Technology Feasibility Analysis

Version 2.3 - 4/4/2024

CRAFT (Ceramic Recording Automation and Classification Team)

Project Sponsor:
Dr. Leszek Pawlowicz

Faculty Mentor:
Vahid Nikoonejad Fard

Course Organizer:
Dr. Michael Leverington

Team Members:
Kimberly Allison
Aadarsha Bastola
Alan Hakala
Nicholas Wiley

Table of Contents

1. Introduction

The rapid progression of modern technology has brought with it the modernization of many long existing scientific fields. While most areas of science have seen rapid increases in progress, reliability, and efficiency in their field, archeology has fallen behind due to the uniquely human decisions involved in the field. Did you know that 40% of archeologists disagree with each other on their sherd identifications? Even worse, archeologists disagree more with themselves; it is not uncommon for veteran archeologists to change their minds over *half* of the time on sherd classifications. These visual assessments, which originate from years of experience, and extensive knowledge of art, history, culture, and many other subjects, do not lend themselves to something a computer can easily be helpful in.

Unfortunately, many archeologists do not get the opportunity to reassess their work. Many sherds are historically, culturally, and religiously significant to the indigenous groups they originate from. Since these indigenous groups expect to have their property returned to them researchers have a limited window of time to make their assessments. Losing the ability to reassess sherds makes the already present identification issues even more problematic. If researchers have tens of thousands of sherds archiving all of them poses a significant challenge.

The recent strides taken in AI, machine learning, and neural networks are constantly shortening the number of tasks we believe can only be done by humans. Our sponsor, Dr. Pawlowicz, plans to use neural networks to reduce inaccuracy in sherd identification. Combining his expertise in archeology with personal research into machine learning he has developed a Convolutional Neural Network (CNN) which can accurately and reliably identify sherds. His goal is to refine this CNN and make it accessible to the archeology community.

In its current state the CNN and its supporting technologies are lacking in many of these areas. The current model was restrained by Dr. Pawlowicz's personal computer, which not only limits how fast the CNN can be tested and trained but restricts it to being an already outdated model. Aside from this our CNN, as with all machine learning models, needs large data sets to be trained on. The current data upload method requires Dr. Pawlowicz to manually photograph each sherd. With thousands of sherds regularly coming in for research the unique physical nature of our data is creating a massive bottleneck on training the CNN. Aside from this the CNN's

hyperparameters have had little development and have been identified by our sponsor to need significant further work.

There are several glaring issues with the current state of the solution that our team will have to resolve to bring the solution to a finished and functional state. As mentioned earlier, the CNN is not yet at a level of accuracy and reliability for it to be considered in a finished state. To bring the CNN up to a level of accuracy for it to have real applications in the field our team will update the CNN's outdated model. To correct the inefficiency with uploading CNN data, our team seeks to create and develop a conveyor belt application that will automate the process of sherd data capture and upload. After the CNN has made its assessment, the application will store the newly identified sherd in a database. This gives researchers the opportunity to go back and reexamine no longer available sherds. Our sponsor's vision is not just to create an accurate neural network but to bring this CNN to the archeology community so that it can remedy the widespread issue of inaccuracy and disagreement in sherd identification. To do this, the technology needs to be accessible. Our team is developing a mobile application to make the CNN's identification abilities available to anyone.

2. Technological Challenges

2.1. Improvement of Image Classification Model:

Our sponsor has trained an early version of the CNN model, which is based on ResNet152V2 which classifies sherds. We aim to increase the accuracy and consistency of the model. However, the CNN we have right now is limited by computational power during the training, and we will need to train the newer version of the CNN in a computing cluster. We might need to venture into the possibility of using Visual Transformers instead of CNN or a better model than ResNet152V2 to improve the model.

2.2. Mobile Application:

The application here serves as the front end for a user to upload a picture of a sherd. Then, the app uses an image classification model to classify the sherd. The user then can edit the classification if the model has failed to classify the sherd correctly. The app then can save this data to a database. Since the app is meant to be used on a site, we need to keep in mind that there might not be internet access all the time to upload the data directly into a database. Therefore, the app should be able to save the data in the cache. Then, when the device is connected to the internet, the app should upload all the cached data into the database. We also must focus on maximizing the user experience in the application, as we expect this application to be rolled out to a wide audience.

2.3. Conveyer Belt Application:

Machine learning algorithms require large data sets to be trained with. The unique nature of our data being physical artifacts introduces significant overhead to the data training process. This presents our first issue of how we will digitally capture the physical sherd data. With thousands of sherds to upload Dr. Pawlowicz's current method of individually photographing sherds is too inefficient. Another factor to consider is that our data is not an indefinitely available resource. Many of the sherds belong to Native American tribes who expect to have their culturally significant historical artifacts returned to them. Often archeologists lose the chance to make a reassessment of their sherd classification so we will have to find a way to make our captured

sherd data available on a database to be accessed for further assessment. Once we can reliably and efficiently capture sherd data and upload it to a database, we will want to have the sherd processed by an image classification model so it can evaluate the sherd. The CNN's evaluation of the sherd should be stored with it in the database as well.

2.4. Image Classification Model Integration:

Once we have an improved Image Classification model, this model then needs to be integrated into the mobile application. Doing so will enable more archeologists to make use of it, which is one of our sponsor's core goals. Like the mobile app our conveyor belt application will need to integrate the improved model for batch processing of sherds.

3. Technology Analysis

3.1. Mobile Application

3.1.1. Introduction to issue

The current version of the mobile application is a prototype with limited functionality. The application can take photos and classify them, but the classification data cannot be manipulated or saved. Additionally, the app is not available on systems that utilize iOS.

3.1.1. Desired Characteristics

An optimal mobile application should include implementation of the improved convolutional neural network, the ability to modify classifications determined by the neural network, the ability to upload gathered information to a database, and an improved, intuitive user interface.

- **Image Classification Model Implementation:**
  The framework should have a way to implement the Image Classification model, so that the model can be used by a user in the application.
- **Performance:**
  The framework should be as efficient as possible, as the CNN does take a good chunk of processing power, we would ideally need to have a framework that does not add extra overhead for the classification process.
- **Database Support:**
  Our goal is to make an app that saves the classified data in a database. Even in cases where the device is not connected to the internet, the app should be able to cache the data, then when the device is connected to the internet, the app is then able to upload the results into the database. So, it is absolutely vital to have caching and database support for the application.
- **Intuitive User Interface Support:**
  Since the application is created for a wide audience, we must make sure anyone is able to use the application. The app shall be created using modern design principles and should

be highly usable. Our vision for the application is one that is simple for our users to operate, with a very flat learning curve.

- **Modularity and Maintainability:**
  We want the framework to be as modular and as simple as possible to help with the maintainability of the application. An ideal approach for modularity and maintainability would be to have a single codebase for all the platforms.

3.1.2. Alternative Approaches

- **Flutter:** Flutter is an open-source software development kit developed by Google Inc. It was initially released in May 2017. Flutter can develop native apps in an array of platforms from a single codebase. Flutter is a framework based on an efficient and modern programing language, Dart. PUBG Mobile, eBay, and Google Classroom are some notable applications that were created using Flutter.
- **React Native:** React Native is a cross platform mobile development framework developed by Meta Inc. It was initially released to the public on March 26, 2015. This framework allows developers to use React along with Native Platform Capabilities. Some notable applications that utilize React Native are Facebook, MS Office, Tesla App, Nerd Wallet, etc.
- **.NET MAUI:** .NET MAUI is multi-platform application UI toolkit based on the .NET framework used to build cross platform native applications. It was developed by Microsoft Inc. and released on May 23, 2022. NBC Sports Next, Escola Agil, etc. are some apps created using this toolkit.
- **Swift:** Swift is a high-level programming language created by Apple Inc., initially released on June 2, 2014, for the development of applications in their platform IOS, iPadOS and MacOS. This language cannot be used to develop applications for any other platforms that is not Apple's. Lyft, LinkedIn, Airbnb, and Slack are some apps that are written in Swift.
- **Kotlin:** Kotlin is a modern programming language created by JetBrains, initially released on July 22, 2011. Kotlin is officially supported by Google for Android development. It can be used to create apps on another platform however it is only recommended for the Android platform. Uber, Duolingo, and Netflix are some apps created using Kotlin.

3.1.3. Analysis

We analyzed all the alternatives we considered and scored them in this section.

- **Flutter:** Flutter is based on Dart, which is a compiled language that tends to offer better performance close to Native. Flutter can have a single codebase for IOS and Android. The app in flutter is rendered directly by the GPU, which makes the performance better for even a cross platform framework. Flutter can integrate open-source packages from pub.dev to get additional features in the application, which reduces development time and promotes modularity.
- **React Native:** React native is based on JavaScript, it cannot render directly using GPU and relies on bridge to communicate with GPU which results some performance overhead. React Native also has a huge community and has application libraries, which reduces the development time for implementation of complex features and simplifies the maintainability and modularity for the application.
- **.NET MAUI:** .NET MAUI is based on C#; it uses the .NET framework to create native mobile applications. This framework was created on top of Xamarin (currently deprecated), which was designed to be efficient. .NET MAUI is relatively new and has very limited community support.
- **Swift**: Swift is one of the most efficient languages for mobile development, since it is designed for only Apple's platform, which is IOS/MacOS. Compared to any other languages or frameworks it has the most access to Apples's IOS/MacOS APIs. The development is only limited to Apple's platforms; It has excellent support from Apple and a large community.
- **Kotlin:** Kotlin is a modern language created for native Android development. It provides excellent performance but is limited only to development for Android platform only. Kotlin also has a large community and a stable popularity among android developers.

3.1.4. Chosen Approach

We created a table and scored (score out of 5) our possible approaches according to our desired characteristics in the table below:

	Model Integration	Performance	Database Support	Intuitive UI Support	Modularity and Maintainability
Flutter	5	4	5	5	5
React Native	5	3	5	4	5
.NET MAUI	4	3	5	4	3
Swift	4	5	5	5	1
Kotlin	4	5	5	5	1

Flutter had some limitations in performance as it was a framework created for multiple platforms, which means not all platforms will have the peak optimization; however, in this category, only Swift and Kotlin outperformed flutter as they were only created to develop on a single platform. All the platforms provide excellent support for integration of CNN, as most of them have a library for CNN integration, but the community support for Flutter and React Native for the CNN libraries is large. This will be very useful during the application development cycle of the application. One of the major reasons we chose Flutter as the framework of choice was because of Modularity and Maintainability, as Flutter supports cross platform app development using a single codebase. This is also a feature that was offered by React Native. However, the implementation of this feature is easier to implement in Flutter compared to other frameworks. Database integration and support for an intuitive UI were the characteristics that all frameworks scored almost similarly.

From our analysis above and limited testing, team CRAFT has decided to move forward with using Flutter as the framework of choice for developing the mobile application. Flutter scored highest in our analysis as it had all the features our team desired.

3.1.5. Proving Feasibility

Team CRAFT will use Flutter to develop an application prototype that will feature a very modern and intuitive UI with very little performance overhead, keeping the codebase modular and simple to maintain. Our stretch goals for the prototype also includes implementation of database in the application.

3.2. Conveyor Belt:

3.2.1. Introduction to issue

The current application can process one photo at a time. In some cases, researchers need to process and classify mass amounts of sherds, and there is currently no way to do that in an efficient manner. A conveyor belt that could automatically classify and save sherd information would both save resources and shorten project completion times.

3.2.2. Desired Characteristics

An almost fully automatic conveyor belt system would be able to detect when a sherd has been centered under an attached webcam, process the photo on a variety of background colors, classify the sherd using the developed CNN, and save the data.

- **Performance:**
  The technology we are using must be efficient and should be able to run on a computer with less computing power without sacrificing accuracy for speed.
- **Automatic Object Detection:**
  The conveyor belt will consist of a webcam and small conveyor belt. When a sherd has passed underneath the webcam, a photo should be taken, saved, processed, and identified.
- **Photo Background Manipulation:**
  In its current state, our client's CNN model can only process photos on a white background. In some cases, like on the conveyor belt, the background will be a solid color, but not white. The program should be able to cut the photo out from a solid black background to process the sherd.
- **Community Support:**
  We would like the technology to have good community support, as this makes the development experience very efficient.

3.2.3. Alternative Approaches

- **OpenCV:**
  OpenCV is the largest open-source computer vision library, with multi-language and platform support. OpenCV is commonly utilized by both large companies and start-ups alike, including Applied Minds, VideoSurf, Google, Microsoft, and Sony. The library has over 18 million downloads and a robust community of 48 thousand developers.
- **SimpleCV:**
  SimpleCV is an open-source computer vision framework meant to abstract functions and algorithms from more complex libraries such as OpenCV. It is built in the Python programming language and runs on MacOS, Windows, and Linux systems.
- **Mahotas:**
  Mahotas is an open-source computer vision library released in 2010 that contains a variety of algorithms and functions written in C++. It has an easy-to-use Python interface, while benefiting from the speed of C++.

3.2.4. Analysis

- **OpenCV:**
  OpenCV supports most major operating systems, including mobile systems such as Android and iOS. It has interfaces for MATLAB, Java, Python, and Python, and contains over 2500 optimized algorithms. As the largest open-source library for computer vision, OpenCV also has a large, active community of developers. Due to the number of features OpenCV contains, it is more complex and difficult to implement than some other computer vision libraries.
- **SimpleCV:**
  SimpleCV does not perform as quickly as some of other alternative libraries. The development community is not currently very active, meaning that community support for the application is limited. Additionally, SimpleCV lacks consistent updates, and has not had any stable releases in recent years.
- **Mahotas:**
  Mahotas utilizes a Python interface for its algorithms, many of which are written in C++ to improve speed. With around 100 algorithms available, the library is noticeably smaller

than others on our list of alternatives. Updates to Mahotas have been slowed down in the past years, and community support for the library is generally fairly limited.

3.2.4. Chosen Approach

We created a table and scored (score out of 5) our possible approaches according to our desired characteristics in the table below:

	Performance	**Automatic Object Detection**	**Photo Background Manipulation**	**Community Support**
OpenCV	5	4	4	5
SimpleCV	3	3	3	2
Mahotas	4	3	3	3

Mahotas maintained solid performance due to its implementation of functions written in C++, but OpenCV remained the library with the best performance out of the proposed solutions. Automatic object detection and photo background manipulation were comparable among all proposed options. One of the largest differences between the different options for the conveyor belt was the level of community support available during project development. OpenCV is the largest and most widely used library currently available, and the resources available for Mahotas and SimpleCV are not nearly as robust or detailed.

Based on our analysis conducted above and the limited testing of different computer vision tools, team CRAFT has decided to use OpenCV to develop our Conveyor Belt application. OpenCV has all the desired features that our team agreed on and scored the highest across each of the listed categories compared to the alternatives.

3.2.5. Proving Feasibility

Team CRAFT will use OpenCV to develop a prototype for an application that can take photos of a sherd using a webcam which is processed to manipulate the background and then the edited photos will be saved with a metadata provided by the user.

3.3. Image Classification Model Improvement:

3.3.1. Introduction to issue

The model we currently have from our project sponsor is based on Resnet152v2, which is an older architecture for machine learning initially developed around 2016. The Resnet architecture has had updates, and studies have shown that its performance has also increased since 2016. The current model's issue is that recent advancements in other CNN models and Visual Transformers outperform the ResNet architecture.

3.3.2. Desired Characteristics

- **Accuracy:**
  The most important aspect to consider with image classification is accuracy. If the trained model is not at least as accurate as the current model, then it is useless.
- **Performance:**
  The image classification model needs to be able to run on a standard computer. If this is not the case, then the model is essentially useless because it cannot be run.

3.2.3. Alternative Approaches

- **ResNet:**
  ResNet improves upon early CNN models by allowing more convolutional layers without reducing the gradient during training by skipping convolutional layers that do not impact further layers much in early training. Skipping these layers also speeds up the training time.
- **Swin Transformer:**
  Swin Transformer uses a Transformer instead of a CNN to classify images. It outperforms all CNNs prior to its creation by a large margin and takes less computational resources to train.
- **ConvNext:**
  ConvNext is the modification of a standard ResNet model towards the Swin Transformer to close the gap between CNNs and Vision Transformers. It competes well with the Swin Transformer in terms of accuracy.

- **Fine-Tuning Hyperparameters:**
  While using a predefined model yields good results, they are designed for general use cases. Fine-tuning the hyperparameters of existing models can result in improved accuracy and training time.

3.2.4. Chosen Approach

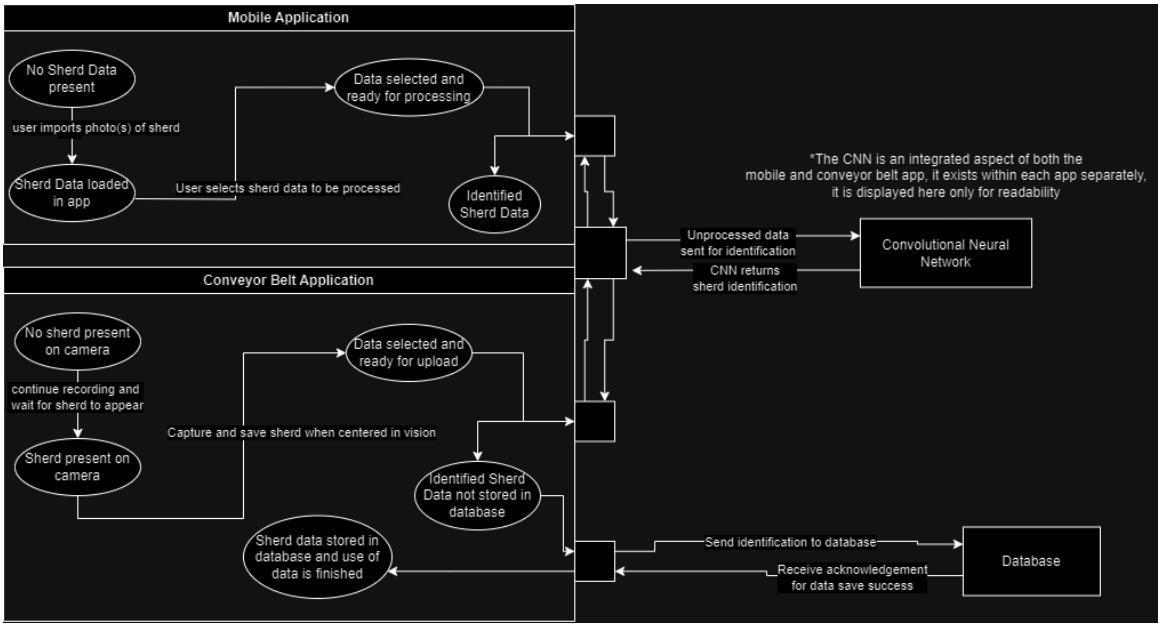
For this project, to create the best image classification model we can, our team will be training a ConvNext model and Swin Transformer model. Once the models are trained, our team will fine-tune the hyperparameters of each model and compare the results with the original models. We chose not to use ResNet because it is unable to compete well with ConvNext and Swin Transformer.

3.2.3. Proving Feasibility

Team CRAFT will train two models for sherd classification, one utilizing ConvNext, and the other Swin Transformer. After comparing the results, we will fine-tune the hyperparameters of both models and determine which of the models is the most accurate.

4. Technology Integration

The unique nature of our sponsor's requests has left us with three technological challenges that are not as directly related as one might expect in a typical development process. Instead of focusing on one software product with multiple challenges within it we have been given two smaller tasks: 1. the development of a conveyor belt, 2. The mobile application. Moreover, we have been given one large task: the refinement and progression of the image classification model.



Above is a state diagram detailing how the technological challenges we expect to face will interact with each other. Due to the unique nature of our tasks mentioned, the state diagram has three main sections: 1. the mobile application and its states, 2. the conveyor belt application and its states, 3. the CNN as a black box that will provide its function to the conveyor belt application and mobile application. Although the CNN is portrayed as a separate entity, as noted in the diagram both applications have their own version of the CNN. Despite this, the CNN is exactly the same in both instances, which is why it has been portrayed as a single entity to be interacted with.

Our mobile application's function is to identify sherd images taken in the field. Due to this, the initial state starts with no sherd images present. Upon the importing of sherd images to

be processed, the app progresses to a new state where data is present and loaded into the app. Once data is present, the user can select sherd images to be identified using the image classification model. Now our app will have to make use of the image classification model (also a part of the app) and will send data to be processed and receive the sherd data with the model's identifications. This moves us into our final state where data is finished being identified by the model.

Our conveyor belt application has a similar flow of states to that found in our mobile application. The conveyor belt application is used with a camera and a conveyor belt. Sherds along the conveyor belt are captured by the camera. Our initial state is that the camera is capturing the conveyor belt, but sherds have yet to be loaded onto it. A sherd coming into view of the camera transitions us to our next state, where a sherd is present on the camera. Capturing an image of this sherd once it is centered transitions us to our next state, where the just captured data has been saved and is ready for processing. Like the mobile application, we will send this data to the image classification model (also part of the app) for identification. Upon receiving the identified data back from the model, we are transitioned into our next state where our application has the sherd data identified, but not saved in a database. Sending our data to the database and receiving back an acknowledgment notifying us of a successful data save allows us to transition to our final state, where we are finished using the current sherd data.

5. Conclusion

Reliable, efficient sherd classification is both a scientific and personal issue. While excessive amounts of labor and money are spent on methods that often offer no consistency, communities continue to wait for pieces of their history to be returned to them.

Team CRAFT's plan involves training a ConvNext model and Swin Transformer model for classifying sherds. Once this is completed, we will tune the hyperparameters of each model and compare the end results with the original models to determine the most accurate image classification model.

Once an improved model has been built, an improved mobile application will be developed in Flutter, creating a quick, easy-to-use way for archeologists to identify sherd types while in the field. The application, which will be available on iOS and Android, will save data associated with each sherd while giving archeologists complete control through the ability to edit each piece of data. Using a conveyor belt system that implements the improved model, archeologists can catalog and classify mass amounts of sherds in a shorter time, saving valuable resources like money and labor. OpenCV will be utilized to separate sherds from their background, where they will eventually be run through the developed model.

Team CRAFT aims to build upon Dr. Pawlowicz far more consistent method for sherd classification using artificial intelligence. An improved classification model alongside other technology will provide archeologists with powerful tools to classify these artifacts, helping advance our knowledge of the ancient world.