Abstract

Digitization of archaeology is in great demand. Since 2009, a team of researchers and students led by Dr. Cobb has been investigating the area around Vedi, Armenia, aiming at understanding human life and mobility in the ancient landscapes of the Near East. A large volume of sherds was excavated and documented with photography. Inspired by the recent advancement in computer vision and machine learning, this project attempts to explore various deep learning models to classify and compare those sherds unearthed. To accomplish the objective, OpenCV will be utilized to do data processing, including scaling, color correction and cropping. Additionally, Pytorch is used to develop deep learning models. This report discusses the techniques applied and the current progress in the data processing. In the near future, various Convolutional Neural Network (CNN) and Vision Transformer (ViT) based models will be tested with the dataset cleansed by the aforementioned pipeline. It is hoped that insights gained from the project can help archaeologists manage the massive quantity of ancient artifacts in the future.

Acknowledgements

We would like to express our sincere gratitude to our supervisor, Dr. Wong, Kenneth K.Y. for the guidance and suggestions provided throughout the whole project. Without his clear guidance, dedicated involvement and support throughout the process, this project would not have been possible.

My team would also like to thank Dr. Cobb, Peter J. and his team for sharing necessary resources and archaeological expertise regarding their fieldwork in Armenia for this project.

Lastly, my team would like to thank Miss Mable Choi from the Centre for Applied English Studies for offering help in report writing and presentation.

Member Contribution

Table 0.1 illustrates how work is distributed in the project team. This interim report focuses on the results we achieved at the former stage of this project.
<table>
<thead>
<tr>
<th><strong>Data Processing</strong></th>
<th>Scaling and Resizing</th>
<th>Sam</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color Correction: Color Calibration</td>
<td></td>
<td>Fiona</td>
</tr>
<tr>
<td>Color Correction: White balance</td>
<td></td>
<td>Ricky</td>
</tr>
<tr>
<td>Masking and Cropping</td>
<td></td>
<td>Ricky</td>
</tr>
<tr>
<td><strong>Machine Learning</strong></td>
<td>Develop the initial model</td>
<td>Sam and Fiona</td>
</tr>
</tbody>
</table>
# Contents

Abstract ....................................................... i

Acknowledgements ............................................. i

Member Contribution .......................................... i

List of Figures .................................................. v

List of Tables .................................................... v

Abbreviations ..................................................... vii

Notations and Symbols ......................................... vii

1 Introduction ..................................................... 1
   1.1 Background: Archaeology and Technology ....................... 1
   1.2 Background: Ceramics sherds from Armenia ....................... 1
   1.3 Project objective ............................................ 2
   1.4 Project contribution .......................................... 3
   1.5 Outline of the report ......................................... 3

2 Methodology ..................................................... 4
   2.1 Overview .................................................... 4
   2.2 Dataset .................................................... 4
   2.3 Image Processing ............................................ 5
      2.3.1 Color Correction .......................................... 5
      2.3.2 Scale Calibration ........................................ 9
      2.3.3 Masking .................................................. 13
      2.3.4 Cropping ................................................ 14
   2.4 Convolutional Neural Network .................................. 16
   2.5 Vision Transformer .......................................... 17
   2.6 Summary ................................................... 20

3 Results and Discussion ....................................... 21
   3.1 Overview ................................................... 21
   3.2 Current Status: Image Processing ................................ 21
3.2.1 Color Correction ........................................... 21
3.2.2 Geometric Calibration Results and Interpretation ............. 26
3.2.3 Masking by thresholding ................................... 30
3.2.4 Cropping .................................................... 31
3.3 Current Status: Machine Learning ................................ 33
  3.3.1 Model for Transfer learning ................................ 34
  3.3.2 Loss Function ............................................. 34
  3.3.3 Optimizers ............................................... 36
  3.3.4 Preliminary Model Evaluation .............................. 37
3.4 Limitations and Difficulties .................................... 39
  3.4.1 Image Processing .......................................... 40
  3.4.2 Pre-Processed Dataset .................................... 40
3.5 Future Plan .................................................... 41
3.6 Schedule and Milestones ...................................... 43

4 Conclusion ....................................................... 44

References ......................................................... 45
List of Figures

2.1 Two types of RAW images in the dataset ........................................ 4
   (a) Taken with archaeology photography scale ............................... 4
   (b) Taken with Macbeth ColorChecker and a scale ............................ 4
2.2 Image processing pipeline ............................................................. 5
2.3 Planar calibration setting of finding the projection matrix from real world to image plane .......................................................... 10
2.4 Colorspaces ................................................................................. 13
2.5 Collision detection: Only 4 cases that do not intersect ..................... 14
2.6 CNN Architecture .......................................................................... 16
2.7 ViT workflow .................................................................................. 19
3.1 Color checker detection in the two types of raw images .................... 21
3.2 Outermost corners (in red) in a 24 color calibration card .................. 26
3.3 Before (left) and after (right) calibration in one sample of 4-color calibration card ............................................................... 28
3.4 Before (left) and after (right) applying RANSAC in SIFT method ....... 30
3.5 Simple thresholding for color patch detection ................................... 30
3.6 Masking with global thresholding .................................................... 31
3.7 Masking for object detection ............................................................. 32
3.8 Objects and black squares detected ................................................... 32
3.9 Sherd and its mask in an image ......................................................... 33
3.10 Sample image data generated ......................................................... 33
3.11 A simple residual building block ..................................................... 34
3.12 Graphs of loss and accuracy in preliminary training ....................... 37
3.13 Graphs of loss and accuracy after data augmentation, using SGD ....... 38
3.14 Graphs of loss and accuracy using Adam optimizer ......................... 39
3.15 Distribution of data for each label .................................................... 41

List of Tables

0.1 Contribution of Members ............................................................... ii
3.1 Color correction result using least squares ...................................... 22
3.2 Comparison between using least squares and constrained least squares method 23
3.3 Final color correction results using intensity independent calibration method 24
3.4 Loss and Accuracy values of different models ........................................ 39
3.5 Project Schedule ................................................................................. 43
Abbreviations

AABB Axis-aligned Bounding Box
ANN Artificial Neural Network
CCM Color Correction Matrix
CNN Convolutional Neural Network
Lab Lightness, Red/Green, Blue/Yellow
NLP Natural Language Processing
ViT Vision Transformer

Notations and Symbols

\( a(.) \) activation function in neural network
\( X \) 2D matrix representation of color image
\( K \) key matrix in attention mechanism
\( Q \) query matrix in attention mechanism
\( V \) value matrix in attention mechanism
\( d_k \) normalization factor dependent on the key vector dimension
\( W \) linear transformation
\( F \) F score
\( M_{CCM} \) color correction matrix
1 Introduction

1.1 Background: Archaeology and Technology

Humans are always wondering about their past: how do people live the life in the past, why and how are human cultures are formed, and more. For many years, archaeologists try to find the answers. Their work involves surveying, excavation, and eventually analysis of data collected, hoping to reappear and understand the ancient and recent human past through material remains.

Various kinds of archaeological remains excavated by archaeologists around the world. With the rise of digital cameras, photography becomes a major means of documentation vital to both the historical and archaeological record. However, advancements in computer vision and deep learning make computers capable of diving one step further, not only in better storing information but also improved analysis of findings in archaeology.

Data is the foundation in today’s digital world. When it comes to archaeological data, there are so many exciting possibilities and potentials to be discovered. By utilizing technology, we believe data can transcend various industries, culture and even time constraints, yielding important insights to the aforementioned questions in archaeology that human has sought for a long time.

1.2 Background: Ceramics sherds from Armenia

Since 2009, a team of researchers and students led by Dr. Peter Cobb have been investigating the area around Vedi, Armenia, aiming at understanding human life and mobility in the ancient landscapes of the Near East [1]. It is located at the southeast edge of the vast and fertile Ararat Plain. This region has served as a contact point for Turkey, Iran (Persia) and Russia over the past few centuries. It has always been a significant hub for transportation, including on the Silk Road. Today, Armenia is one of the nations involved in the Belt and Road initiative [2].

As a result, a large volume of ceramic sherds unearthed from this site cover both the Bronze Age and Medieval periods. So far, the number of sherds reached around 20,000 and will grow along with new excavations in the next couple of years. It opens the possibility to compare diachronic changes and continuity in local resource exploitation.

After collecting these sherds and photographing them, a series of image processing work
needs to be done manually. The existing workflow consists of calibrating the colors and scaling sizes of sherds and cropping an area that best represents the material. Finally, annotation and inspection are carried out to group similar sherds into the same categories. All of these take a lot of time and effort and mistakes could be made.

This is where machine learning and computer vision come into play. It is hoped that these technologies can automatically standardize the image data and classify the ceramic sherds based on their textures and colors. This could potentially improve the efficiency and accuracy of post-excavation analysis in this project. If all goes well with the experiment and application, it is anticipated that archaeologists will be able to handle the tremendous number of ancient objects excavated in other sites accurately and efficiently in the future.

1.3 Project objective

This project attempts to explore various deep-learning models to classify and compare sherds excavated at the Vedi Fortress Archaeological Site, in Armenia. It is expected the final product of this project, by taking in sherd images, can group sherds into the identified categories. The project can be divided into two phases: data processing and machine learning model development for image classification.

The first stage consists of color correction, image scaling and image cropping. There are three major issues with the current image data. First, different light sources and sensors may lead to inconsistent colorways across image data. Thus, the program should be able to recognize the color card in each image and calibrate them in order to perceive the colors of objects correctly regardless of differences in illumination. Second, grains on sherds are important features for recognizing the texture of sherds. However, the images may be distorted due to the distance and angle between the camera and the sherds. Lastly, the shapes of sherds are irrelevant information that may confuse the model. To minimize the impact, our program should be able to resize the sherds and crop images to a standard size in order to make all images comparable. The goal at this phase is to develop a fully automated solution making image data less environment- and device-dependent before feeding to the models.

The second stage involves implementing computer vision architectures. Since there is no guarantee in machine learning, the experiment will be carried out on various CNN and ViT based models, exploring the optimal possible parameters. A comprehensive analysis of these experimental results can provide some insights into finding the success and/or
failure factors of models developed. Hopefully, the final deliverable will be able to classify sherds into the identified groups according to their image data and other attributes with satisfactory performance.

With the aid of this pipeline, the team can handle fragments unearthed and also those yet to be discovered from Armenia effectively. Furthermore, it is hoped that the project can open a new door in archaeology, allowing archaeologists to handle the large amount of ancient objects excavated like pottery vessels and sherds accurately and efficiently in the future.

1.4 Project contribution

This project is anticipated to revolutionize sherd image classification in archaeology. The computer will be trained to group sherds both discovered and yet to be discovered automatically and effectively. By utilizing computer vision and machine learning, the delivery can accelerate the sherd processing time for the team led by Dr. Cobb in Armenia.

It is hoped that the project can yield insights in archaeology, allowing archaeologists to organize and study the many ancient objects excavated with the aid of machine learning in the future.

1.5 Outline of the report

This report is organized into four chapters. Chapter 1 provides the background and rationale. It offers an overview and significance of our work toward the findings in Vedi, Armenia. It also sets out the goals of the study.

Chapter 2 describes the methodology involved in this project. Techniques and tools that are applied for data processing will be discussed. Additionally, it covers two types of model architectures, CNN and ViT.

Chapter 3 then presents the results achieved in the project, including the current status and problems encountered, followed by the future plan and project schedule.

Chapter 4 rounds off this report by providing a summary. This chapter restates the motivation and progress of this project.
2 Methodology

2.1 Overview

This chapter details the workflow of the proposed solution of this project. It presents important theories, technology implementation and types of evaluation that are applied to achieve the aforementioned objectives. It will also describe the nature of our image dataset. The proposed product of this project is divided into two major phases, namely the data processing phase and the deep learning model development phase.

The first half of this chapter focuses on algorithms and tools that are utilized for image scaling, color correction and cropping. The second half covers two types of model architectures, CNN and ViT, that will be used for development.

2.2 Dataset

Photography is an important documentation method for archaeological findings. Sherds excavated by Dr. Peter Cobb and his team at Vedi Fortress in Armenia are documented as digital photos. Each sherd is recorded from the front and the back sides. All photos are stored in CR2 or CR3 file format. CR2 and CR3 files store uncompressed RAW image data captured from the camera directly and the application of this project will take them as input.

(a) Taken with archaeology photography scale
(b) Taken with Macbeth ColorChecker and a scale

Figure 2.1: Two types of RAW images in the dataset

While most of the sherds were taken with a photography scale that contains four standard colours, a 24-patch Macbeth ColorChecker has been used for a substantial amount of data as well and especially for labeled data. Figure 2.1(a) and Figure 2.1(b) illustrate the two
kinds of data respectively. The photography scale and ColorChecker serve as references for color calibration and scaling.

Currently, our dataset consists of about 40,000 in unprocessed raw images. 2,000 of them are labeled with the classes belonging while the rest are not. Labels are named based on sherd color and texture. Since image data quality is crucial in machine learning, the images need to be processed before proceeding to the next stage.

2.3 Image Processing

Image processing is the area that deals with the analysis, enhancement and manipulation of digital images for feature extraction, recognition, and classification purposes [3]. Image processing for this project attempts to minimize impacts caused by the external environment mentioned previously.

![Image processing pipeline](image)

**Figure 2.2: Image processing pipeline**

The OpenCV library will be used for image processing. OpenCV is a powerful open-source computer vision library that provides optimized algorithms to accomplish various computer vision tasks, including but not limited to object identification, face recognition and model extraction. The application will utilize OpenCV to develop a pipeline (Figure 2.2) to address the following tasks: color correction, scaling images and cropping sherds out, so that the raw images are transformed into a standardized form for training.

The remaining section of this chapter will elaborate on the different techniques and theories used.

2.3.1 Color Correction

The raw data is taken under different lighting conditions and it is crucial to standardize the color before feeding into the machine learning model. The methodology of color calibration
is explained in this section, which involves adjusting the color response of input device to a known calibration target [4, 5].

The first step of color calibration is to find the position of the color chart in the image. The contours of the image are detected to obtain any bounding rectangle coordinates. The bounding rectangles are then filtered using maximum and minimum threshold area and the limit HSV values of each color. However, it was found that the black patch on the color chart could not be identified on every image. To guarantee the accurate detection of the black patch, automatic white balancing is first done before detecting the contours. The automatic white balancing algorithm first calculates the average color of an image then shifts the color by scaling the chroma distance derived from the amount of luminance and the average color in LAB color space [6]. After that, the color patches on the input image color chart will be identified.

Color correction is essentially the process of mapping device RGBs to corresponding device independent XYZs [7]. A $3 \times 3$ linear transformation Color Correction Matrix (CCM) allows linear mapping from RGB to XYZ. Linear color correction can be expressed mathematically as:

$$ C = I \times M_{CCM} $$

where $I$ is $N \times 3$ set of raw linear detected input RGB responses for N pixels:

$$ I = \begin{bmatrix} R_1 & G_1 & B_1 \\ R_2 & G_2 & B_2 \\ \vdots \\ R_N & G_N & B_N \end{bmatrix} $$

and $C$ is $N \times 3$ matrix of reference XYZ triplets:

$$ C = \begin{bmatrix} X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \\ \vdots \\ X_N & Y_N & Z_N \end{bmatrix} $$

Therefore, color correction involves finding CCM, which is the mapping from RGB to XYZ. The most ideal mapping would be to minimize CCM such that there is minimum deviation between the color patches and target spectral colors:

$$ Min_M \| M \times I - C \| $$

6
Three different methods of calculating the color correction mapping, least square, constrained least square, and intensity independent RGB-to-XYZ color calibration, are experimented in this project to calibrate the images, and their methodologies are explained below.

2.3.1.1 Least Square Method  The least square method can be used to obtain the $3 \times 3$ Color Correction Matrix $M_{CCM}$, utilizing the linear RGB distance function [8] to minimize the sum of squared errors, which can be expressed as:

$$M_{CCM} = (I^T I)^{-1} I^T C$$

Color transformation is done by multiplying $M_{CCM}$ with the color matrix of the original image, which transforms the data into a linear absolute color space. Assessment is needed to see whether the conversion result is close to the reference value. The measurement for evaluation is the loss function. The optimal CCM is found after minimizing the loss. Loss function is defined as the weighted square sum of the color difference between the standard reference data and detected image data. To calculate the color difference, conversion from linear absolute color space to Lightness, Red/Green, Blue/Yellow (Lab) color space is required, since the common standard for color difference (CIEDE2000) is based on the Lab color space [9].

To minimize the loss and find a better mapping, the Nelder-Mead method can be used for nonlinear optimization. The Nelder-Mead method helps to find the $M_{CCM}$ that produces the minimum loss [10]. However, true global optimum cannot be guaranteed but Nelder-Mead works reasonably well for problems that do not have many local minima.

2.3.1.2 Constrained Least-Squares Regression  Traditional least square method makes no premise about which colors will be more accurately or less accurately mapped. If there is incomplete calibration data, some important colors may be calibrated with high colorimetric error depending on the data. This variable performance is not desirable [11]. To ensure the correct mapping of a certain color, constrained least-squares regression can be used.

The method imposes the constraint that the CCM should map a basis surface with no error when solving the least squares problem, minimizing the difference sum of all squares between the mapped RGB data and corresponding XYZ reference values [11]. It is particularly useful when the color calibration data is incomplete. And generally, there is a higher
importance of mapping white with higher accuracy for color reproduction. Therefore, in this project, a constraint of finding the mapping which maps white without error will be added.

### 2.3.1.3 Intensity Independent RGB-to-XYZ color calibration

The above methods require a condition that RGBs measured in a real scene has the same shading profile as reference XYZs. However, the reference XYZs given by the color checker manufacturer were obtained when there was uniform lighting on the color chart, which is hard to accomplish for the archaeology photography team. In order to address the problem of non-uniform lighting variation leading to incorrect calibration results, the method of intensity independent RGB-to-XYZ color calibration can be adopted.

The method determines the $3 \times 3$ transformation matrix by minimizing the angular error between the mapped RGB values and the reference XYZs, which is shading independent. It can be represented as following, where $\{\vec{a}\}_{i=1}^{N}$ is the set of camera RGBs, and $\{\vec{b}\}_{i=1}^{N}$ are their correspondent XYZs:

$$E(M) = \sum_{i=1}^{N} \cos^{-1} \left( \frac{M\vec{a}_i \cdot \vec{b}_i}{|M\vec{a}_i| |\vec{b}_i|} \right)$$

To find the non-global minimum angular error, a standard search algorithm, Nelder-Mead nonlinear optimization, is used to solve for $M$. Experiment has shown that the color calibration performance using this intensity independent method is twice as good as using the least squares regression [12].
2.3.2 Scale Calibration

For a projective camera viewing a planar scene, the \(i^{th}\) observation point can be formulated as:

\[
\begin{pmatrix}
  s_i u_i \\
  s_i v_i \\
  s_i
\end{pmatrix} =
\begin{pmatrix}
  p_{00} & p_{01} & p_{02} \\
  p_{10} & p_{11} & p_{12} \\
  p_{20} & p_{21} & p_{22}
\end{pmatrix}
\begin{pmatrix}
  X_i \\
  Y_i \\
  1
\end{pmatrix}
\]

where \((X_i, Y_i)\) denotes the real world 2D coordinates of a point on the object to be photographed, it is assumed the object lies flat on a surface with uniform \(z\)-coordinate. The projected coordinates of the corresponding point on the image are \((u_i, v_i)\).

\(P_p = \begin{pmatrix} p_{00} & p_{01} & p_{02} \\ p_{10} & p_{11} & p_{12} \\ p_{20} & p_{21} & p_{22} \end{pmatrix}\) is the homography matrix which entails \(P_p \simeq K[R \ T]\) where \(R\) and \(T\) are the rotation and translation matrices respectively [10].

From the matrix, one can deduce

\[
s_i = p_{20}X_i + p_{21}Y_i + p_{22}
\]

as the regularization constant.

Each observation can be represented as

\[
u_i = \frac{s_i u_i}{s_i} = \frac{p_{00}X_i + p_{01}Y_i + p_{02}}{p_{20}X_i + p_{21}Y_i + p_{22}}
\]

and similarly

\[
u_i = \frac{s_i v_i}{s_i} = \frac{p_{10}X_i + p_{11}Y_i + p_{12}}{p_{20}X_i + p_{21}Y_i + p_{22}}
\]
Figure 2.3: Planar calibration setting of finding the projection matrix from real world to image plane

With some algebraic manipulation, one can obtain:

\[
\begin{align*}
    u_i &= X_i p_{00} + Y_i p_{01} + p_{02} - u_i X_i p_{20} - u_i Y_i p_{21} \\
    v_i &= X_i p_{10} + Y_i p_{11} + p_{12} - v_i X_i p_{20} - v_i Y_i p_{21}
\end{align*}
\]

As the overall scale in the homography matrix matters not in that multiplying \( \mathbf{P}_p \) by a constant factor does not alter the transformation being described, one can set \( p_{22} \) to 1 so that 8 degrees of freedom can be enforced. This results in a total of 8 unknowns to be solved for in the projection matrix. Since a system of linear equations in \( n \) unknown variables requires at least \( n \) linearly independent equations, and each observation point of pose mapping provides 2 equations, at least 4 points will be needed to carry out projective calibration.

To illustrate, suppose there are four points \( O, A, B \) and \( C \) of known relative distances from one another within the real world space (Fig. 2.3) that is of a regular square shape. The occurrence of an arbitrary quadrilateral within the image plane is common in photographs due to the different possible angles of the camera plane causing a change in orientation of the image plane [13]. On the right hand side of Fig. 2.3 is the projected form of the real world plane object onto the image plane.

To remediate the change of perspective and recover a desired regular shape in the output image, one can set up a \( 3 \times 3 \) camera projection matrix \( \mathbf{P}_p \).
As shown below using the example of Fig 2.3:

\[
\begin{pmatrix}
    u_o \\
    v_o \\
    u_a \\
    v_a \\
    u_b \\
    v_b \\
    u_c \\
    v_c
\end{pmatrix} = 
\begin{pmatrix}
    X_O & Y_O & 1 & 0 & 0 & 0 & -u_oX_O & -u_oY_O \\
    0 & 0 & 0 & X_O & Y_O & 1 & -v_oX_O & -v_oY_O \\
    X_A & Y_A & 1 & 0 & 0 & 0 & -u_aX_A & -u_aY_A \\
    0 & 0 & 0 & X_A & Y_A & 1 & -v_aX_A & -v_aY_A \\
    X_B & Y_B & 1 & 0 & 0 & 0 & -u_bX_B & -u_bY_B \\
    0 & 0 & 0 & X_B & Y_B & 1 & -v_bX_B & -v_bY_B \\
    X_C & Y_C & 1 & 0 & 0 & 0 & -u_cX_C & -u_cY_C \\
    0 & 0 & 0 & X_C & Y_C & 1 & -v_cX_C & -v_cY_C
\end{pmatrix}
\begin{pmatrix}
p_{00} \\
p_{01} \\
p_{02} \\
p_{10} \\
p_{11} \\
p_{12} \\
p_{20} \\
p_{21}
\end{pmatrix}
\]

By computing the inverse of the \(8 \times 8\) matrix, the parameters in the projection matrix \(P_p\) can be derived.

In the case of more than four points used, it is not possible to obtain a unique solution by solving exactly for the system of over \(8\) linear equations in \(8\) unknowns, where the linear equations are linearly independent of one another.

An estimation method would be used to compute the camera projection matrix.

Denote the scenario as \(Ap = b\) where \(A\) stands for the \(2n \times 8\) matrix with \(n\) being the number of points used such that

\[
A = \begin{pmatrix}
    X_1 & Y_1 & 1 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 \\
    0 & 0 & 0 & X_1 & Y_1 & 1 & -v_1X_1 & -v_1Y_1 \\
    X_n & Y_n & 1 & 0 & 0 & 0 & -u_nX_n & -u_nY_n \\
    0 & 0 & 0 & X_n & Y_n & 1 & -v_nX_n & -v_nY_n
\end{pmatrix}
\]

\(p\) the vector containing the parameters of the projection matrix

\[
p = \begin{pmatrix}
p_{00} \\
p_{01} \\
p_{02} \\
p_{10} \\
p_{11} \\
p_{12} \\
p_{20} \\
p_{21}
\end{pmatrix}
\]
and \( \mathbf{b} \) the vector containing \( 2n \) image coordinate values

\[
\mathbf{b} = \begin{pmatrix} u_1 \\ v_1 \\ \vdots \\ u_n \\ v_n \end{pmatrix}
\]

A common method of estimation is the ordinary least squares method, in which one tries to compute the estimated \( \hat{\mathbf{p}} \) that minimises the sum of squared errors (SSE) \( \| \mathbf{b} - A\mathbf{p} \| \).

Expanding the SSE as

\[
SSE = \| \mathbf{b} - A\mathbf{p} \|
\]

\[
= (\mathbf{b} - A\mathbf{p})^T(\mathbf{b} - A\mathbf{p})
\]

\[
= [\mathbf{b}^T - (A\mathbf{p})^T](\mathbf{b} - A\mathbf{p})
\]

\[
= (\mathbf{b}^T - \mathbf{p}^TA^T)(\mathbf{b} - A\mathbf{p})
\]

\[
= \mathbf{b}^T\mathbf{b} - \mathbf{b}^TA\mathbf{p} - \mathbf{p}^TA^T\mathbf{b} + \mathbf{p}^TA^TA\mathbf{p}
\]

Differentiating SSE with respect to \( \mathbf{p} \) gives

\[
\frac{\partial SSE}{\partial \mathbf{p}} = -2A^T\mathbf{b} + 2A^TA\mathbf{p}
\]

Setting the derivative as 0 and noting the second derivative results in a positive definite matrix \( A^TA \) to justify the solution is minimum, the estimated projection vector is

\[
\hat{\mathbf{p}} = \begin{pmatrix} \hat{p}_{00} \\ \hat{p}_{01} \\ \vdots \\ \hat{p}_{21} \end{pmatrix} = (A^TA)^{-1}A^T\mathbf{b}
\]

which would be rearranged into a \( 3 \times 3 \) projection matrix used in the transformation between image and real world planes.


2.3.3 Masking

Masking is a technique used to highlight a specific region from the image, which is underneath many types of image processing, including scaling, color correction and cropping in this project. It can be defined as setting certain pixels of an image to some null value such as 0 (black color) so only that portion of our image is highlighted where the pixel value is not 0. Thus, a mask allows the computer to extract regions from images that are of completely arbitrary shape and focus only on the portions of the images.

Masking could be done by thresholding. Three kinds of thresholding will be introduced to tackle different situations, which are simple thresholding, Otsu’s binarization and adaptive thresholding.

Color detection is a variation of simple thresholding. Since color is a spectrum, upper and lower bounds are set to filter the desired color only. It is noted that the color ranges are defined in HSV (Hue, saturation, value) colorspace (figure 2.4a) instead of RGB colorspace (figure 2.4b). It is because the hue channel models the color type, it is very useful in image processing tasks that need to segment objects based on its color.

![Colorspaces](image)
(a) HSV Colorspace  
(b) RGB Colorspace

Figure 2.4: Colorspsaces

Simple thresholding is straight forward. For every pixel, the same threshold value is applied. If the pixel value is smaller than the threshold, it is set to 0, otherwise it is set to a maximum value. One limitation of this algorithm is that the arbitrary chosen value is fixed among all images.

In contrast, Otsu’s binarization avoids having to choose a value and determines it automatically based on the image histogram. An image histogram is a type of histogram that
acts as a graphical representation of the tonal distribution in a digital image. The optimal threshold would be in the middle of two major values in an image histogram.

Lastly, adaptive thresholding determines the threshold for a pixel based on a small region around it. Compare with the previous two thresholding methods using one global value as a threshold, while adaptive thresholding determines different thresholds for different regions of the same image which gives better results for images with varying illumination.

The above three thresholding methods are tested to mask the region needed in the pipeline.

2.3.4 Cropping

In our classifying process, color and texture matter but not the shape of sherds. Thus, with proper masks, one can crop the area that is inside the boundary of sherds only while others are discarded. Furthermore, more than one subimage can be cropped out from the original image in order to create a vast, extensive and standardized dataset.

The procedure of cropping works as follows. First, the program detects objects in an image and identifies the sherd. The next step is to apply masking so that the background is removed, leaving only the sherd. Lastly, data samples are cropped within the region of the masks.

Object detection, masking and cropping are common tasks in computer vision and have already been implemented in OpenCV [13], making this part relatively easy to deal with technically.

Assume thresholding gives the proper masks of objects (Sherd, color card or scale card), then the remaining challenge of this part is to identity which object is a sherd.

![Figure 2.5: Collision detection: Only 4 cases that do not intersect](image)

(a) Case 1  (b) Case 2  (c) Case 3  (d) Case 4
As for each object identified in an image, all cards contain black squares excluding the sherd. By using this explicit knowledge, one could determine if the object is a sherd. More specifically, all black squares and objects are bounded by Axis-aligned Bounding Box (AABB) which are represented by 4 corners’ coordinates. Figure 2.5 demonstrates the only 4 cases that 2 AABB do not intersect. With only 4 comparisons of every pair of red and green bounding boxes, the computer can identify a sherd efficiently.
2.4 Convolutional Neural Network

With standardized sherd images, now we can get the computer to learn from the dataset via different machine learning models, CNN in particular. CNN is a specialized type of neural network that is specifically designed to process pixel data and is widely used in image recognition and processing. They apply a mathematical operation called convolution in place of general matrix multiplication in at least one of their layers [14]. As processing images is computationally intensive and may lead to overfitting to unwanted noise of the images [15]. Therefore, convolutional layers in CNNs lower the number of parameters whilst keeping features critical for image-focused tasks to reduce the computational complexity.

In 2012, the success of Krizhevsky’s AlexNet which is a CNN architecture proves the potential of CNN in computer vision [16]. Many similar variants have since been proposed to achieve better performance, including VGGNet, GoogleNet and ResNet. However, most modifications involve increasing the depth of the CNNs to estimate the target modeling function with increased nonlinearity and better feature representations [17].

The main components of the CNN architecture consist of convolutional layers, pooling layers and fully connected layers (Figure 2.6).

The convolutional layer makes use of kernels that glide through the input and perform scalar
product with each pooled section of the input vector. The convolutional layer convolves each kernel across the spatial dimensionality of the input in order to produce feature maps [18].

Next, the pooling layer aims to gradually reduce the dimensionality of the representation [19]. It operates on each feature map of the input. Max and average pooling are the two most common types of pool that provide the maximum and average of all values from the portion of the image covered by the kernel respectively.

The fully connected layers, which are made up of neurons directly connected to the two adjacent ones, are identical to those in traditional neuron layers in Artificial Neural Network (ANN). Input from the prior pooling layer is used, and it is flattened before feeding into the subsequent layers.

In the project, large pre-trained CNNs like VGG19 and ResNet50 are to be extensively utilized to process the image data and learn the parameters for the classification task. Different hyper-parameters, loss functions and optimizers will be tested on the archaeological data.

As the most common neural network that is applied to analyze visual imagery, CNN can be implemented with different existing libraries. The PyTorch framework is chosen to develop the models for this project due to its simplicity, flexibility and high processing speed [20]. It serves the project well for academic purposes.

2.5 Vision Transformer

While CNN has achieved phenomenal success in computer vision tasks, the fundamental pixel-convolution paradigm suffers from the following difficulties [21]: 1) CNN process image patches uniformly without taking the significance of each patch into account. For instance, a large portion of a cloudy sky may not be as important as pedestrians in segmentation tasks. 2) High-level filters may not apply to a lot of image data even though a small subset of data utilizes such filters. Computational efficiency may suffer if high-level filters are applied to all images. 3) Relationships between features that are spatially separated are poorly captured by CNN. Convolutional filters are limited to small regions. For many tasks, it may be crucial to comprehend how features that are spatially distant from one another interact. There are workarounds available, like increasing the kernel size and model depth. However, these result in a less efficient computer and a more complex model. As a result, the transformer paradigm borrowed from the field of Natural Language Processing (NLP)
will be used as another type of model attempting to address the issues above.

The transformer architecture has been used in NLP to learn the contextual relation between words. It has two distinct mechanisms: an encoder that reads input text, and a decoder that generates the prediction for the task [22]. As it can easily learn long-term dependencies, parallelization is possible and they are excellent for transfer learning, making it widely used in recent years.

Each encoder and decoder consist of a self-attention layer and a feed-forward layer. In a self-attention layer, each input vector from the image data is transformed into 3 vectors, namely the query, key and value vectors. These vectors from different input vectors are then stacked together to form the query, key and value matrices $Q$, $K$ and $V$, an attention score is calculated as follows:

$$Attention(Q, K, V) = \text{softmax}(\frac{Q \cdot K^T}{\sqrt{d_k}})$$

where $d_k$ is a normalization factor dependent on the key vector dimension. The attention mechanism models how important each vector is to a given input vector and hence the relationship between each part, no matter how far apart they are spatially, can be captured. The feed-forward layer is typically made of two linear transformation layers and a nonlinear activation layer, of the following form:

$$\text{Feedforward}(X) = W_2 \cdot a(W_1 X)$$

where $W_1$ and $W_2$ are the linear transformations and $a$ is the activation function.

In the context of computer vision, a tokenizer divides an image’s pixels into a variety of visual tokens that represent the image’s semantic concepts, and a transformer is used to model the relationships between the tokens. The visual tokens can then be fed into image classification layers. iGPT, ViT, and DeiT are three notable transformer architectures that can be used for the image classification task. [23].

The ViT workflow in Figure 2.7 involves splitting an image into patches where each patch is represented by a visual token. Position embedding vectors will be added to model the relative spatial positioning of the patches. The resulting vectors will be fed to a transformer encoder to learn the relationships between patches. Classification can be performed by adding a learnable classification token to the input sequence. By passing the output of the transformer encoder to a typical MLP, the class of the image can be predicted.
Transformers will be investigated to see whether there exists a considerable improvement in the performance of sherd classification and to compare with the CNN approach to study the differences. Again, this will be implemented in Pytorch.
2.6 Summary

The chapter presented the rationale and implementation for the adoption of image processing needed and the two model architecture. Additionally, a new measure of accuracy is introduced to better evaluate models. In the next chapter, the current progress of the project will be discussed.
3 Results and Discussion

3.1 Overview

This chapter demonstrates the progress we made in this project. The project is currently in the machine learning stage. Section 3.2 and section 3.3 illustrate the results achieved in image processing and machine learning respectively. Section 3.4 will discuss the limitations we encountered during the process. Section 3.6 and section 3.5 shows the project schedule and the future plan respectively.

3.2 Current Status: Image Processing

The image processing pipeline has been proposed and deployed. The following would discuss the results the team achieved and the experiments done. Meanwhile, as machine learning models learn from the image data provided, the quality of images is critical to the performance of the stage. The team would continue seeking ways to optimize and improve the pipeline.

3.2.1 Color Correction

Data with both the 24-patch ColorChecker and 4-patch photography scale have been color corrected. As mentioned in Chapter 2.3.1, the color calibration pipeline first performs automatic white balancing, and detects the position of ColorChecker in the raw image. The color patches are found by detecting rectangles in the image, defining the boundaries of the reference colors and then identifying them. Two types of raw images from the database (Fig. 2.1) will be used for demonstration. Figure 3.1 shows that only the reference color patches are detected on the data, and the algorithm draws rectangles around the detected charts.

![Figure 3.1: color checker detection in the two types of raw images](image-url)
After obtaining the color values of the patches on the chart image and the original reference color values, they are fed into the Color Correction Model. As mentioned in Chapter 2.3.1, three different methods of finding the optimal Color Correction Matrix (CCM) have been experimented. Their results are presented below. For each color correction matrix $M$, the $\Delta E$ errors has been calculated in the CIELab color space for more accurate comparison between different methods. Intensity independent mapping method was ultimately chosen to be the color calibration technique used in the pipeline.

### 3.2.1.1 Least Square Method

The initial Color Correction Matrix (CCM) is acquired by utilizing the least square method. The CCM is then optimized by minimizing the loss function. The loss function is defined as the weighted square sum of the color difference between the standard reference data and detected image data. To calculate the color difference, CIE2000 is used because it gives an overall best performance compared to other color-distance formulae. [24] Hence, the color space is transformed into CIE Lab in order to use CIE2000.

**Table 3.1: Color correction result using least squares**

<table>
<thead>
<tr>
<th>Unprocessed</th>
<th>Color corrected</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Unprocessed Image" /></td>
<td><img src="image2.png" alt="Color corrected Image" /></td>
</tr>
</tbody>
</table>

Table 3.1 shows the result of color correction using traditional least squares method. The performance is not satisfactory, as the whole image has an incorrect prominent yellowish tint. The mean CIELAB $\Delta E$ error using the Macbeth 24 ColorChart is 8.1 and the mean $\Delta E$ error using the 4-patch color card is 11.7, which is very high. That is because there is incomplete achromatic and chromatic calibration data, using only the four-patch color card. Therefore, without any a priori premise about which surfaces that will be mapped with low or high error, the white and yellow color, particularly in this case, may be calibrated
with high colorimetric error depending on the data. This variable performance is not desirable. [11] To mitigate this problem and ensure the correct mapping of a certain color, constrained least-squares regression was experimented in this project.

### 3.2.1.2 Constrained Least Squares Regression

There is generally a higher importance of mapping white with higher accuracy for color reproduction, so the white point preserving method was used to improve the performance. The white point preserving constrained least squares method finds the mapping which maps the white surface with very low error when solving the least squares problem.

Table 3.2: Comparison between using least squares and constrained least squares method

<table>
<thead>
<tr>
<th>Unprocessed image</th>
<th>Constrained least squares</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Unprocessed Image" /></td>
<td><img src="image2.png" alt="Constrained Least Squares" /></td>
</tr>
<tr>
<td><img src="image3.png" alt="Background Color" /></td>
<td><img src="image4.png" alt="Background Color" /></td>
</tr>
</tbody>
</table>

Table 3.2 shows the result of color correction using constrained least squares regression. The color patches are calibrated to be very similar to their reference values, with the background color corrected to its original white color. The mean CIELAB $\Delta E$ error using the 4-patch color card has improved from 11.7 to 10.8. The performance is better than the naive least square method, and the sherd is closer to the color of the physical object. Finlayson has also reported that the white point preserving mapping performs better than the basic least-squares method. It does not lead to significant colorimetric overhead, and the errors from both fitting techniques are similar [11]. Therefore, applying the constrained least squares method is more preferable than the naive least square regression.
3.2.1.3 Intensity Independent RGB-to-XYZ color calibration  A major disadvantage of the least square methods is that they do not address the problem of nonuniform lighting variation in the image. A large part of the dataset has photoed part of the ColorChart under the shadows and another part being much brighter. However, a necessary condition for using least squares color correction is that RGBs measured in a real scene has the same shading profile as reference XYZs, unless the shading profile is already known which is very costly. Therefore, the intensity independent RGB-to-XYZ calibration method is used. Nelder-Mead searching has been performed to find the CCM that maps RGBs to XYZ which minimizes the shading independent angular error.

Table 3.3: Final color correction results using intensity independent calibration method

<table>
<thead>
<tr>
<th>Unprocessed image</th>
<th>Final results</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Unprocessed image" /></td>
<td><img src="image2" alt="Final results" /></td>
</tr>
</tbody>
</table>

Table 3.3 shows the final results using the intensity independent color correction technique. The corrected color patches are significantly brighter and closer to their true reference colors than the raw images. The white background is also accurately captured. The mean CIELAB $\Delta E$ error using the Macbeth 24 ColorChart is 5.9 and the mean $\Delta E$ error using the 4-patch color card is 8.4, which are the smallest of all experimented methods. That is because the nonuniform irradiance across calibration target would not greatly affect the calibration result. However, the four-patch photography scale lacks both chromatic and achromatic reference colors. Therefore, its performance is not on par with the 24 ColorChart, which has sufficient reference colors for precise calculation of the CCM. This is
an unavoidable limitation, since most of the sherds were taken with the 4-patch scale, and all
known reference colors in the images have already been utilized. It has been recommended
to use the 24 ColorChart in the future documentation of excavated materials.
3.2.2 Geometric Calibration Results and Interpretation

The calibration aims to adjust the geometric properties of an image to recover a desired perspective from the misalignment between the desired world plane and observed image plane.

An image is retrieved from the database of the archaeological excavation findings with a calibration card placed nearby.

A workflow of camera calibration is proposed as follows:

1) At least four reference points are detected from the calibration card in data images.

2) With the set of points in the desired real world space and their correspondences onto the image coordinate space, a camera projection matrix which is a homography in this case is computed.

3) The inverse of the homography matrix is applied to every other pixel to construct a standardized image.

4) Apply steps 1) to 3) to every image in the raw data.

The proposed workflow has been implemented on a set of images of the same calibration card type.

![24 color calibration card](image)

Figure 3.2: Outermost corners (in red) in a 24 color calibration card

In the project’s context, with a sample data image as example, image plane coordinates of \([263, 109], [268, 151], [373, 105], [376, 147]\) corresponding to four corners of the calibration board (Fig. 3.2) are identified.
The goal is to back-project them from the image plane onto the real world plane as the desired coordinates.

A prior ratio of \( r = \frac{\text{number of pixels}}{\text{unit cm in real world}} \) is designated so that scaling takes place along with the perspective correction. This secures the consistency of the size of the calibration board such that it is the same for our image data to be input into the machine learning model. Texture and pattern scales could matter substantially in distinguishing certain pottery sherds from some others, thus the need to carry out the scaling.

To determine the real world coordinates of the four points detected, a fixed point can be assigned to be of the same coordinates as in the projected image. The upper left most point is set to be identical before and after the projection transformation, of coordinates \( [u_{\text{fixed}}, v_{\text{fixed}}] \) where \( u_{\text{fixed}} = 263 \) and \( v_{\text{fixed}} = 109 \). The three other points are assigned as follows \([u_{\text{fixed}}, v_{\text{fixed}} + 2r], [u_{\text{fixed}} + 5r, v_{\text{fixed}}], [u_{\text{fixed}} + 5r, v_{\text{fixed}} + 2r]\).

The mapping of points are:

\[
\begin{bmatrix}
263 \\
109 \\
268 \\
151 \\
373 \\
376 \\
147
\end{bmatrix} \rightarrow \begin{bmatrix}
[u_{\text{fixed}}, v_{\text{fixed}}], [u_{\text{fixed}}, v_{\text{fixed}} + 2r], [u_{\text{fixed}} + 5r, v_{\text{fixed}}], [u_{\text{fixed}} + 5r, v_{\text{fixed}} + 2r]
\end{bmatrix} = \begin{bmatrix}
[263, 109], [263, 109 + 2r], [263 + 5r, 109], [263 + 5r, 109 + 2r]
\end{bmatrix}
\]

The system of linear equations to be solved is then:

\[
\begin{bmatrix}
263 \\
109 \\
268 \\
151 \\
373 \\
376 \\
147
\end{bmatrix} \begin{bmatrix}
263 & 109 & 1 & 0 & 0 & 0 & -263(263) & -263(109) \\
0 & 0 & 0 & 263 & 109 & 1 & -109(263) & -109(109) \\
263 & 109 + 2r & 1 & 0 & 0 & 0 & -268(263) & -268(109 + 2r) \\
0 & 0 & 0 & 263 & 109 + 2r & 1 & -151(263) & -151(109 + 2r) \\
263 + 5r & 109 & 1 & 0 & 0 & 0 & -373(263 + 5r) & -373(109) \\
0 & 0 & 0 & 263 + 5r & 109 & 1 & -105(263 + 5r) & -105(109) \\
263 + 5r & 109 + 2r & 1 & 0 & 0 & 0 & -376(263 + 5r) & -376(109 + 2r) \\
0 & 0 & 0 & 263 + 5r & 109 + 2r & 1 & -147(263 + 5r) & -147(109 + 2r)
\end{bmatrix}\]

By multiplying the inverse of the \( 8 \times 8 \) matrix on both sides, the projection matrix \( P_p \) can be found.
Then each point in the desired calibrated plane can be found from:

\[
\begin{pmatrix}
    s_i u_i \\
    s_i v_i \\
    s_i
\end{pmatrix}
= \begin{pmatrix}
    p_{00} & p_{01} & p_{02} \\
    p_{10} & p_{11} & p_{12} \\
    p_{20} & p_{21} & p_{22}
\end{pmatrix}
\begin{pmatrix}
    X_i \\
    Y_i \\
    1
\end{pmatrix}
\]

\[
\begin{pmatrix}
    X_i \\
    Y_i \\
    1
\end{pmatrix}
= \begin{pmatrix}
    p_{00} & p_{01} & p_{02} \\
    p_{10} & p_{11} & p_{12} \\
    p_{20} & p_{21} & p_{22}
\end{pmatrix}^{-1}
\begin{pmatrix}
    s_i u_i \\
    s_i v_i \\
    s_i
\end{pmatrix}
\]

However, when using only four points, the transformation in some samples could introduce a perspective warping that is not suitable for inspection by archaeologists.

The number of reference points used is a probable reason for certain undesirable results. More points used may lead to a more accurate result in capturing the various image distortions occurring in the photographing process. When only the minimum of 4 are points used, it could lead to parts of the image deviating slightly from their true appearance in the real world. Furthermore, the points to be selected could affect the calibration. In the past implementation, the four reference points are rather close to one another, using points farther away may enhance the quality of calibration of the area within the area enclosed.
by the points. To tackle the above mentioned problems, using more than four points and the minimization of the sum of squared errors method to estimate the homography matrix is used.

Observe that in Fig. 3.3 in using more than four points, the original tilting in a vanishing manner towards the left is amended. When compared with the pre-calibrated image on the left, the calibration is oriented in a manner more aligned with that of a regular rectangle. The top and bottom edges of the calibration card match more with the horizontal axis than the pre-calibrated image.

### 3.2.2.1 SIFT Method

One implementation to use more than 4 points is to utilize SIFT to carry out the identification of numerous interest points in a separate query image of the desired perspective which only has the calibration card, obtaining \( \text{des}_q \) as the query key points descriptors to describe the information entailed by the interest points. On the training full image which contains the sherd and a calibration card that may differ slightly from the query image, SIFT is again carried out to obtain descriptors \( \text{des}_t \). A matching function is used to determine which interest points are contained within the training image \( F_{\text{match}}(\text{des}_q, \text{des}_t) \) where a common \( F_{\text{match}}(\cdot) \) used in the industry is the FLANN (Fast Library for Approximate Nearest Neighbors) based matcher. The steps later are similar to the current workflow in use, which involves computing the homography between the two sets of points that are identified as good enough matches, giving a \( 3 \times 3 \) projection matrix \( P \) to be used for the required transformation. The advantage of this method is that there is no need to write code logic to deduce the desired coordinates, as well as more points being used that may contribute to better calibration performance. The downside is the inconsistency of the calibration matrix computed for each image, and potential mismatching of correspondences detected.

Random Sample Consensus (RANSAC) is used to remove outlier correspondences given by the matcher.

As seen in the Fig 3.4 in which a 24-color calibration card is used, the red dots on the left image indicate the key points detected by SIFT for both the smaller query calibration card image and the larger training image. The green lines show the matches between the query image and the training image. The outlier correspondences can be removed after applying RANSAC such that only the key points from the calibration card in the larger image correspond to the key points in the smaller query image. This helps increase the robustness of the SIFT method to prevent the existence of outlier matches from interfering
3.2.3 Masking by thresholding

Masking is widely utilized in our image processing pipeline, like color patch detection and object detection. To find regions needed for different tasks, different thresholding techniques are applied.

Simple thresholding is used to detect color patches on the color card and scale card. Upper and lower bounds are set then the computers are able to filter out the desired color within the range as in figure 3.5, so that color patches are extracted for color correction and scaling.

Additionally, masks of the sherd in an image are needed for cropping. As shown in figure
3.6, either simple thresholding or Otsu’s binarization does not work well. It is because an image has different lighting conditions in different areas. Take figure 3.9a as an example, the middle region is brighter while two sides of the image are darker. This situation is common in the real-life dataset.

![Original Image](image1)

(a) Original Image

![Simple thresholding](image2)

(b) Simple thresholding

![Otsu Binarization](image3)

(c) Otsu Binarization

Figure 3.6: Masking with global thresholding

As a result, adaptive thresholding is applied in detecting objects. As explained previously, this can tackle the issue of the image dataset. After some fine-tuning of parameters, the algorithm is able to identify objects in an image. And after filling, masks of objects can be created. Figure 3.7 illustrates the masking by adaptive thresholding.

3.2.4 Cropping

As for the cropping part, the program can locate objects in an image using masks. Figure 3.8 illustrates the object detected in a sample image and thus it gives the location information of objects (in green bounding boxes). The next step is to identify sherd in an image.
It is noted that all cards contain black squares (in red bounding boxes) which are already detected in color correction and scaling.

As aforementioned, one could make use of the AABB collision detection algorithm to determine which object is a sherd efficiently and accurately. Other means are also tested for sherd detection, like template matching and histogram matching. However, the performance is not good.

Then, the sherd is cropped out and a mask is constructed as in figure 3.9, outlining the region of the sherd only. The next step is to generate sub images data within the mask.

As stated previously in the methodology, the background (the white table) is irrelevant data. Subimages that contain the sherd only would be generated for training. Currently, 5 subimages of size 256px by 256px dimension would be extracted randomly from each image.
Figure 3.9: Sherd and its mask in an image

Figure 3.10: Sample image data generated

Figure 3.10 illustrates the effect. Note that these subimages may contain overlapped areas yet the color distributions would be different thus these are not duplicated data.

By cropping in such a manner, the program is able to create a vast, extensive and standardized dataset for the deep learning models. Random cropping is also a way of data augmentation which hopefully would help train the models.

3.3 Current Status: Machine Learning

After the processed data is ready, a preliminary CNN model has been accomplished to classify the sherds. Here, the architecture of the initial model will be introduced. Currently, the test accuracy is about 25% which is low. The team will also analysis the results obtained in this section and continue refining the model.
3.3.1 Model for Transfer learning

In the preliminary implementation stage, Resnet 18 is utilized. The main reason being it is relatively shallow and has the fewest parameters out of Resnets, so as to facilitate quicker results for inspection.

Resnets makes use of the idea that rather than trying to use multiple stacked layers of nonlinear functions to obtain an approximation of the desired mapping from input to output, a residual mapping is fit instead:

\[ F(X) := H(X) - X \]

where \( H(X) \) is the desired mapping, \( F(X) \) the residual mapping. It is hypothesized [25] that the residual mapping is easier to optimize than the original desired mapping.

In our implementation, the last layer of Resnet 18 is modified to give an output equal to the number of prediction classes for our task. The pretrained parameters are kept. However, as the results to be explored in detail later shows, Resnet 18 is still too deep for our task with a relatively small dataset which leads to heavy overfitting. A simpler model and better cleansing of data labels is to be discussed for use in the next steps.

3.3.2 Loss Function

For the task of classification into multiple discrete labels, cross entropy loss is chosen for training and evaluation of the model.

For the \( i \)th sample \((x_i, y_i)\) of data, the output of the model as a vector of unnormalized log-odds (logits) \( p_i \) can be normalized via the softmax function into the vector \( f_{\theta}(y | x_i) \) where the \( k \)th element is

\[
  f_{\theta}(y | x_{i,k}) = \frac{\exp(p_{i,k})}{\sum_{j=1}^{C} \exp(p_{i,j})}
\]
with $C$ the number of prediction classes. $f_\theta(y \mid x_i)$ can be interpreted as the density function of the model’s prediction of label given parameters $\theta$ and observed data.

The target ground truth is in form a one-hot vector with distribution

$$g(y \mid x_i) = 1(y = y_i)$$

where $1()$ is the indicator function.

The cross entropy for the sample $(x_i, y_i)$ is given by

$$H(g(y \mid x_i), f_\theta(y \mid x_i)) = -\sum_{y \in Y} g(y \mid x_i) \log(f_\theta(y \mid x_i))$$

$$= - \log f_\theta(y_i \mid x_i)$$

The entire cross entropy loss from dataset $D$ is

$$L_{CE} = \sum_{(x_i, y_i) \in D} H(g(y \mid x_i), f_\theta(y \mid x_i))$$

$$= \sum_{(x_i, y_i) \in D} - \log f_\theta(y_i \mid x_i)$$

To justify the use of this loss, note that in trying to minimize this loss, the likelihood function of parameters $\theta$ given data $X$ is maximized:

$$\arg \min_{\theta} L_{CE} = \arg \min_{\theta} \sum_{(x_i, y_i) \in D} - \log f_\theta(y_i \mid x_i)$$

$$= \arg \max_{\theta} \sum_{(x_i, y_i) \in D} \log f_\theta(y_i \mid x_i)$$

$$= \arg \max_{\theta} \prod_{(x_i, y_i) \in D} f_\theta(y_i \mid x_i)$$

which is equivalent to maximum likelihood estimation of all samples in the dataset.
3.3.3 Optimizers

Two optimizers are utilized and compared in the training stage.

3.3.3.1 SGD Stochastic gradient descent (SGD) is a variant of the vanilla gradient descent:

\[
\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta)
\]

where \(\theta\) denotes the parameters to be updated, \(\eta\) the learning rate and \(J\) the loss function.

In SGD, rather than computing \(\nabla_{\theta} J\) on the entire dataset to update the parameters, only a subset of data will be used for each update:

\[
\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta; x_{i:i+n}; y_{i:i+n})
\]

where \(n\) is the size of the subset of data to be used per update.

3.3.3.2 Adam Adaptive Moment Estimation (Adam) adjusts learning rates adaptively for every parameter, based on the correctness of direction as well as update frequency of each parameter. It combines similar concepts as RMSProp with a decaying average of gradients from previous steps, and momentum that reinforces correct direction and reduces oscillations when learning.

In essence the update rule is:

\[
\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t + \epsilon}} \hat{m}_t
\]

with the first and second moments of previous gradients stored as

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

where \(m_t\) is a momentum-like quantity attempting to estimate the first moment of gradients \(E[g]\) using the previous step gradient and the past momentum \(m_{t-1}\), similar for \(v_t\) except
the second moment of gradients $E[g^2]$ is estimated instead of the first, with $\beta_1$, $\beta_2$ being fixed decay constants.

To prevent the problem of $m_t$ and $v_t$ biasing off to zero due to their initialization as zero vectors [26], a bias corrected estimation of the first and second moments of gradients are:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

### 3.3.4 Preliminary Model Evaluation

The data has been split into 80% for training, 10% for validation, and 10% for testing. Each of them should have the same distribution of data class to ensure that each split has an accurate representation of the dataset. For the preliminary training results, a severe problem of overfitting has been found. As shown in Table 3.4, the training accuracy (0.96) is extremely high and the loss (0.20) is very low. However, both the validation accuracy (0.26) and test accuracy (0.25) deviates hugely from the training accuracy, and the test loss is high (4.59). That means that the model has high variance with high validation error but low training error. Overfitting to training data is severe and the model does not generalize well.

![Graphs of loss and accuracy in preliminary training](image)

Figure 3.12: Graphs of loss and accuracy in preliminary training

On top of that, the model converges very quickly. According to Figure 3.12, it has already converged at around epoch twelve. However, the model performs very poorly and overfit,
that could be due to data scarcity and many of the data is underrepresented. Therefore, data augmentation has been performed on the data to mitigate overfitting.

3.3.4.1 Data Augmentation is a method used to increase the diversity of the training set by applying random transformations. Random vertical and horizontal flipping has been done to flip the images randomly with a 0.5 probability. Random rotation to rotate the images by some angle, and random cropping to crop the images at a random location will also be done in the next step.

After performing random vertical and horizontal flipping, it has been found that the overfitting problem has improved and variance has been lowered. As shown in Table 3.4, the training accuracy has lowered from 0.96 to 0.23, and training loss has increased from 0.20 to 2.85. The training error has increased. Moreover, the model has a better performance than the preliminary one, as validation loss remains below 4.0, while the one before augmentation has a steady validation loss of around 4.5. The test loss has also decreased from 4.59 to 3.60. That is because data augmentation has improved the model prediction and prediction confidence by creating variability in data.

![Graphs of loss and accuracy after data augmentation, using SGD](image)

Figure 3.13: Graphs of loss and accuracy after data augmentation, using SGD

However, Figure 3.13 illustrates that the validation accuracy and loss fluctuates a lot. It shows that there may still be the problem of overfitting. Both the training and validation set are not representative enough, and the model is not properly regularized. Possible solutions that would be experimented are obtaining more data points and performing model regularization. On top of that, the model is still not efficient because it performs poorly on validation and test sets, without any improvement to the test accuracy (0.25) and validation accuracy (0.19). It also does not have enough confidence when making predictions, as its test loss is high (3.60). Several sources of error have been identified in Chapter 3.4 and
possible improvements are suggested in Chapter 3.5.

Table 3.4: Loss and Accuracy values of different models

<table>
<thead>
<tr>
<th></th>
<th>Preliminary model, SGD</th>
<th>After Augmentation, SGD</th>
<th>After Augmentation, Adam</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training</td>
<td>Test</td>
<td>Training</td>
</tr>
<tr>
<td>Acc.</td>
<td>0.96</td>
<td>0.25</td>
<td>0.23</td>
</tr>
<tr>
<td>Loss</td>
<td>0.20</td>
<td>4.59</td>
<td>2.85</td>
</tr>
</tbody>
</table>

3.3.4.2 Comparison between SGD and Adam optimizer After performing data augmentation, the performance of the model using SGD and Adam optimizer are also compared. As presented in Figure 3.14 and Table 3.4, both training and test loss using Adam optimizer are higher than using SGD, and the training accuracy of using Adam optimizer is lower than using SGD. The performance of the model with SGD is better than the one with Adam. Studies also show that the generalization performance of Adam is significantly worse than SGD on image classification problem. [27] Therefore, the SGD optimizer has been used for this current stage.

3.4 Limitations and Difficulties

This section discusses the limitations of image processing and the dataset. Since this is an industrial project, some problems the team faces are common and close to the real-life situation. However, this problem analysis also provides direction on how the team can improve the pipeline and the model in the future.
3.4.1 Image Processing

In the data processing of this project, one major issue is doing the color calibration with the 4-patch color card which is used in the majority of images. As explained in Chapter 3.2.1, CCM derived from only 4 distinct color patches cannot calibrate colors as well as from a 24-patch card. It is a mathematical bottleneck since it provides fewer reference colors. Thus, our team proposed to use other image color enhancement techniques like white balancing and constrained least squares regression to minimize the impact. Meanwhile, the team will continue to explore other color calibration algorithms.

Another major issue is the white layers on sherds which are called incrustation. Incrustation is caused by years of weathering conditions, like pressure, temperature change, humidity, etc. The incrustation covers the original color and texture which confuses the models, resulting in poor performance. Ideally, the cropped pieces of sherd should not include those incrustations as it does not belong to the original texture and color of the sherd. However, it is anticipated that a minimal surface is covered. Thus, only a small proportion of cropped images contain incrustations. Furthermore, it involves another complex computer vision task called segmentation which the team needs more data to identify those white residues. This limitation is considered a minor drawback and is not included in the scope of this project.

3.4.2 Pre-Processed Dataset

After image processing, a few problems have been found in the dataset that may contribute to the low accuracy. One major challenge would be a relatively small dataset. Despite having 40,000 sherd photos, only about 2,000 of them are labeled. There are 92 classes in total so there is only around 20 labeled raw images for each class on average. Although data augmentation techniques like random cropping and flipping have been applied to create synthetic samples, it would be better to have more labeled images provided by the archaeological team in the future.

Additionally, the labeling of the dataset is confusing. There are 92 classes but some classes are very similar. For example, there are 6 different classes for different degrees of roughness, from yellow coarse to yellow coarse-05. The computer may not be able to distinguish the subtle difference, and human errors in labeling are very likely. Also, every piece of sherd has front and back sides but they share the same label even though the color may look different. In supervised learning, the label is what the computers learn from the dataset. Thus, the labeling issue is a major limitation that results in the low accuracy of the current
Lastly, the data is imbalanced even after processing. As shown in Figure 3.15, only a few classes have more than 200 data samples while most of them have less than 60 images. It is a common challenge in real-life machine learning tasks as we cannot anticipate what sherds can be found on the site. A lot of classes may be underrepresented. It is also possible there are new classes in the future.

3.5 Future Plan

Currently, the initial model is only about 25% in accuracy and the next stage is to conduct more experiments on various deep learning models and try to address the problems listed above like overfitting and imbalanced data in order to improve the accuracy.

The immediate step is to refine the dataset. More data augmentation techniques will be applied to generate more synthetic samples. So far, we have used flipping and random cropping. We may test using random rotation as well. Furthermore, similar classes will be combined to simplify the problem. For example, the label brown dark coarse-1 to dark brown coarse-5 may combine into 1 single class called brown dark coarse. By doing so, it is hoped that we may have a more extensive, balanced and vast dataset so that over-fitting is less likely to happen. Deep learning models usually require a large dataset of decent quality. It is worth spending more effort to refine our dataset.
Regarding deep learning, the team would continue to fine-tune the current model to achieve better results. The team will try to divide the classification task into two models: one to classify based on texture, and the another based on color. After that, the results will be joined together to see if a better accuracy can be achieved. Therefore, it would be easier for the model to understand the relationship between the labels and the images. There will also be more data for each class, alleviating overfitting and data imbalance problem.

Moreover, other neural networks will be explored in the next stage. The team will build simpler models, instead of ResNet to lower the model complexity, which mitigates the problem of a small dataset. Also, many pre-trained models do not take color into consideration when performing classification. Building a model from scratch can possibly allow it to learn better from labelled data. Since there are a lot of unlabeled data, semi-supervised learning models may be useful taking account of them too. Also, in a recent development, ViT would be a choice for this classifying task as a relatively new deep learning model.
3.6 Schedule and Milestones

Table 3.5 shows the project schedule. The team has finished setting up the image processing pipeline and training a preliminary model. Currently, the team is fine-tuning the model and exploring other CNN and ViT architectures.

<table>
<thead>
<tr>
<th>Month</th>
<th>Task</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep 2022</td>
<td>Meet with project supervisor</td>
<td>Completed</td>
</tr>
<tr>
<td></td>
<td>Write project proposal and website</td>
<td>Completed</td>
</tr>
<tr>
<td>Oct 2022</td>
<td>Study OpenCV for computer vision and image processing</td>
<td>Completed</td>
</tr>
<tr>
<td></td>
<td>Image data cleansing</td>
<td>Completed</td>
</tr>
<tr>
<td></td>
<td><strong>Milestone 1</strong> A sufficiently large data set of processed images for training</td>
<td>Completed</td>
</tr>
<tr>
<td></td>
<td>Images color corrected and cropped to the same size</td>
<td>Completed</td>
</tr>
<tr>
<td>Nov 2022</td>
<td>Literature review for CNN and vision transformer</td>
<td>Completed</td>
</tr>
<tr>
<td></td>
<td>Develop preliminary models to establish baseline</td>
<td>Completed</td>
</tr>
<tr>
<td>Dec 2022</td>
<td>Continue to develop and train models</td>
<td>Completed</td>
</tr>
<tr>
<td></td>
<td>Prepare first presentation</td>
<td>Completed</td>
</tr>
<tr>
<td></td>
<td><strong>Milestone 2</strong> At least 1 model is trained to classify sherds into groups</td>
<td>Completed</td>
</tr>
<tr>
<td>Jan 2023</td>
<td>Interim report</td>
<td>Completed</td>
</tr>
<tr>
<td></td>
<td>Integrate models with various architectures</td>
<td>In Progress</td>
</tr>
<tr>
<td>Feb 2023</td>
<td>Fine-tune and wrap up models built</td>
<td>In Progress</td>
</tr>
<tr>
<td></td>
<td>Conduct experiments on different models and data sets</td>
<td>In Progress</td>
</tr>
<tr>
<td></td>
<td><strong>Milestone 3</strong> Models with various architectures are trained to do classification</td>
<td>Pending</td>
</tr>
<tr>
<td></td>
<td>Include CNN and vision transformer models</td>
<td>Pending</td>
</tr>
<tr>
<td>Mar 2023</td>
<td>Evaluate and compare the performance of different models</td>
<td>Pending</td>
</tr>
<tr>
<td>Apr 2023</td>
<td>Finish final implementation</td>
<td>Pending</td>
</tr>
<tr>
<td></td>
<td>Finish final presentation and report</td>
<td>Pending</td>
</tr>
</tbody>
</table>
4 Conclusion

Inspired by the recent advancement in computer vision and machine learning, this project attempts to explore various deep learning models to classify and compare sherds unearthed in the Vedi Fortress Archaeological Site, Armenia.

The first stage of the project will utilize OpenCV to do data processing, including scaling, color correction and cropping. The next stage will develop deep learning models for classification. Currently, the team has implemented the image processing pipeline and a preliminary model is trained. The accuracy is about 25% and a few problems are identified in the dataset, which will be addressed in the future. Meanwhile, the team will explore more architectures and fine-tune the current model to achieve a better result.

It is hoped that the project can yield insights into archaeology, allowing archaeologists to handle the tremendous amount of ancient objects excavated with the aid of machine learning in the future.
References


