these points a contour line is then drawn to separate each of the objects. This can be best shown in a 1D topographic Figure 2.4

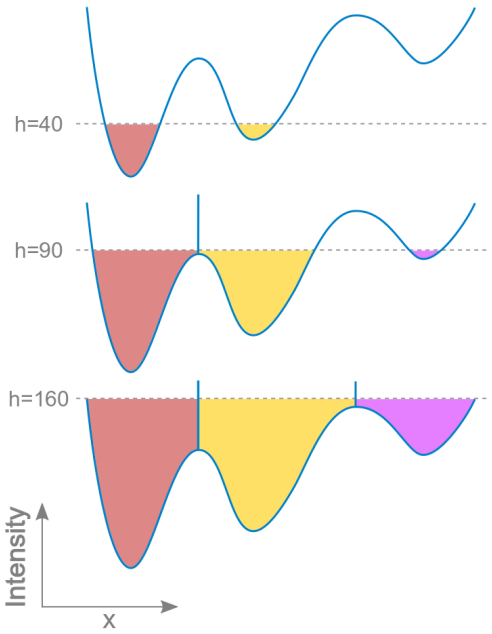


Figure 2.4: Watershedding in the 1D space. Image reporduced from [10].

2.3.3. CNN

Although multiple solutions could be used to solve the image classification problem, it had been decided to used a CNN model as it has shown great performance over the last decade [11]. Unlike PCA and PLS-DA, CNNs extract features at multiple dimension levels rather than a once off dimensionality reduction operation. This allows the network to learn spacial hierarchies of features in images and capture intricate patterns. CNNs are also known for their robustness in handling variability [12].

A CNN is a type of neural network that follows a Feed-Forward Neural Network Architecture which consists of multiple layers. These networks use images with labeled input data ("ground truths") as inputs. In the training process these ground truths are compared to the classes predicted by the model in each leaning epoch. All the layers in the network will not in be discussed in detail although a rough overview will be given. Figure 2.5 is an example of the type of of architecture that will be used.

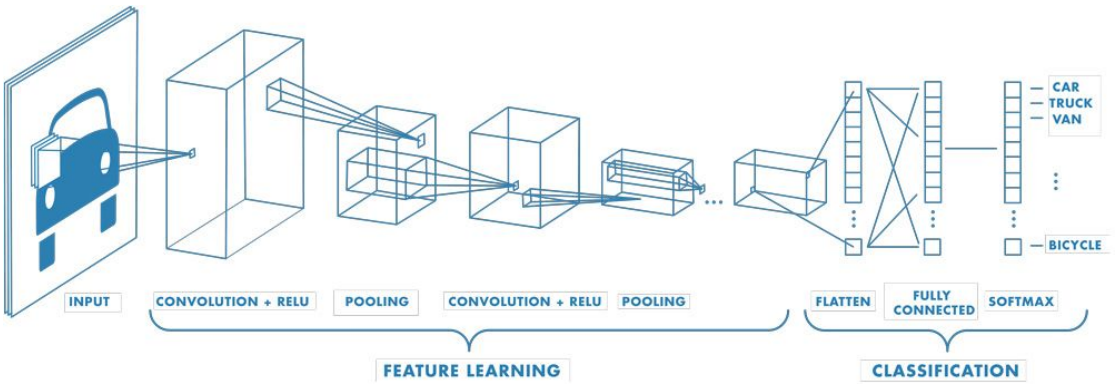


Figure 2.5: Typical CNN classifier architecture. Reproduced from [13]

Convolutional Layer

Even though the network is referred to as a convolutional neural network, it uses a cross-correlation operation for training.

$$G(x[n], h[n]) = \sum_{i=-\infty}^{\infty} x[i] \cdot h[i-n] \tag{2.1}$$

The cross-correlation equation calculates the similarity between two sequences. The use of cross-correlation comes to play when using filters. A filter is a small matrix of size $P \times P$ which "shifts" across the original image from left to right, top to bottom, forming a activation map. The more a patch of the original image "looks" like the filter, the higher the value in the activation map will be. The process allows for detection of meaningful information, which is vital for classifying images [14]. Each filter has a specific set of weights that are updated during the training process. Figure 2.6 visually illustrates this convolution process.

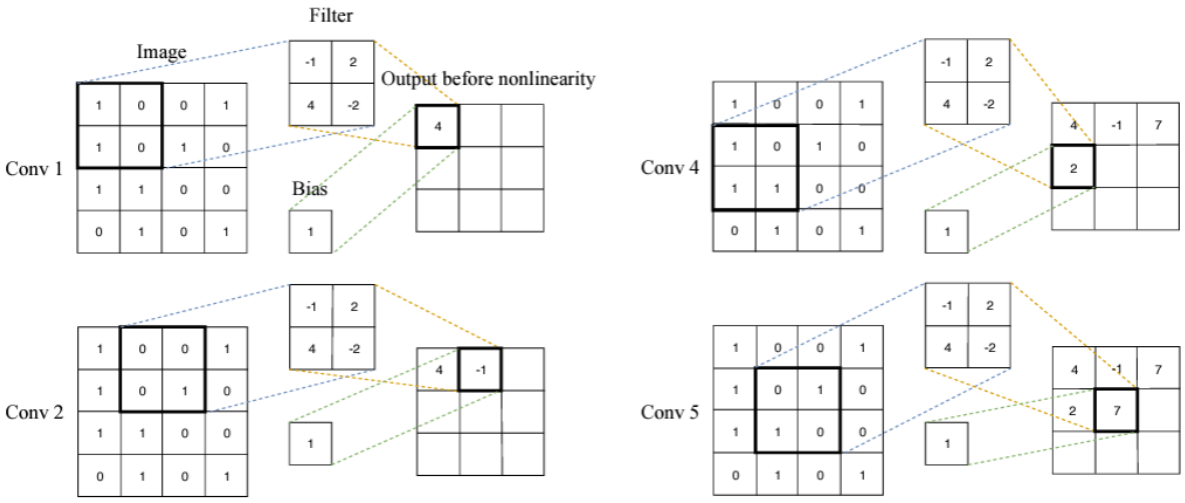


Figure 2.6: A simple visual example of convolution using a 2x2 filter. Image reproduced from [14]

Activation Layer

After the filtering step, the activation map is initially a linear layer. To add non linearity, which allows the network to learn complex patterns, a activation function is added. The typical activation function used is ReLU. Essentially ReLU sets all values that are less than 0 to 0, otherwise anything above remains unchanged. This activation function also helps address the vanishing gradient problem in training. [14]

$$f(x) = \max(0, x) \tag{2.2}$$

Max Pooling Layer

To minimize the spatial volume of the input image, a max pooling layer is used. Reducing the spatial size means that the images requires less weights and thus the eventual training is less computationally expensive. Max pooling is done by calculating the maximum value of each window frame and then storing this with a new output dimension.

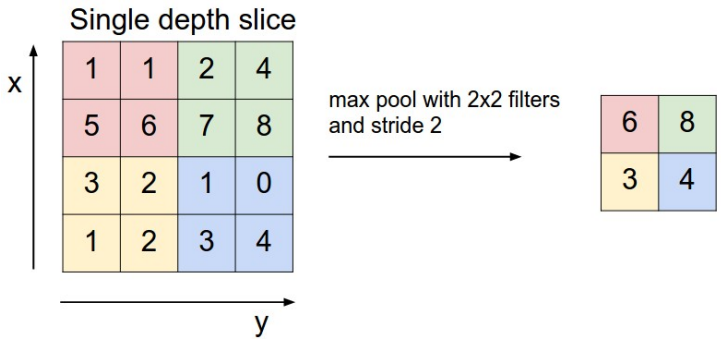


Figure 2.7: A simple pooling example using a 2×2 filter and stride of 2. Image reproduced from [15].

Fully-Connected Layer

Finally, the output of the layers is converted into a one-dimensional vector using flattening. In this layer, each neuron, with its now flattened weights and biases, is interconnected with each other. After simple multiplication is done with each of these neurons a softmax function is then used to finally classify these features into one of the available classes. The softmax function requires a 1-dimensional input and is thus the reason why the flattening layer is necessary. The classes predicted for each image are then compared to their actual classes and the mean squared error or some other loss metric is calculated. Finally this loss is used to update the weights and biases in the network as it progressively improves its ability to predict accurately. This is done through a process called back-propagation using one of several standard stochastic methods such as gradient decent.

Loss function

The loss function used in training is the Sparse Categorical Cross-Entropy loss. This allowed us to provide labelled data of type [0,1,2,3,4,5] instead of having to one hot encode all of our data.

$$L(y, \hat{y}) = -\sum_{i=0}^{5} y_i \log(\hat{y}_i)$$

Optimizer

For the training process the Adam optimiser was chosen. The optimizer is popular since it combines two other optimizers namely; RMSprop and Momentum. Momentum takes into account the past gradients so that the "jump" rate is more smooth in each iteration. This accelerates the convergence rate and prevents the model from getting stuck at local minima. RMSprop uses a moving average of squared gradients to normalize the gradient itself. This can make the learning process more adaptive and resilient to issues like exploding or vanishing gradients. To calculate the adaptive learning rates for each parameter the and second moments of the gradients are calculated as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \qquad v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

Where:

- g_t is the gradient of the loss to the parameters at iteration t
- m_t is the first moment/mean
- v_t is the second moment
- β_1 and β_2 are hyperparameters that control the decay rates of the moment estimates

The parameter updates are calculated as:

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Where ϵ is a small constant to prevent division by zero.

Batch Normalisation

When the batches are not normalised, the network might update one weight significantly more than the other. What happens then, is that the gradient descent trajectory will fluctuate back and forth along one dimension, resulting in a longer time to reach minima.

By implementing batch normalization each weight update becomes more balanced across the dimensions. Each batch is normalized as:

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

Where:

- $x^{(k)}$ is the activation of the k^{th} input.
- μ_B is the mean of the activations in the batch.
- σ_B^2 is the variance of the activations in the batch.
- ϵ is a small constant added to prevent dividing by zero.

Dropout

A regularization technique used in training is dropout. At each batch iteration it sets a random fraction of input units to zero. These units are "ignored" in a sense. The whole idea of doing this is to promote independence among neurons and prevent co-adapting. (Making sure that neurons don't become overly reliant on neighboring ones - that is preventing these units to fix up the mistakes of other units). Placing the dropout layer too early in the network can impede learning so commonly it is added in the final layer of the network.

Popular architectures

Over the past few decades there have been multiple architectures of CNNs that have stood out. LeNet being one of the first networks used for handwriting recognition. VGGNet and AlexNet which have performed esspecially well in object recognition. Also noteworthy is ResNet which introduced residual connections and GoogleNet which added an inception layer [16]. It had been decided that the LeNet structure is the most suitable for our problem because of its relatively simplistic design and robustness. The other architectures are built with very deep networks for intricate feature detection. In our case this extra depth is most likely unnecessary and would only provide a marginal performance increase.

2.4. Chapter summary

In this chapter, we considered the regulations regarding maize grading as well as provided some background concepts about image processing and Convolutional Neural Networks. In the next chapter, we will be considering the hardware design to actually start capturing the data.

Chapter 3

Hardware design

This chapter gives a overview of how the system works. It dives into the hardware components and the architectural design of the camera box system, as well as the configurations and explanations for various design choices made.

3.1. Components

3.1.1. Raspberry Pi 4 Model B

When a image is taken it will have to undergo intensive processing such as image registration, watershedding, etc. In Chapter 4, we go into more detail regarding this. It is also required that the processing should be fast enough for high quality images. A basic single-board computer (SBC), such as the Raspberry Pi Zero with its limited 512MB memory and 1GHz processor, would thus not be able to handle the tasks.

The Raspberry Pi Model 4 B is chosen as it is the newest in range. It offers a improved performance with a 1.5GHz processor and 2GB of memory. It also has two micro HDMI ports, which allows visualizion of the camera feed on a monitor. The USB ports enable the use of a wireless mouse and keyboard so that one can control the device without having to touch it once the camera has been aligned. The way in which the SBC is designed allows for large amounts of storage on either an SD card or some other USB device. More specifications of the Raspberry Pi's can be found at, Zero [17], Model 4 [18].

3.1.2. Raspberry Pi Camera Module 3

The Raspberry Pi camera module 3 has been chosen. The module 3 is equipped with a 12MP camera with the highest square resolution of 2592x2592. The camera comes with auto focus capabilities which is useful as it allows for sharp images of the kernels regardless of the distance from them.

3.1.3. Camera holder

To keep the camera module and the SBC attached to each other and held in one place, a simple 3D printed attachment was made. Figure B.1 in the appendix shows how the attachment looks like.

3.2. Camera system concept designs

When designing the camera system a iterative approach was taken. Firstly a small system was made in which most of the foundational functionality was tested and validated. Thereafter to improve the system and address some of its shortcomings a second iteration design had been developed.

3.2.1. First prototype design

This initial design was quite simplistic. Consisting of a open box to place the kernels in and a tripod holding the SBC, camera and light source. In in this case the images that were taken were of high quality and proved satisfactory. Most of the image processing software had also been tested and validated. With satisfied results the next step was moving into a more advanced system which addressed some of the initial shortcomings. Some of the shortcomings of the initial system are as follows: firstly one would like to take images in a consistent environment. In this case outside light has a significant affect on the images taken. One would also like to do full analysis on the kernels and in this design only images of one side of the kernel is taken. Essentially half of the information is lost.

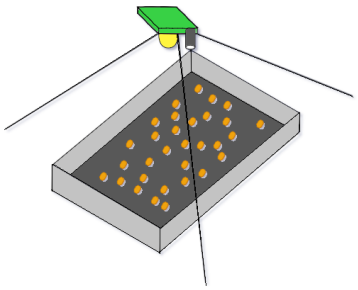


Figure 3.1: Initial tripod setup.

3.2.2. Photo booth iteration design

For the second iteration, a design, resembling somewhat of a “photo-booth” had been developed. This consists of a wooden box approximately a meter in height with a opening on one side in which a batch of maize is slid into. This box creates a consistent environment for images to be taken. The maize would be set on a glass platform which also allows images to be taken of both sides of the kernel. Ideally one would want to have two cameras. Due to the scope of the project only one camera is used which will be manually moved from the top to the bottom between each set of images taken. For the rest of the Chapter the focus will be on the specific design choices of this second iteration.

3.3. Photo-booth design choices

3.3.1. Kernel container

The kernel container will be used to place the maize batch samples in. The choices for size and platform design are discussed below.

Size

The size of the platform had to be carefully considered since it is undesirable for kernels to stack on top of each other, blocking the view. The aim was to choose the size, so that all the particles can easily fit and lie flat on the surface. The platform's size could also not be unpractically large. Too much wasted space would mean that the camera would have to be placed higher, which would decrease the overall image quality. The trick lies in minimising height while maximising space.

After weight tests were conducted, it was determined that a 150 g sample contains approximately 375 kernels, with each maize kernel weighing approximately 0.4 g. From measuring multiple kernels was also assumed with a high degree of confidence that the average square cutout area of a kernel side is approximately $160\,\text{mm}^2$. The minimum total surface area needed to hold these kernels is then given by:

$$\text{Total surface area of platform} = 375 \times 160\,\text{mm}^2 = 60,000\,\text{mm}^2$$

To create more space between the kernels, a $30\,\text{cm} \times 30\,\text{cm}$ container is used with a total surface area of $90,000\,\text{mm}^2$. This ensures that the kernels are reasonably well spread.

Glass platform

The kernel container is equipped with a 32cm $\times$ 32cm transparent glass sheet, which allows visibility from both sides. However, using glass does come with some challenges. The pictures taken through the glass are not as sharp as those taken directly. Refer to Figure 5.3 in Chapter 5 for a direct comparison. Another drawback, is that when kernels are placed on the glass repeatedly, after a while the glass gets dirty and must be cleaned. The glass also produces reflections when LEDs shine on the surface. To counter this the LEDs are placed at a specific height and location, as detailed in the next section. There are other alternatives like plexiglass, but the photo quality is not as sharp. Things like anti-glare glass does exist, which doesn't completely take away the glare but reduce it significantly. The price of such glass is quite expensive. Nevertheless, for the purposes of this project, the standard glass sheet performed well and proved satisfactory.

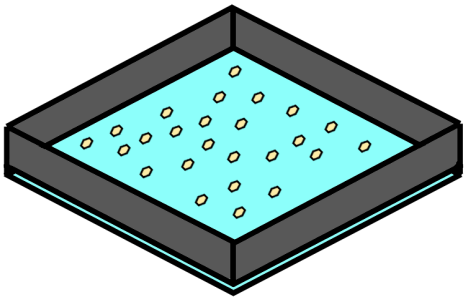


Figure 3.2: A 32cm× 32cm maize container with glass platform, which slides into larger box.

3.3.2. Camera height

According to the Raspberry Pi Camera Module 3 datasheet [19] the standard camera has a 75° diagonal field of view. The design is that of a square block and thus requires that the pictures be taken in the square format. Consequently this means that the image size is limited by the vertical field of view which according to the datasheet [19] is 41°. Using simple trigonometry one can calculate the optimal height in which the camera should be placed. The final height equates to about 430 mm.

$$\text{Camera Height (mm)} = 160 \tan(69.5°) = 427.9\,\text{mm} \approx 430\,\text{mm} \tag{3.1}$$

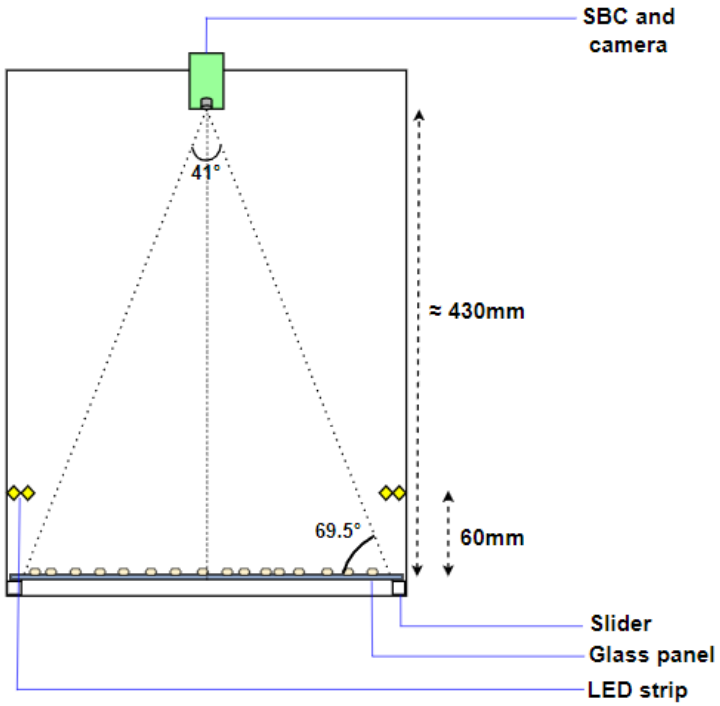


Figure 3.3: Side view of box.

3.3.3. Lights

To ensure that the kernels are all equally well lit, an LED strip has been placed all around the inside of the box. The LEDs are positioned 6 cm in height from the surface of the glass on both sides. Placing the LEDs this way almost entirely removes any potential shadows. By placing the LEDs on the sides of the box as opposed to the top is to counter the effects of reflections on the glass. Essentially, the reflected area of the glass will only be confined to the edges and nowhere else.

To deal with the slight amounts of reflections, a non reflective covering was placed on the edges of the glass. This strip slightly reduces the usable size of the container space. In this case insulation tape was used and seemed to perform well. There is however still room for improvement, as shown later in Chapter 5.

The 6 W LED strip [20] is connected in series to a 9 V battery and resistor. After running multiple tests using different resistor values it was empirically determined that a 470 Ω resistor value provided the best perceived brightness level through the camera preview.

For the bottom circuit, it is important to remember that the glass absorbs and reflects light. So in order to achieve the same lighting conditions the LEDs had needed to shine brighter. This required a lower resistor value to be used in the circuit. Specifications of the glass's transmittance could not be found online, however for regular plates the common range for visible light transmittance is around 85-90% [21].

To calculate a very rough estimate of the bottom resistance value two broad assumptions are made. The first is that the glass absorbs approximately 10% of light. The second is that the brightness is somewhat linearly proportional to the current flowing through the circuit. The second assumption is not entirely true since brightness is not perfectly linearly proportional to the power of the LED. This is due to Gamma correction [22].

$$
\begin{aligned}
\text{Total Resistance in Top circuit} &: R_{\text{Top}} = 470\Omega \\
\text{Current for Top} &: I_{\text{Top}} = \frac{9V}{470\Omega} = 19.15 \text{ mA} \\
\text{Current for Bottom} &: I_{\text{Bottom}} = 1.1 \times I_{\text{Top}} = 21.065 \text{ mA} \\
\text{Total Resistance for Bottom} &: R_{\text{Bottom}} = \frac{9V}{21.065\text{mA}} = 436.36\Omega \\
\text{Bottom Resistor Value} &: R_{\text{Bottom}} = 427.25\Omega
\end{aligned}
$$

3.3.4. Sliding sheets

To create a solid background for the kernels a black cardboard sheet is slid into the top and bottom of the box. Cardboard was used as it is rigid enough to stay firm in one place without having creases, and thin enough to slide in and out underneath the container without moving the kernels. The following diagram shows how the sheets are slid out from the bottom and on top of the container after each photo is taken.

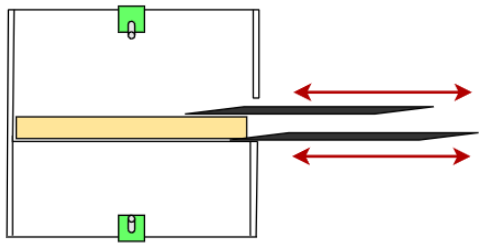
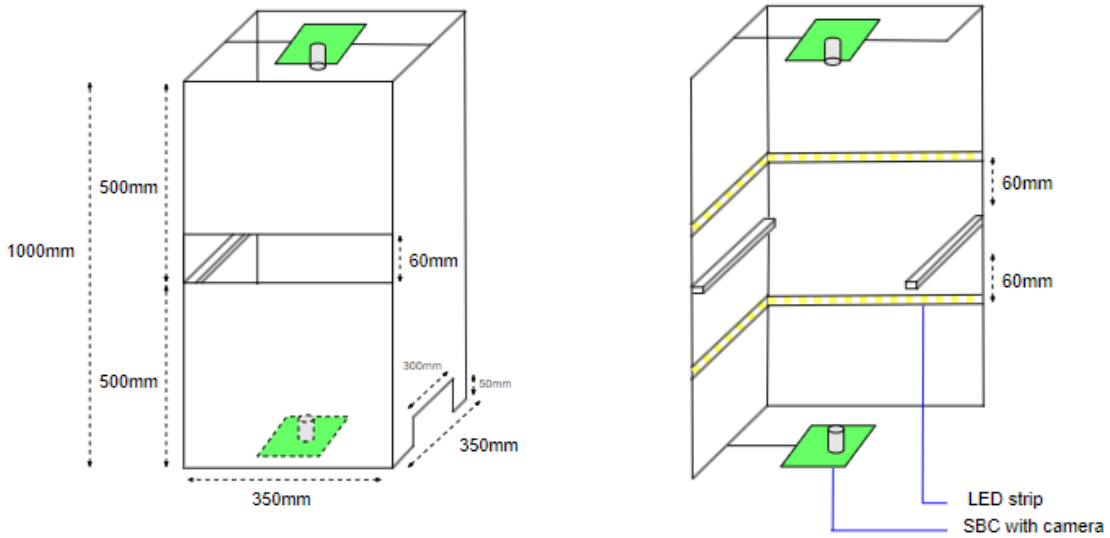


Figure 3.4: Side view, showing the sliding of the black sheets. Note; diagram is not to scale.

3.3.5. Box

The specific dimensions of the box are show below in Figure 3.5. The inside view shows the built in internal slides which allow the container to be slid into. The green rectangles represent the SBC with the camera and the yellow line represent the LED strip. Images of the entire setup can be found in the Appendix B.7.



(a) Outside view **(b)** Inside view

Figure 3.5: Photo-booth design

The process for capturing photos is as follows:

1. Activate LEDs and position the camera in place.
2. Place 150 g batch into a smaller container and make sure the kernels are well spaced.
3. Slide a black covering within the box.
4. Slide kernel container above the covering.
5. Take the top photo.
6. Slide the covering from out the bottom and place it on top of the kernel container.
7. Take the bottom image.

3.4. Chapter Summary

This Chapter gave an overview of the entire hardware system and described some of design choices that were made. The next Chapter will focus on the image processing software, detailing how system processes the top and bottom images once they have been captured by the hardware system above.

Chapter 4

Software and System Design

This chapter separates the software design into three sections, namely: Processing, Training and Evaluation. Once images have been captured, they are initially subjected to the processing stage. Thereafter, the images are used either for training or evaluation.

4.1. Image processing

This section is arguably the most complex and fundamentally important of the project. The images captured using the system in Chapter 3 are processed in order to extract individual kernels, matching them with top and bottom sides and then further preparing them for the network input. In the training case these images with their ground truths are stored and used to train the network, while in a normal setup the images are passes without labels though the network. This processing stage is further divided into 5 parts. The Figure 4.1 below highlights each step and the order in which they happen.

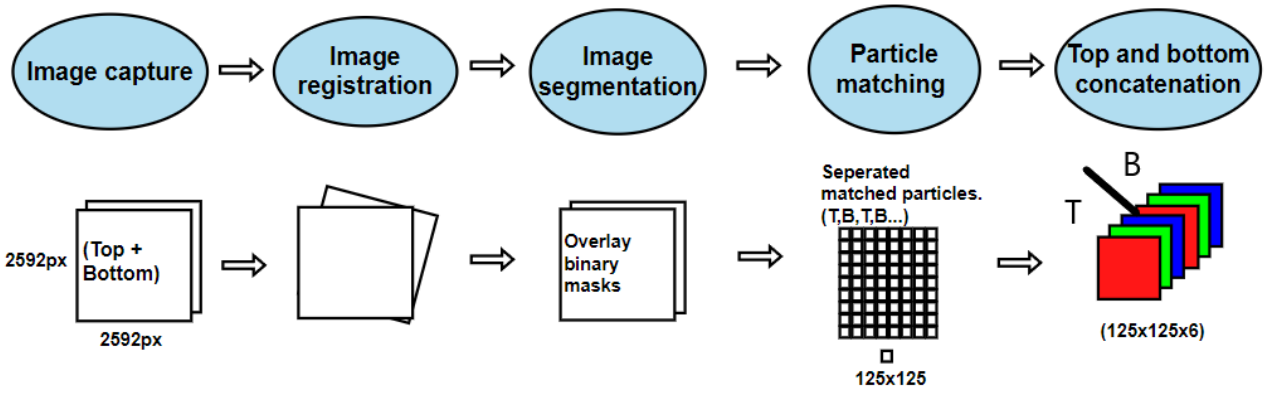


Figure 4.1: Image processing flow diagram.

4.1.1. Image capture

A python script, using the PiCamera2 library is used to capture the images. In the script the auto focus is enabled and the image resolution is set to the maximum square format of 2952×2952 pixels. A preview display is shown to the connected monitor. This allows the user to fine tune and adjust the camera's positioning as needed.

4.1.2. Image registration

The goal after taking the top and bottom images is to have them perfectly aligned so that kernels can be identified. Being limited to one camera, it forces one to have to constantly move it between the top and bottom sections of the box. This movement causes inaccurate placements which induces some skewness between the two images. To combat this, image registration is implemented in software. A python script using the OpenCV library is used to implement this functionality. The Figure 4.2 in appendix shows the before and after effects this has on our images.

Figure 4.2: Before and after effect of image registration. Left: (Original images of top and bottom). Right (Top and bottom images after registration script).

4.1.3. Image segmentation

Once the images are correctly aligned, the next step is to extract the kernels from the image and save them as their own PNG files. We will delve deep into each step of this process. For most of these steps, the image processing tool Fiji [23] was used as it performed more robustly compared to other tools. The following sections are postfixed with letters corresponding to Figure 4.3.

Color thresholding (b)

The first step in the image segmentation process is to distinguish all kernels from the background. A black cardboard background works well here as the kernels stand out. To choose the values which capture the kernels and not the background, extensive ‘eyeballing‘ tests were done to see if the range covers all the particles entirely. The lightest particles are the white maize and the darkest are fusarium. Figure 2.1 gives a good representation of the color ranges. In order to threshold the background out, while still maintaining to capture all the types of kernels, the HSB color space was used and the final values were chosen as: H = 0-255; S = 0-255; B = 35-255. Where, the Hue value represents all the colors on the color wheel. The Saturation value represents all the types of shades of gray. The Brightness values represents all colors that are above 14% brightness on the standard

scale. (0 black and 100% white). The HSB color space was chosen above other spaces such as RGB, is because brightness is the key separating distinguisher between the kernels and background. In Fiji [23] this thresholding takes place and a red mask is drawn over the pixels that fall into this range. This red mask is shown in Figure 4.3b

Binary mask and eroding (c)

The next step involves converting the mask image into binary format. This is done by assigning a value of 0 to the background and a value of 1 to the foreground. Foreground referring to the red pixels that cover the area of the kernels. Since many of the kernels are packed closely to each other, slight erosion is applied to create space between them. The reason as to why erosion was used and not other techniques like opening, is because we ideally want to maximise the space between the kernels as much as possible. Some of the information of the particle is temporary lost by doing this but is later recovered. Maximising space between the kernels significantly aids in the watershedding process.

Watershedding algorithm (d)

A initial watershedding algorithm runs and draws contour lines between each of the kernels. The fundamentals of this algorithm is explained in Chapter 2. Although this algorithm correctly separates most of the kernels, some are still not correctly separated. In some cases kernels are cut through the middle by the algorithm, refer to Figure 4.3d. To remedy this, the Convex Hull algorithm is introduced as a subsequent step.

Convex hull algorithm (e)

This algorithm essentially aids in the separation by filling in holes in the kernels that were incorrectly made by the watershedding algorithm. What one has after this is particles that are mostly correctly selected and also not as tightly overlapping. This step also groups some of the kernels on the touching edges, however from the initial watershedding step there are still many contour lines remaining that separate the majority of the kernel.

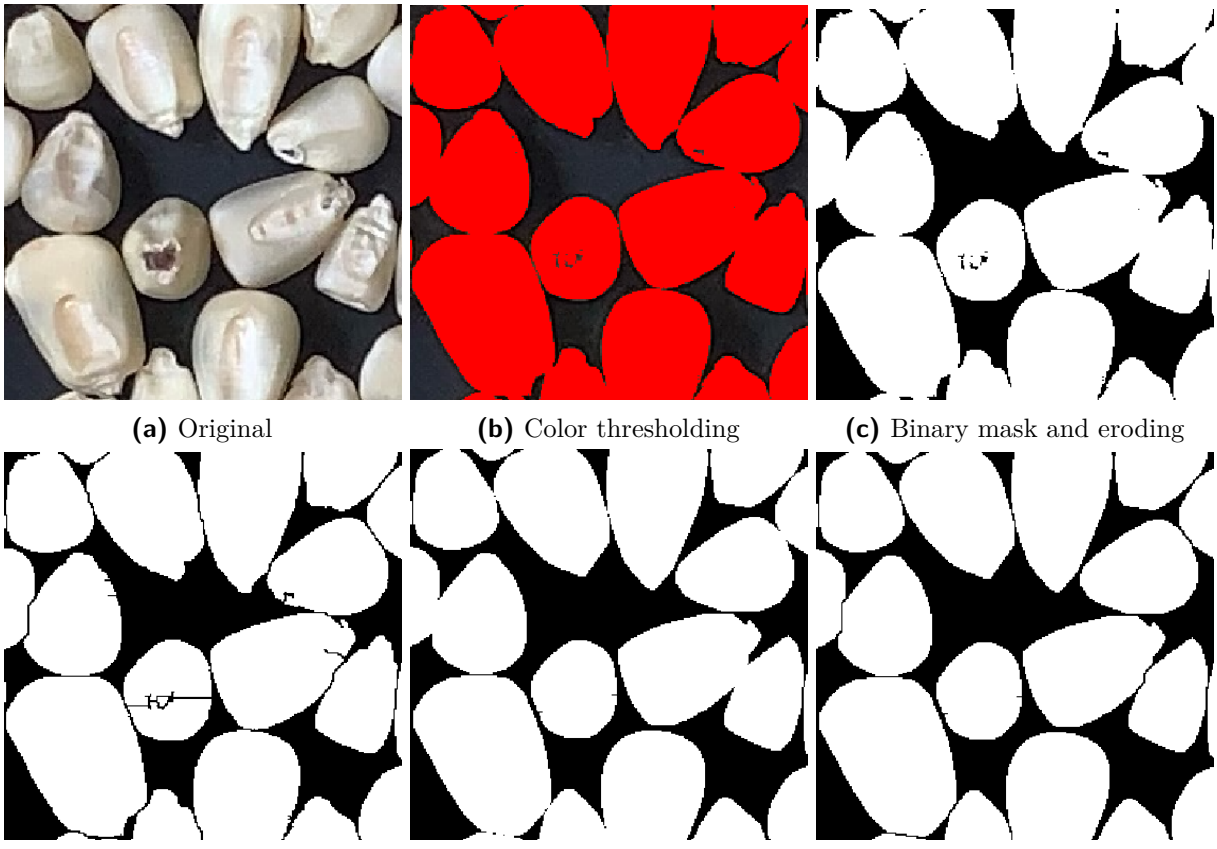
Final water shedding, contour detection (f)

Now that the particles are in a cleaner format, one can run the water-shedding algorithm once more. We are then left with a final mask which is then overlaid onto the original image to extract the kernels.

Example

Below in Figure 4.3 is an example of how the image segmentation process works on a small subset of kernels. For this example a unique case was chosen. Take note of the circular

kernel in the middle. This kernel can be seen as a problematic case as it is lying face down. The kernel is initially incorrectly separated into two separate structures and then is enclosed after the convex hulling step. This indicating the robustness of this algorithm even in adverse conditions.



(a) Original **(b)** Color thresholding **(c)** Binary mask and eroding

(d) Water shedding **(e)** Convex hull **(f)** Final water shedding step

Figure 4.3: Visual examples of each image segmentation step.

The next step is to actually extract the kernels from the image. For this a python script had been written that uses OpenCV's contour detection functionality to determine the coordinates of each of the particles. Contour detection, to put simply, selects all the pixels which are of the same color. Before kernels can be saved they first need to go through the particle matching process which is further explained in next step.

4.1.4. Particle matching

Implementing these image segmentation steps does not always yield perfect results. Sometimes small particles, glares and incorrect splitting happens. While missing a few kernels would not have such significant affect on the grading, what we don't want is to feed the model with faulty/incomplete images. Thus two checks have been added to decrease the chance of this happening.

After running some manual tests, is was determined that the smallest kernel will not have an area smaller that 2200 pixels. A size check using this threshold is added to eliminate all small unwanted particles.

The next step involves using a matching principle for the top and bottom image. What this entails is as follows: The contours are used to create a square cutout of each of the particles. The centres of these square images are then determined as coordinates. The top side center is then compared to the bottom side center. If the euclidean distance measured between the top and bottom centres are less than 10 pixels then a match is found and the kernel pair is saved. A length of 10 pixels may seem little, but after image registration the kernels are almost exactly aligned. An assumption is made that if the image segmentation process runs on two images, then normally the same mistake will not occur on both sides of an image. This step effectively filters out almost all discrepancies.

Figure 4.4: A snippet showing how pairs of kernel pairs are saved. Where "#_T" is top and "#_B" bottom

4.1.5. Dual image concatenation

After a pair have been correctly identified, the final step is to convert the images into a format suitable for the neural network. To do this one needs to somehow combine the two images. The decision was made to have the input space being 6 layers deep, consisting of two images each with dimension 125x125 pixels and in RGB format. By using a python script with OpenCV functionality we perform horizontal axis concatenation to create the final input shape of (125x125x6).

4.2. Training

This section provides detail on the samples that we were received from the SAGL as well as how the model was trained.

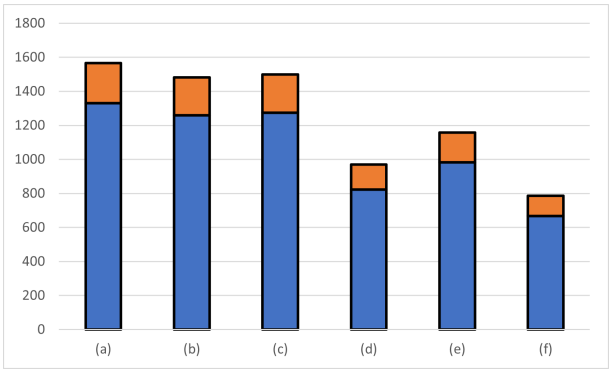
4.2.1. Data balance and split

We initially had 7 categories of maize, among which there is normal and severe fusarium. It was decided to remove the "severe" category in training. The reason for this is due to the fact that fusarium in the practice of grading isn't classified into separate classes. Additionally, it seemed according to personal inspection that there were some overlap between these categories in the batches provided. Another worry is that by having both classes, potential bias would be induced. Since these defects don't occur that often in the real world it was important not create an overemphasis on them in the training set. Furthermore it was decided to split the data into training and validation sets with 85:15 ratio. There is no test set allocated since the SAGL has provided separate graded batches on which final testing is done. The Table 4.1 and Figure 4.5 below show the available kernels and the training spilt.

Table 4.1: Amount of extracted kernels from each category for training.

Category	# Kernels
(a) White Maize	1566
(b) Discolored WM	1481
(c) Yellow Maize	1500
(d) Insect Damaged	969
(e) Heat Damaged	1157
(f) Fusarium	786
Total	**7459**

Figure 4.5: Visualisation of class balance. Where blue training set. Orange validation set



4.2.2. Model architecture

A well designed structure that has been used in the past for image recognition is LeNet-5. This architecture was used as a baseline to build our model off. LeNet-5 used input images of size 32x32 with 3 convolutional layers. A good visual example can be shown in Appendix Figure D.2. The input structure we have following the image processing is (125x125x6). To, in a sense, scale this to the 32×32 size, two convolutional layers were initially added. Following this is then the normal LeNet-5 architecture. Our structure also consists of 5

convolutional layers as well as a global-pooling and a dropout layer. Smaller kernels of size 3x3 were used as they performed slightly better than larger kernels. This could be due to the fact that the larger kernels detected too much unnecessary details in each image. Additionally a dropout percentage of 50% was used in the final layer. Table 4.2 shows the overall architecture of our network.

Table 4.2: Final model architecture

Layer (type)	Output Shape	# Parameters
`Original Image`	(125, 125, 6)	0
`Conv2D`	(125, 125, 16)	880
`BatchNormalization`	(...)	64
`MaxPooling2D`	(62, 62, 16)	0
`Conv2D`	(62, 62, 32)	4640
`BatchNormalization`	(...)	128
`MaxPooling2D`	(31, 31, 32)	0
`Conv2D`	(31, 31, 64)	18496
`BatchNormalization`	(...)	256
`MaxPooling2D`	(15, 15, 64)	0
`Conv2D`	(15, 15, 128)	73856
`BatchNormalization`	(...)	512
`MaxPooling2D`	(7, 7, 128)	0
`Conv2D`	(7, 7, 256)	295168
`BatchNormalization`	(...)	1024
`MaxPooling2D`	(3, 3, 256)	0
`GlobalAveragePooling2D`	(256)	0
`Dense`	(128)	32896
`Dropout`	(50%)	0
`Dense`	(6)	774
Total params		428694 (1.64 MB)
Trainable params		427702 (1.63 MB)
Non-trainable params		992 (3.88 KB)

4.3. Evaluation

This section describes the samples that were provided that will be used for testing the neural net.

4.3.1. Testing data

For the testing phase, SAGL provided five batches of graded samples containing each a mixture of defective and non-defective kernels. These batches had arrived near the end of the project deadline and after the neural network had already been trained on the original

samples. There was however a slight miscommunication in terms of the type of batches to expect and this lead to the following two obstacles which were initially unaccounted for in development:

Each of the 5 batches that arrived were 250g in size, which is 100g larger than what the system was designed for. To address this the batches had simply been split in half and required two sets of images to be taken.

As previously mentioned one of the limitations of the system is that small defective kernels below 6.35mm are not accounted for. The batches that arrived had much of these small shavings in. These particles had a significant impact on the image segmentation process and in some cases were the direct cause for a batch to be classified as a lower grade. Furthermore, some of the batches had other types of defective kernels that were not been included in the training stage. Diplodia and YM (yellow maize) insect damage are two examples of this. To address this, these particles were manually removed in the testing process. Additionally the weights of the given grades were also adjusted accordingly. This is further explained in Chapter 5

Below are the examples of two batches. The batch on the right contains the small particles below 6.35mm whereas the one on the left does not. Accompanying this are the resulting masks which illustrates the significant impact these shavings have on the segmentation process, making clean segmentation of the maize kernels very challenging.

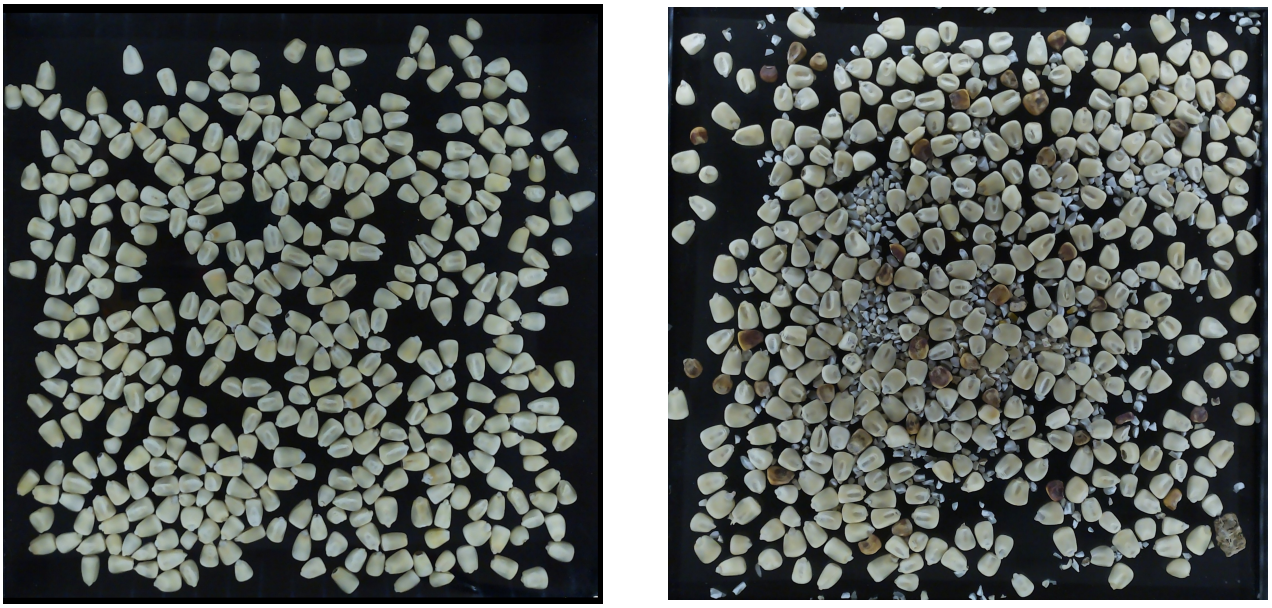


Figure 4.6: Original images of a “clean” and “dirty” batch.

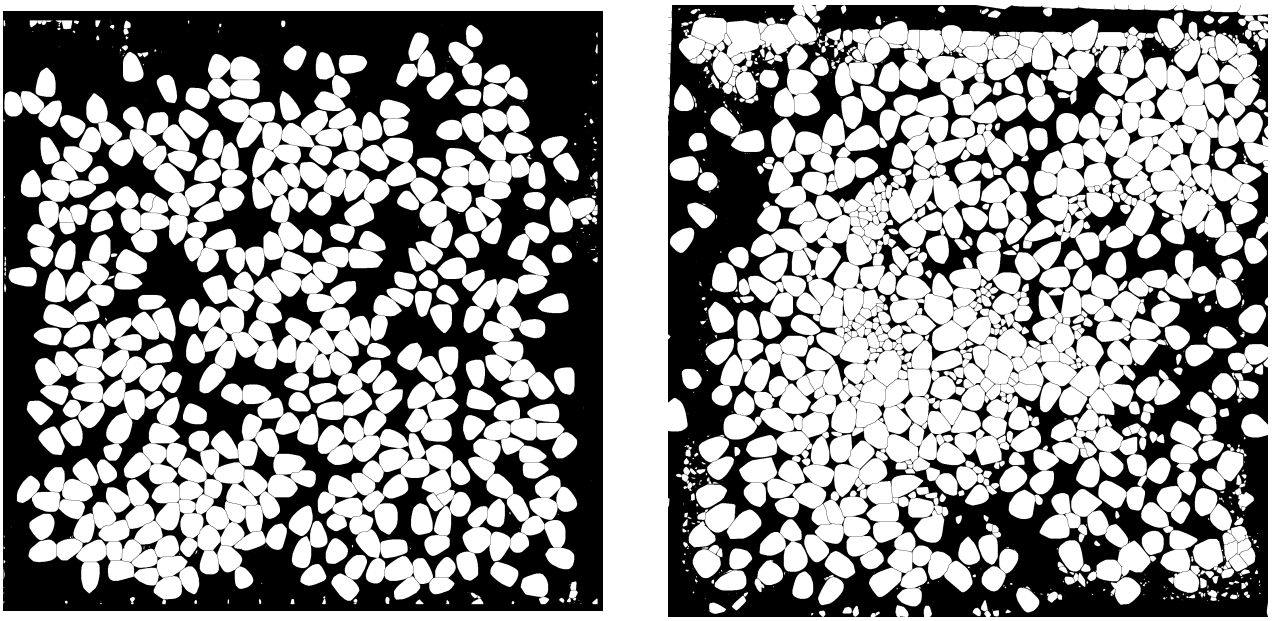


Figure 4.7: Final masks of a “clean” and “dirty” batch.

4.3.2. Output

After processing the original images the evaluation script is run which inputs the particles into the model. A report is then generated by using the regulations Table 2.1 as reference. The report shows the proportions of each class and displays the final grade of the batch in the terminal. Figure 4.8 is a example of how the report is displayed in the terminal. The “Other Colored” category refers to the sum of discolored kernels and the non primary white/yellow kernels in the batch. The “Defects above 6.35mm” category refers to the sum of fusarium, insect and heat damaged kernels.

```
-----------------------------------------
| White Maize          :   88.62% ( 296)   |
| Discolored           :    1.50% (   5)   |
| Yellow Maize         :    1.80% (   6)   |
| Insect Damage        :    0.90% (   3)   |
| Heat Damage          :    6.29% (  21)   |
| Fusarium             :    0.90% (   3)   |
-----------------------------------------
| Defects above 6.35mm:    8.08%          |
| Other colored        :    3.29%          |
-----------------------------------------
| Combined deviations :   11.38%          |
-----------------------------------------
| Final Grade          :   WM2            |
-----------------------------------------
```

Figure 4.8: Terminal output of a sample batch.

4.4. Chapter summary

In this Chapter we discussed image processing and training system. The results obtained from this will discussed in detail in the next Chapter.

Chapter 5

Results

This chapter describes the performance of the image extraction process, training, execution time and tests on unseen samples.

5.1. Image extraction

Running image segmentation on batches of kernels without proper dividers will seldom yield perfect results. This is due to the nature of kernels being small in size and bunching up. As mentioned in the prior chapter, the batches that are evaluated are those with small screenings and shavings removed. Doing image processing with these small particles included are beyond the scope of the project.

In general, the script that extracts kernels performs reasonably well but never perfectly. On average around 1% to 5% of kernels are not extracted cleanly and are thrown out by the validation script. The primarily factor influencing how much are missed seems be in direct correlation to the spread of kernels on the platform. This is unfortunate and has a rather big impact on the grading. The kernels that are missed seem to be completely random. In some cases the algorithm decides to slice the kernels straight though the middle. The kernels in Figure 5.1 is by all mean distinguishable to the naked eye, but is incorrectly segemented by the algorithm. This is most likely a consequence of applying watershedding algorithm to a binarized image. In other instances, the kernels are slightly overlapping which causes the algorithm to group them as one. Luckily both these cases are flagged as "non-kernels" by the center and size checks in software. For interest sake when evaluating the system on the batches with small shavings included, this "miss" rate jumps up to 15% to 20%, which is not adequate performance for real world use.

Another thing that seldomly happens is that a little bits of glare still seem to pop up on the edges of the glass. This glare appears on the insulation tape and by chance the algorithm splits some of these top and bottom sides into two squarish shapes. This is illustrated in Figure 5.2. This squarish shape then bypasses the center and size checking scripts and is seen as a kernel. Luckily this does not happen very often and is a minor issue that can be very easily solved. One can use better anti reflective material for covering, or more advanced colour checks or simply add a cropping step to the image.

Figure 5.1: Example of a kernel that has been unexpectedly cut through middle. Original (left); Mask (Right);

Figure 5.2: Example of squarish shape glare on top and bottom sides which is misclassified as a kernel.

The impact that the glass has on the quality of the images is shown in Figure 5.3. There is overall not much of a difference but looking carefully, one can notice a slight shade of blur on the bottom image, making finer details less apparent. However for the purposes of this model these slight differences have little effect on the overall performance.

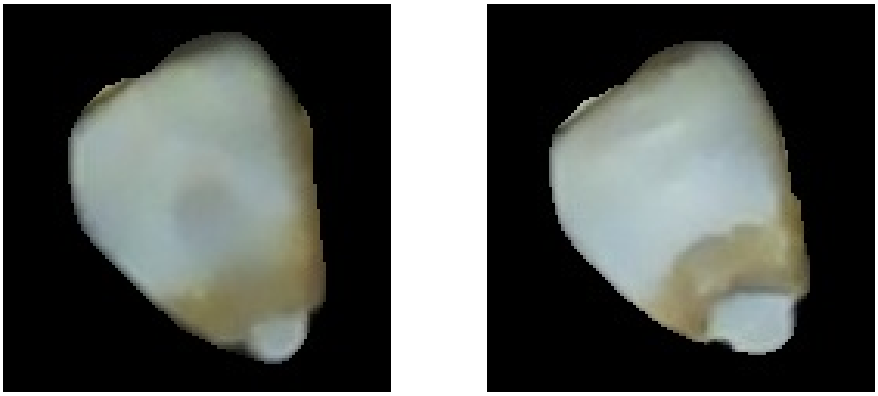


Figure 5.3: Image quality through glass comparison. Glass: (Left / bottom side). Direct: (Right / topside).

5.2. Training results

The model achieved a training and validation accuracy of 0.9777 and 0.947 respectively. These results indicate a very good fit on the training data and a quite high validation accuracy suggests that the model should generalize well to unseen images. Throughout the training process it was observed that the accuracies plateaued at around 50 epochs as shown in Figure 5.4. The validation accuracy experienced significant fluctuations up to that point. Up until around the 38th epoch, the training weights were stuck at a local minima and struggled to improve. After this point it is believed that he Adam optimizer adjusted its learning rate or some other hyperparameters that may have helped the model escape this local minima.

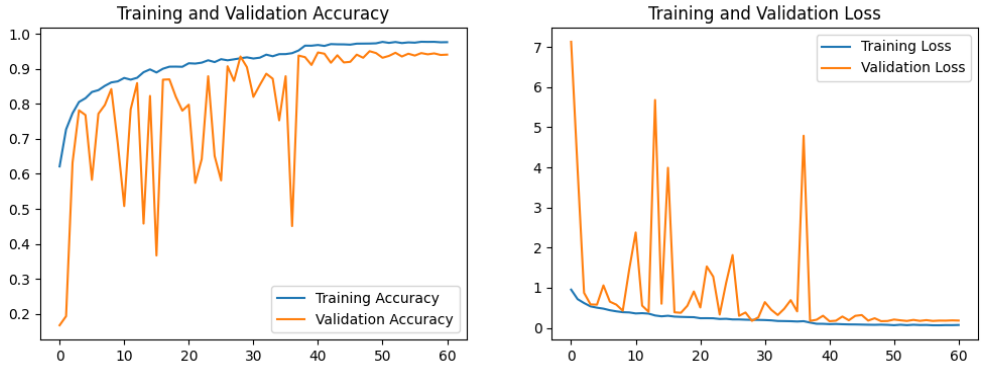


Figure 5.4: Training results of accuracy and loss per epoch.

Validation set performance

To demonstrate the per class performance of the validation data the confusion matrix is show below in Figure 5.5. The validation set consisted of 1117 particles with the following accuracies:

0 White Maize = 98.35%

1 Discolored Maize = 96.96%

2 Yellow Maize = 100.00%

3 Insect Damage = 93.13%

4 Heat Damage = 95.78%

5 Fursarium = 70.09%

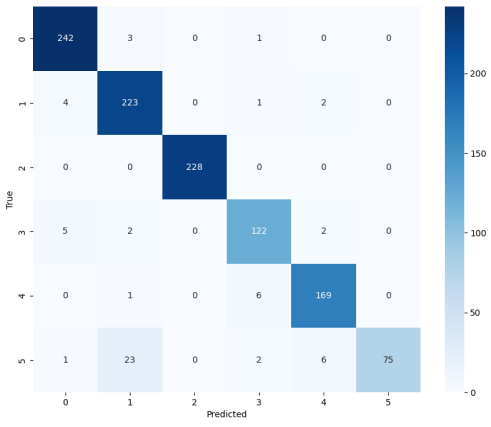


Figure 5.5: Confusion matrix

From these results it is clear that separating yellow maize from white maize is perfectly accurate. This is understandable given the complete color difference between yellow maize and the other types. Classifying insect and heat damage also performed reasonably well.

However, the biggest concern when looking at the above results is the poor performance of the fusarium category. Upon individual inspection to the images that were misclassified, it became clear that this performance value is a bit misleading. Many of the fusarium kernels had been predicted to be discolored by the model. Figure 5.6 shows a snippet of these misclassified kernels. From visual inspection is is clear that fusarium kernels shared striking visual similarities to the discolored kernels themselves. Figure 5.7 shows actual discolored kernels which can be used as comparison.

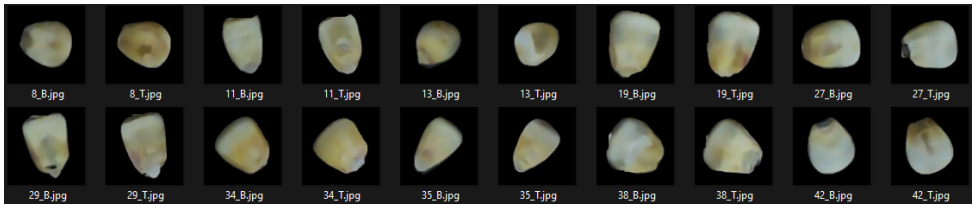


Figure 5.6: Example of fusarium kernels that were misclassified as discolored.

Figure 5.7: Example of actual labeled discolored kernels.

Comparing to related work

While directly comparing our results to that obtained in a different environment may not be be entirely justifiable, such a comparison may still provide some valuable insights. Table 5.1 below shows the validation accuracies compared to the initial benchmark accuracies shown in Chapter 2.

The detection of pure white maize performed well compared to that of the hyperspectural system. For yellow maize the model had a perfect prediction, which matches the performance of the multispectural system. Both insect damage and heat damage were also on par with the other systems. However in the case of fusarium the CNN-model under-performed with a 70% accuracy. It is believed that the hyperspectural images taken of fusarium were in the spectrum range where fungal fluorescence could be well detectede. It is known that infrared lighting can reveal fungal fluorescence that is not visible to the naked eye. In the future one can use some NoIR camera camera to perhaps achieve better results in this category.

Table 5.1: Comparison of our results with the accuracies of previous studies.

Available categories	Benchmark Accuracy %		
	Hyperspectral Camera	Multispectral Camera	CNN-Model (On validation set)
White Maize	88.3	-	98.35
Discolored White Maize	-	-	96.96
Yellow Maize	75.0	100	100
Insect Damaged	95.8	88.57	93.13
Heat Damaged	95.0	100	95.78
Fusarium	100	88.57	70.09

Overall these results should be taken with a grain of salt since the testing environments of the systems differed dramatically. Nevertheless what this does indicate is that one can achieve a relatively high degree of classification accuracy by using ordinary cameras and examining purely the outside shell of a kernel.

5.3. Run time

One of the main goals of the system is that is that it be designed to make the grading process quick and efficient. Given that we were using two cameras, the image capturing process took around 2 minutes. If there was a dual camera system in place this image capturing process would be done in practically an instant. After the images have been captured the program scripts runs and the entire process from start to finish takes around 90 seconds on the Raspberry Pi. Figure 5.8 shows each step and their relevant times taken. The image segmentation process is by far the longest process to run, thereafter the particle matching and concatenation process. Particle evaluation is a step that involves inputting our images through the model and the final cleaning stage refers to a step that removes all the temporary files created. With further optimisation in code, and faster SBCs one might speed the process up. However for now 90 seconds remains reasonably fast compared to the time it would take to manually grade the kernels. Additional note: (As of 3 November 2023, Raspberry Pi has recently release the new Raspberry 5 SBC which claims to run 2-3x fast than predecessor)

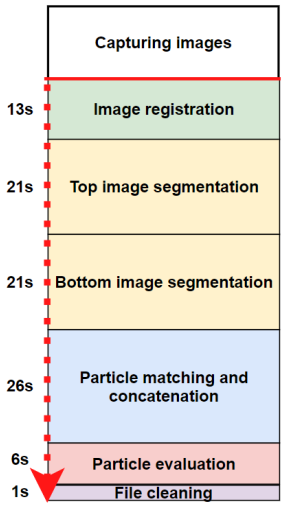


Figure 5.8: Timeline flow diagram

5.4. Testing on graded batches

As previously mentioned, the the SAGL have provided five 250 g samples of graded maize batches. These include a mixture of different types of kernels and in most cases, the proportions of each kernel type are provided as percentages. Any foreign materials and small shaving found in the batches have been removed and the original percentages given by the SAGL had to be adjusted accordingly. The conclusions are drawn from these new adjusted tables since it shows a more fair comparison of the models performance.

5.4.1. Batch 1

Batch 1 consists mainly of yellow kernels. The biggest challenge encountered is the presence of yellow maize kernels with insect damage. The model had not been trained to recognise insect damage on yellow kernels and thus unfortunately the dominant yellow hue seems to "overpower" the insect damaged features when the image was processed by the model. As a result almost every insect damage instance was seen as a plain yellow maize kernel. An example of this can been seen in Figure: 5.9. In addition heat damage performed relatively well, whereas some slight amounts of insect damaged kernels were seen as fursarium. Exact percentages of discolored and white maize were unknown with the final grade being unchanged. The full batch results are shown below in Table 5.2.

Table 5.2: Batch 1 results, (adjusted SAGL grade by removing foreign matter and small shavings

Batch 1	**SAGL Grade Adjusted(%)**	**Model Grade (%)**	**Error**
White Maize	*	0.77	*
Discolored Maize	*	0.46	*
Yellow Maize	93.02	96.62	3.6
Insect Damage	3.28	0.31	2.97
Heat Damage	1.27	1.23	0.04
Fusarium	0	0.62	0.62
Defective above 6.35mm	4.55	2.16	2.39
Other coloured	2.43	1.23	1.2
Combined deviations	**6.98**	**3.39**	**3.59**
Class and Grade	**YM1**	**YM1**	-

5.4.2. Batch 2

In comparison to the other batches, batch 2's classification is subpar. The batch consisted of mainly white maize with lots of smaller pieces and shavings. It is believed that some of the smaller pieces that were manually discarded had been initially classified as valid

Figure 5.9: Examples of a insect damaged kernel that was mistakenly misclassified as a yellow maize

maize kernels by the SAGL. This is likely that main reason for the low classification accuracy on this batch. The white maize category had an error of 5.14% and many of these kernels were incorrectly identified as being heat damaged or discolored. Upon reviewing the model's results, the inconsistencies were not apparent and finding kernels that were clearly misclassified in these two categories was challenging. This high error is likely due to again the same reason as mentioned above. Additionally, diplodia was present in the batch and unfortunately is one of the categories not been present in the training phase. The final combined deviation error was 5.14% and the grade given to the batch differed from that of the original. The full batch results are shown below in Table 5.3.

Table 5.3: Batch 2 results, (adjusted SAGL grade by removing foreign matter and small shavings

Batch 2	**SAGL Grade Adjusted(%)**	**Model Grade (%)**	**Error**
White Maize	95.44	90.30	5.14
Discolored Maize	0	1.34	1.34
Yellow Maize	0	0	0
Insect Damage	0	0.45	0.45
Heat Damage	0.21	3.73	3.52
Fusarium	4.14	4.18	0.04
Diplodia	0.21	-	-
Defective above 6.35mm	4.56	8.36	3.8
Other coloured	0	1.34	1.34
Combined deviations	**4.56**	**9.7**	**5.14**
Class and Grade	**WM1**	**WM2**	-

5.4.3. Batch 3

This batch was an interesting case. It contained many broken kernels and some oddly shaped circular particles. Figure 5.10 shows an example of these circular particles. It was difficult to know what to disregard and what to keep. It was decided to keep these circular shaped particles in the batch and remove all they broken pieces of kernels. Table 5.4 shows the full report of this batch after making the above decisions.

From the table we can see that white maize and discolored category performed reasonably well. However one of the big things that was immediately noticeable was the under representation of yellow maize kernels. Reviewing each of the classified classes, it was not possible to find a single clear yellow maize kernel that was misrepresented as another class. What this tells us is that the only logical explanation for this under representation is that some of the broken pieces that were removed had been initially labeled by the SAGL as non defective yellow maize.

In terms of the circular particles, the model classified most of them as heat damaged kernels. Looking now in hindsight at the report we know that these particles are most probably not maize kernels. From inspect they resemble something looking like seeds but it is unsure what exactly they are. The only category where it feels like the model was incorrect was in the insect damage category. Here in Figure 5.11 we can see the types of kernels that are actually defective free that were classified as insect damage. The final batch grade assigned by the SAGL had been given the grade of "OTHER". This comes primary from the high range of other colored kernels. If these seed particles were removed and perhaps more broken particles kept in the batch it is believed model performance would have looked much better.

Table 5.4: Batch 3 results, (adjusted SAGL grade by removing foreign matter and small shavings

Batch 3	**SAGL Grade Adjusted (%)**	**Model Grade (%)**	**Error**
White Maize	87.39	89.74	2.35
Discolored Maize	0.25	1.65	1.4
Yellow Maize	12.24	4.95	7.29
Insect Damage	*	2.01	*
Heat Damage	*	1.65	*
Fusarium	*	0	*
Defective above 6.35mm	0.12	3.66	3.54
Other coloured	12.49	6.6	5.89
Combined deviations	**12.61**	**10.26**	**2.35**
Class and Grade	**OTHER**	**WM3**	**-**

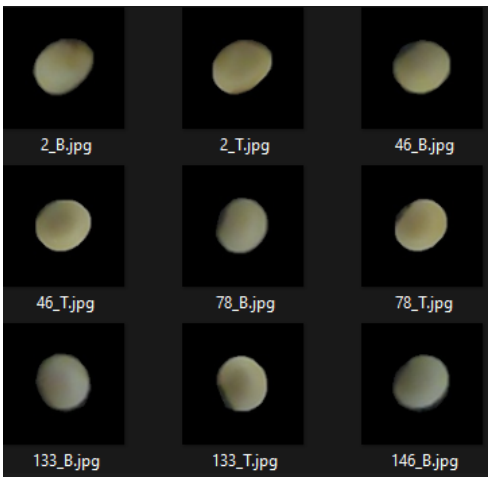


Figure 5.10: Circular particles classified as heat damage.

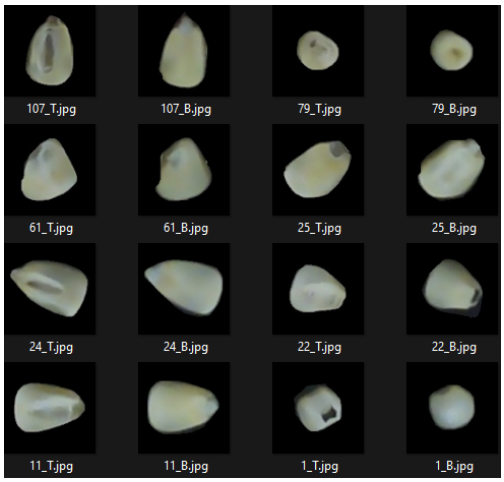


Figure 5.11: Kernels incorrectly classified as insect damage.

Figure 5.12: Examples of incorrectly classified maize in batch 3.

5.4.4. Batch 4

For this batch insect damage, yellow maize and fusarium classes performed well. Even though the network was primary trained on white maize with is defects, the network proved very robust as it still classified most classes correctly. The heat damage category however was a bit over classified. A few fusarium and white maize kernels where incorrectly assigned this class. Looking at the Figure 5.13 we can see that a white maize kernel is slighty darker in the stem (middle) than normal. The neural network seems to have been trained in seeing for dark patches in middle as heat damage. To the right in Figure 5.14 we can see the orange "overpowing" the dark fungal spots at the bottom of the kernel. It is not sure the kernel was originally a yellow maize kernel before it was exposed to fungal damage. If this is the case then this orage hue is unaccounted for and should be looked at in future training. The final combined deviation error was 2.47% an in this case the batch had been assigned the same grade as the original.

Figure 5.13: White maize kernel that had been classified as being heat damaged.

Figure 5.14: Fusarium kernel that had also been classified as heat damaged.

Figure 5.15: Example of kernels that have been misclassified in batch 4.

Table 5.5: Batch 4 results, (adjusted SAGL grade by removing foreign matter and small shavings

Batch 4	SAGL Grade Adjusted (%)	Model Grade (%)	Error
White Maize	*	0	*
Discolored Maize	*	0.54	*
Yellow Maize	83.98	86.44	2.46
Insect Damage	0.23	0.18	0.05
Heat Damage	3.18	9.22	6.04
Fusarium	4.55	3.62	0.93
Defective above 6.35mm	7.96	13.02	5.06
Other coloured	8.07	0.54	7.53
Combined deviations	**16.03**	**13.56**	**2.47**
Class and Grade	**YM3**	**YM3**	**-**

5.4.5. Batch 5

Batch 5 classification performed quite well as the results show in Table 5.6. In almost all of the classes the model got the proportions mostly correct. The final combined deviation error was 2.82%With a "defective above 6.35mm" error of 2% and "other colored" error of 0.97%. However even with this good performance the model didn't grade the batch the same as the original. The "defective above 6.35mm" category was the primary reason as the threshold from regulations is 6% where ours is slightly over.

Table 5.6: Batch 5 results, (adjusted SAGL grade by removing foreign matter and small shavings

Batch 5	SAGL Grade Adjusted (%)	Model Grade (%)	Error
White Maize	92.65	89.68	2.97
Discolored Maize	*	2.10	*
Yellow Maize	*	1.61	*
Insect Damage	0	0.97	0.97
Heat Damage	4.39	5.48	1.09
Fusarium	0	0.16	0.16
Sprouted	0.22	-	-
Defective above 6.35mm	4.61	6.61	2
Other coloured	2.74	3.71	0.97
Combined deviations	**7.35**	**10.32**	**2.82**
Class and Grade	**WM1**	**WM2**	**-**

5.5. Summary of test batches

The following section summarizes the performance of the model on the test batches per category group.

Table 5.7 describes the error rate among "defective kernels above 6.35mm", (category [#2]). What this is referring to is the insect damage, heat damage and fusarium that occurs. On average the error between all the batches amounts to 3.04%. Referring back to the regulations Table 2.1, when classifying a batch as either WM1 or WM2 based on category [#2], there is only a 6% margin of error. Thus this 3.04% obtained is in fact quite high in comparison to this margin.

Table 5.7: Batches performance of category [#2]

[#2] Defectives above 6.35mm			
Batch	**SAGL Grade (%)**	**Model Grade (%)**	**Error (%)**
[1]	4.55	2.16	2.39
[2]	4.56	8.36	3.8
[3]	0.12	3.66	3.54
[4]	7.96	13.02	5.06
[5]	4.61	6.61	2
Average Error:			**3.04**

For the class "other colored kernels" (category [#3]), the average error between the batches amounted to 3.39, as per Table 5.8. With reference again to the regulations Table 2.1. The difference here between grades is 3%. The error rate is thus quite high again.

Table 5.8: Batches performance of category [#3]

[#3] Other colored kernels			
Batch	**SAGL Grade (%)**	**Model Grade (%)**	**Error**
[1]	2.43	1.23	1.2
[2]	0	1.34	1.34
[3]	12.49	6.6	5.89
[4]	8.07	0.54	7.53
[5]	2.71	3.71	0.97
Average Error:			**3.386**

When looking at the "combined deviations" summary Table 5.9 (category [#4]), the total average error is 3.27%. There is more leeway from this regulation category as seen in Table 2.1. All in all this error value is not too bad if this was the only metric used to classify maize batches. However the challenges associated with the previous two categories remain.

Table 5.9: Batches performance of category [#4]

[#4] Combination of deviations			
Batch	**SAGL Grade (%)**	**Model Grade (%)**	**Error**
[**1**]	6.98	3.39	3.59
[**2**]	4.56	9.7	5.14
[**3**]	12.61	10.26	2.35
[**4**]	16.03	13.56	2.47
[**5**]	7.35	10.32	2.82
Average Error:			**3.27**

5.6. Chapter summary

It is important to note the reports given to us by the SAGL are not perfect and should not be seen always as the absolute ground truth. There is always somewhat human error induced of batches.. Only way really tell how good the model is to have an expert grader manually grade each kernel manually and then test this with the model. All in all the model however still performed reasonably well but when comparing the performance to the regulation margins it still suggests clear room for improvement.

In this Chapter the results of the model and the performance of the entire system had been explained. In the next Chapter some final thoughts will be given about the project.

Chapter 6

Conclusion

This chapter discuses the successes and shortcomings of the system, as well as potential future recommendations.

6.1. Summary

The primary objective of the project is to develop an accurate and cost-effective method for grading maize kernels using image processing techniques. While the system designed largely does accomplish this goal, it is in several areas not perfect.

Image capture

The overall box design performs well and creates a consistent environment for the images to be taken. Images through the glass also remain high quality. Some drawbacks of this system is that the glass platform tends to get dirty when used frequently. The size of platform is limited to 150 g samples of maize at a time. Glare on the corners of the image also pose an impact on the overall quality, however this can be easily mitigated.

Image processing and training

The image segmentation section of the project performs relatively well, missing only a few kernels out of every batch. These misses should however be addressed in further research. In terms of speed, the system runs reasonably fast compared to having to manually grade kernels by hand. The high validation accuracies in training process is promising and it proves is that it is very possible to achieve similar levels of accuracy using images from ordinary cameras.

Unseen batches

When testing on the unseen batches the system was met with some challenges. Primarily involving issues with shavings, misclassifying insect damage on yellow maize and tendency for the over-classification of heat damage. The error rate was higher than expected, after having high validation results. When compared to the regulation margins, only two out of

the five batches had been given the correct final grade. The others batches were off by one grade level. The performance of the model should however not be solely reflected by these results as the given samples are not the absolute ground truth and some of the batches went through somewhat inconsistent particle removals.

6.2. Future work

Image capture

- For capturing images, it would be a great addition to have some system that automatically sifts out smaller particles. Possibly even doing separate evaluation on those particles.
- To remove glare, one could use better anti glare materials. Such as anti reflective glass.
- Creating a solution that perfectly separates kernels by using something divider on the container platform.
- Two camera system can make image capturing process even faster.

Image processing and training

- To prevent kernels from being missed one can use more advance software techniques for image segmentation.
- To improve model performance one can expanding the training data set.

Other

- For a direct comparison having a multi-spectral or hyper-spectral camera with which the same kernels can be captured would be ideal.
- Additionally cameras with NoIR cut filters are available for the Raspberry Pi and it should be simple to test a camera that looks beyond the visible spectrum.

Final thoughts

The model could be fine-tuned slightly more. However, it is believed to be unlikely that this will drastically change the performance on unseen cases. The primary obstacle to better performance is with better image segmentation techniques and different experimental batches. What the current system does prove however, is that the principal of image grading is possible. There is significant promise, and with a bit of refinement, such as system could one day able to live up to industry standards.

Bibliography

[1] , "Grading Regulations for Maize, Government Notice No. R.473 ," Department of argiculture, Tech. Rep., 2009.

[2] The South African Grain Laboratory, "Quality Report 2020/2021 Season," South African Maize Crop, Tech. Rep., 2021.

[3] Tinashe Kapuya and Wandile Sihlobo, "South Africa's maize exports: A Strategic Export Market Analysis model approach," *Argicultural bussiness center*. [Online]. Available: https://agbiz.co.za/content/open/south-africa%E2%80%99s-maize-exports-a-strategic-export-market-analysis-model-approach

[4] Kate Sendin and Marena Manley and Paul J. Williams, "Classification of white maize defects with multispectral imaging," *Food Chemistry*, vol. 243, pp. 311–318, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0308814617316096

[5] Kate Sendin and Marena Manley and Federico Marini and Paul J. Williams, "Hierarchical classification pathway for white maize, defect and foreign material classification using spectral imaging," *Microchemical Journal*, vol. 162, p. 105824, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0026265X20337644

[6] Deepanshu Tyagi, "Introduction to SIFT (Scale Invariant Feature Transform)," *Medium*, 2019. [Online]. Available: https://medium.com/data-breach/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40

[7] Mathworks Inc, "What Is Image Registration?" *Mathworks*, 2023. [Online]. Available: https://www.mathworks.com/discovery/image-registration.html

[8] R. Fisher, S. Perkins, A. Walker and E. Wolfart, "Erosion," *Hypermedia Image Processing Reference*, 2003. [Online]. Available:

[9] scikit-image team, "Convex Hull," *scikit-image/Examples*, 2023. [Online]. Available: https://scikit-image.org/docs/stable/auto_examples/edges/plot_convex_hull.html#sphx-glr-download-auto-examples-edges-plot-convex-hull-py

[10] Ignacio Arganda-Carreras, "Classic Watershed," *ImageJ Wiki*, 2019. [Online]. Available:

[11] Richa Bhatia, "Why Convolutional Neural Networks Are The Go-To Models In Deep Learning," *Endless Origins*. [Online]. Available: https://analyticsindiamag.com/why-convolutional-neural-networks-are-the-go-to-models-in-deep-learning/

[12] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu, "Recent advances in convolutional neural network acceleration," *Neurocomputing*, vol. 323, pp. 37–51, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231218311007

[13] Mayank Mishra, "Convolutional Neural Networks, Explained," *Towards Data Science*, 2020. [Online]. Available: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

[14] Andriy Burkov, *The Hundred-page Machine Learning Book*. Andriy Burkov, 2019.

[15] Stanford University, "CS231n Convolutional Neural Networks for Visual Recognition," *CS231n: Deep Learning for Computer Vision*, 2023. [Online]. Available:

[16] Siddarth Das, "CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more…," *Medium*. [Online]. Available: https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

[17] Raspberry Pi Ltd, "Raspberry Pi Zero 2 W," Tech. Rep., october, 2021. [Online]. Available: https://datasheets.raspberrypi.com/rpizero2/raspberry-pi-zero-2-w-product-brief.pdf

[18] ——, "Raspberry Pi 4 Model B Datasheet," Tech. Rep., January, 2021. [Online]. Available: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf

[19] ——, "Raspberry Pi 3 Camera Datasheet," Raspberry Pi Ltd, Tech. Rep., Accessed: August 28, 2023. [Online]. Available: https://datasheets.raspberrypi.com/camera/camera-module-3-product-brief.pdf

[20] SPAZIO, "Technical Data Sheet, LED tapelight," *Printfriendly.com*. [Online]. Available: https://www.printfriendly.com/p/g/BhxsKG

[21] "8 - building decorative glass," in *Building Decorative Materials*, ser. Woodhead Publishing Series in Civil and Structural Engineering, Y. Li and S. Ren, Eds. Woodhead Publishing, 2011, pp. 139–168. [Online]. Available: https://www.sciencedirect.com/topics/chemistry/plate-glass

[22] -, "UNDERSTANDING GAMMA CORRECTION," *Cambridge in Color*. [Online]. Available: https://www.cambridgeincolour.com/tutorials/gamma-correction.htm

[23] ——, "Fiji," *ImageJ*. [Online]. Available: https://imagej.net/software/fiji/

[24] Shipira Saxena, "The Architecture of Lenet-5," *Analytics Vidhya*, 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5/

Appendix A

GA Compliance

GA 1 Problem solving

Chapters 1,2,3,4

In the project the problem identified was the possible human error induced in grading maize batches and the lengthy process of doing so. The problem was solved by developing a camera system that extracts maize kernels from images and thereafter trained a CNN model to classify maize kernels with a high degree of accuracy.

GA 2 Application of scientific and engineering knowledge

Chapters 2,3,4

With the knowledge from electrical and electronic modules, a single board computer (SBC) and camera system was built. Applying knowledge of image processing for image segmentation, and using coding and data engineering experience for creating a fast and efficient scripts and training a machine learning model.

GA 3 Engineering design

Chapters 2,3,4

Considering alternative designs. In the hardware section iterative principle of design was used where different types of systems were considered. The system is designed so that there is in a sense automation when images are captured. Each stage flows seamlessly into the next, as well as the the different softwares used working in conjunction with each other. Engineering techniques such as image segmentation and machine learning have also been implemented.

GA 4 Investigations, experiments and data analysis

Chapters 3,4,5,6

The model has been trained and its performance is directly compared to that of other technologies. The loss curves depicted the progression of the training. There are many explanations about experiments and investigations done throughout project on types of materials and designs and software choices. Furthermore a test case scenario is created where the system is tested in unseen batches of maize. These results are tabulated, summarized and explained in detail.

GA 5 Engineering methods, skills and tools, including IT

Chapters 2,3,4

Some software and packages used are: ImageJ Fiji, Python, TensorFlow, OpenCV, Pi-Camera, etc. Engineering methods such as machine learning and image processing have been used in development in the system.

GA 6 Professional and technical communication

Chapters 1,2,3,4,5,6

The project includes a written report and an oral presentation. These demonstrate competence to communicate effectively, both orally and in writing.

GA 8 Individual work

Chapters 1,2,3,4,5,6

The student will take primary responsibility for successful completion of all aspects of the project.

GA 9 Independent learning ability

Chapters 1,2,3,4,5,6

For successful completion of the project, the student is required to acquire knowledge independently (from the literature or the internet, for example) and without the context of this required knowledge being fully specified in the project definition.

Appendix B

Additional images

Figure B.1: Camera holder

Figure B.2: Camera setup with Raspberry Pi

Figure B.3: Glass container

Figure B.4: Glass container with kernels being slid into box

Figure B.5: Defect kernels as provided by the SAGL.

Figure B.6: Sliding sheets

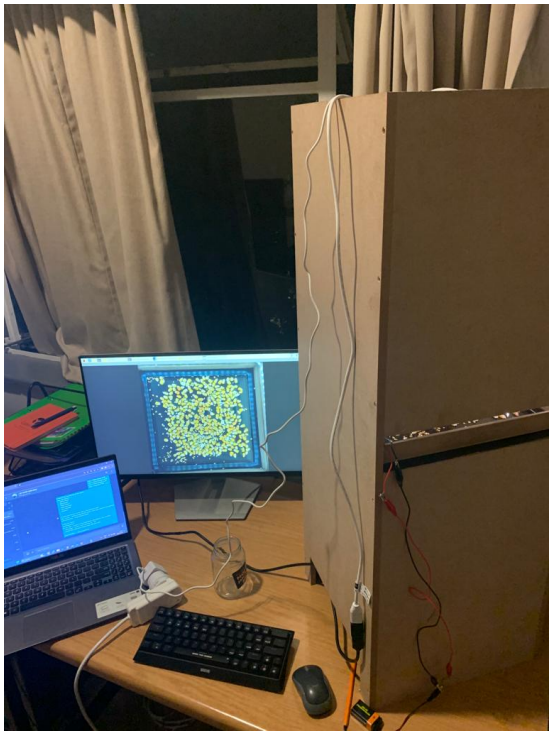


Figure B.7: Overall setup

Figure B.8: Top view setup

Appendix C

Original Batch results (as per SAGL)

Table C.1: Batch 1 Results

Batch 1	SAGL Grade (%)	Model Grade (%)	Error
White Maize	*	0.77	*
Discolored Maize	*	0.46	*
Yellow Maize	88.00	96.62	8.62
Insect Damage	3.10	0.31	2.79
Heat Damage	1.20	1.23	0.03
Fusarium	0	0.62	0.62
Defective above 6.35mm	4.30	2.16	2.14
Defective below 6.35mm	5.10	-	-
Defective kernels (above and below 6.35mm)	9.4	2.16	7.24
Other coloured	2.30	1.23	1.07
Foreign Matter	0.3	-	-
Combined deviations	**12**	**3.39**	**8.61**
Class and Grade	**YM2**	**YM1**	-

Table C.2: Batch 2 Results

Batch 2	**SAGL Grade (%)**	**Model Grade (%)**	**Error**
White Maize	90.0	90.30	0.3
Discolored Maize	0	1.34	1.34
Yellow Maize	0	0	0
Insect Damage	0	0.45	0.45
Heat Damage	0.20	3.73	3.53
Fusarium	3.90	4.18	0.28
Diplodia	0.2	-	-
Defective above 6.35mm	4.3	8.36	4.06
Defective below 6.35mm	5.6	-	-
Defective (above and below 6.35mm)	9.9	8.36	1.54
Other coloured	0	1.34	1.34
Foreign Matter	0.1	-	-
Combined deviations	**10.0**	**9.7**	**0.3**
Class and Grade	**WM2**	**WM2**	-

Table C.3: Batch 3 Results

Batch 3	**SAGL Grade (%)**	**Model Grade (%)**	**Error**
White Maize	70.7	89.74	19.04
Discolored Maize	0.2	1.65	1.45
Yellow Maize	9.9	4.95	4.95
Insect Damage	*	2.01	*
Heat Damage	*	1.65	*
Fusarium	*	0	*
Defective above 6.35mm	0.1	3.66	3.56
Defective below 6.35mm	16.2	-	-
Defective kernels (above and below 6.35mm)	16.3	3.66	12.64
Other coloured	10.1	6.6	3.5
Foreign Matter	2.9	-	-
Combined deviations	**29.3**	**10.26**	**19.04**
Class and Grade	**OTHER**	**WM3**	**-**

Table C.4: Batch 4 Results

Batch 4	SAGL Grade (%)	Model Grade (%)	Error
White Maize	*	0	*
Discolored Maize	*	0.54	*
Yellow Maize	73.9	86.44	12.54
Insect Damage	0.2	0.18	0.02
Heat Damage	2.8	9.22	6.42
Fusarium	4.0	3.62	0.38
Defective above 6.35mm	7	13.02	6.02
Defective below 6.35mm	11.6	-	-
Defective kernels (above and below 6.35mm)	18.6	13.02	5.58
Other coloured	7.1	0.54	6.56
Foreign Matter	0.4	-	-
Combined deviations	**26.1**	**13.56**	**12.54**
Class and Grade	**OTHER**	**YM3**	**-**

Batch 5	SAGL Grade (%)	Model Grade (%)	Error
White Maize	84.4	89.68	5.28
Discolored Maize	*	2.10	*
Yellow Maize	*	1.61	*
Insect Damage	0	0.97	0.97
Heat Damage	4.0	5.48	1.48
Fusarium	0	0.16	0.16
Sprouted	0.2	-	-
Defective above 6.35mm	4.2	6.61	2.41
Defective below 6.35mm	8.0	-	-
Defective kernels (above and below 6.35mm)	12.2	6.61	5.59
Other coloured	2.5	3.71	1.21
Foreign Matter	0.9	-	-
Combined deviations	**15.6**	**10.32**	**5.28**
Class and Grade	**OTHER**	**WM2**	**-**

Table C.5: Batch 5 Results

Appendix D

Extra

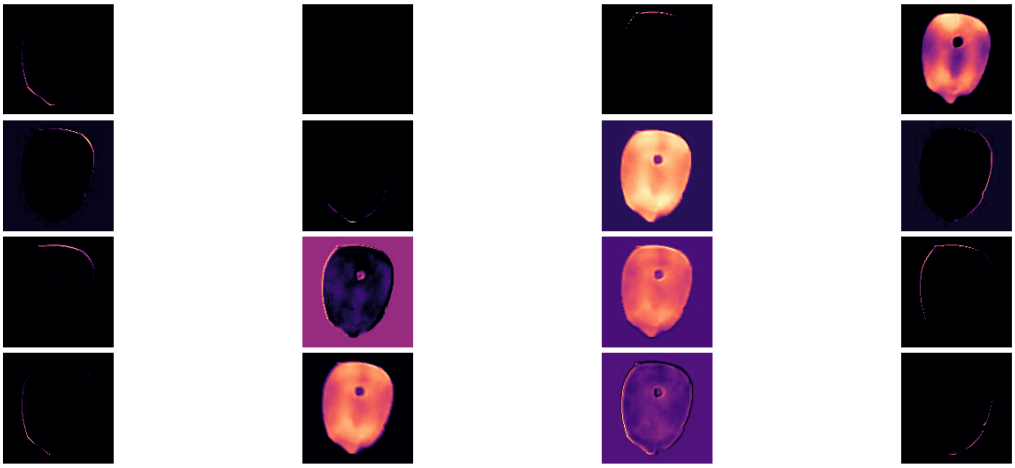


Figure D.1: Feature detection of first layer visualization

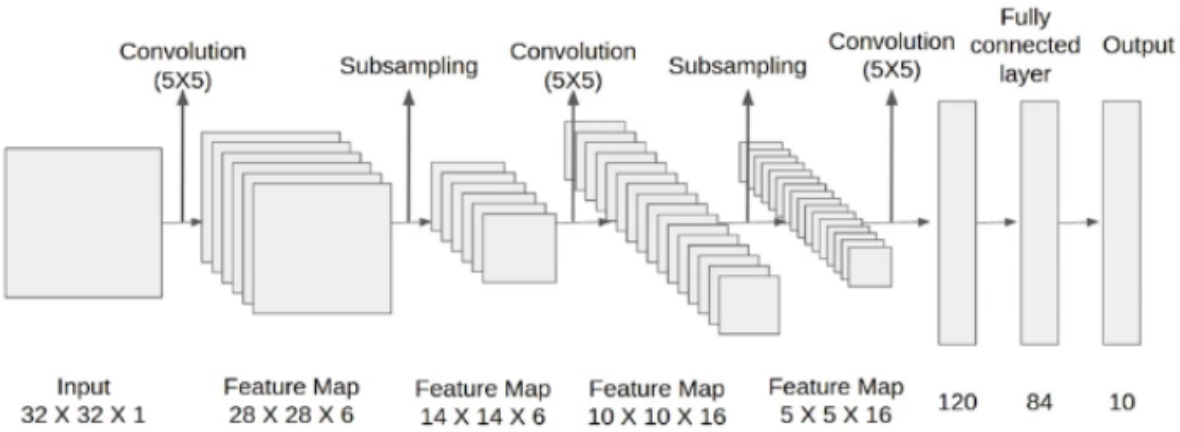


Figure D.2: LeNet-5 architecture. Image reproduced from: [24]

Table D.1: Software used and key functions at each stage of processing

Processing Stage	Frameworks	Key Functions
Image Registration	OpenCV numpy	SIFT_create() BFMatcher() detectAndCompute() knnMatch() findHomography() warpPerspective()
Image Segmentation	ImageJ Fiji Java plugin	IJ.run("HSB Stack") ImagePlus("Hue_Mask") ImagePlus("Saturation_Mask") ImagePlus("Brightness_Mask") ImageCalculator.run("AND create")
	ImageJ Fiji Java plugin	IJ.run("8-bit") IJ.run("Erode")
	ImageJ Fiji Java plugin	IJ.run("Watershed")
	OpenCV numpy	cv2.convexHull() cv2.findContours() cv2.drawContours()
Particle Matching	OpenCV numpy	get_centers_and_contours() pad_image_to_size() np.linalg.norm() cv2.boundingRect() cv2.bitwise_and()
Image Concatenation	numpy	np.concatenate()