CS Capstone Design Document

December 1, 2017

# Remote Seed Identification

Prepared for

OSU Seed Lab

Daniel Curry

Prepared by

Group 12

Ethan Takla

Alex Ruef

Quanah Green

**Abstract**

This document describes the software design of the Remote Seed Identification Project, following the IEEE Std 1016-2009 standard. The document covers multiple components of the project, including the mobile application, a deep learning image classifier, and web server. The Goal of this project is to speed up the incredibly slow Seed Identification process at Oregon State.

## Contents

## 1 Introduction

### 1.1 Purpose

The purpose of this document is to give readers an in-depth look at all of the necessary components required to achieve the desired functionality defined in the requirements document. The document will look at each component of the project from multiple viewpoints in an effort to provide the most comprehensive design as possible.

### 1.2 Scope

Currently, the seed identification process at Oregon State is done by hand and is quite inefficient. Seed analysts must look at every individual seed in a sample using a conveyor belt and a microscope, which can be tedious as the analysts can look at up to 25,000 seeds a day. In addition to working in the lab, researchers will often require seed samples to be taken in the field. Unfortunately, the only way to do this is to send the seeds to a third party company which can take days and cost a fair amount of money. In an effort to make the seed identification project more efficient, the Remote seed Identification project aims to make analysis possible to any individual with a smart phone or tablet.

### 1.3 Intended Audience

The intended audience of this document are stakeholders that plan on using the Remote Seed Identification application to speed up the seed identification at Oregon State. In addition to this, this document is also intended as a reference for Capstone instructors to view the current progress and direction that the project is going.

### 1.4 Conformance

This document conforms to all of the requirements given by Daniel Curry of the Oregon State Seed Lab.

## 2 Glossary

- **Jetson TX2:** Embedded AI computing device
- **OpenCV** Open Source Computer Vision Library
- **Caffe:** An artificial intelligence toolkit
- **SQLite:** Relation database management system
- **API:** Application Programming Interface
- **Xmarian:** Cross-platfrom mobile application development software

# 3 Mobile Application Functionality and User Interface

## 3.1 Overview

This section will go over the target functionality of the Remote Seed Identification mobile application. The application will be what seed analysts interface with to send sample images and receive result reports.

## 3.2 Design Viewpoints

### *3.2.1 Viewpoint 1: Context*

Any person looking to understand the functionality of the Remote Seed Identification application will have to have a good understanding of the C# language and the Android application development suite. In addition to this, anyone looking to modify the user interface itself will have to have experience with the Xamarian UI framework. The code developed for the application will be designed to allow future individuals to make changes and build on the work already done. This means that all of the code designed for the application will be fully commented and modular.

### *3.2.2 Viewpoint 2: Interface*

When the app is ran for the first time the user will have to log in or create a new account. These accounts will be used to track who is sending seed samples so the results can be sent back to the right person. The fields required to create a account will be name, email, and password. Communication with the server will be done using the Volley library to send HTTP requests to an API on the server. Once users have successfully logged in they will be shown a home or a navigation page. If a user has logged in previously the navigation page will be the first page they see. This page will be used to send users to either the send seed sample section or the section where they can see results. There will also be a button to a help page to tell users how to use the app and a little about what it does. There will also be an options button should we need it.

### *3.2.3 Viewpoint 3: Structure*

The Remote Seed Application will be very user intuitive and easy to use. Continuing this, the code itself will be structured in a way that is modular and easy to understand. Because C# is an object-oriented programming language, the code itself will take full advantage of inheritance and polymorhpism in order to make it as efficient as possible. In addition to this, calls to the Jetson TX2 for seed processing will be made easily accessible by modular API calls.

### *3.2.4 Viewpoint 4: Interaction*

There will likely be multiple images for a single sample. Users will first have input how many images will be sent in the sample. Once that is done the user needs to present an image to be sent. Selecting an image can be done from the users images or directly from the phones camera. The images will be sent by HTTP requests to our server which will then be processed by the Jetson TX2. After the images are sent the user will see a message informing them if the image upload was successful or not. Users can be sent a notification and/or an email when the results are done processing. If processing is quick enough users will be send to the results page after uploading a sample. Users will be able to see a list of previous samples sorted by date with the most recent at the forefront. The user must select the sample they want before the results are displayed. Sample output display will depend on what is contained in the results. Users should also receive a PDF of the results in an email.

# 4 Web Server, Database, and Report Generation

## 4.1 Overview

This section discusses the design of the web server and database, as well as the format of the final report sent back to the client. A web server will be running indefinitely on the Jetson TX2 processor, waiting to receive sample processing requests from the clients. Report results will be archived within the database for later analysis if desired.

## 4.2 Design Viewpoints

### *4.2.1 Viewpoint 1: Context*

This viewpoint mainly deals with how the application interfaces with the back end Jetson TX2 processor over the internet, and additionally the way in which analysis reports are sent back to the client application. Any individual looking to work with the database or web server will need to know SQLite and be familiar with python web servers.

### *4.2.2 Viewpoint 2: Interface*

The two entities involved with web-based interactions will be the client application and the Jetson TX2 processor. A server will be needed to facilitate transfer of images and reports between the mobile app and the Jetson processor where the actual analysis will happen. In order to reduce costs and simplify the design, the server will be run on the Jetson itself. A dedicated server could be implemented at a later time, but until the core functionality of the seed identification process has been shown effective there is no reason to spend time and money on a dedicated server. A database will need to be maintained that includes login information and addresses for sending back reports. The database will need be accessed for validating user login information. Furthermore, each time a logged in client establishes a connection to the server, the database will be updated with their most recent IP address. Once a job is processed and a report is generated, the address of the client who requested the analysis will be looked up in the database, and the report will be sent to that address. The database will be implemented in the Jetson application using SQLite. This will be accessed in the server program using the PonyORM python package. This will allow for easy database access from the python based server.

### *4.2.3 Viewpoint 3: Structure*

When an analysis is run, the results need to be recorded and conveyed in an easily readable format. Once the analysis is completed on the Jetson, the results will be passed as numerical arguments to a function in the server interface, which will use the open source ReportLab python library to generate a PDF report, to be sent to the client. This report will contain data on the homogeneity of the sample, and what species were identified. It will also specify the number of seeds identified, and call to attention any seeds that could not be positively identified.

### *4.2.4 Viewpoint 4: Interaction*

The server will be implemented in Python 3 using the http.server package. As we will only be serving simple requests, the built in python http.server package will be sufficient to create a functional server fairly easily. When the user logs in through the mobile app a connection will be opened to the server, which will validate their log in information in the database, and upon successful login update their most recent IP address in the database. When the user decides to upload images to create a report, a connection to the Server will again be opened. The images will be sent to the Jetson over the open connection, and passed directly in to the seed identification API. Once the analysis is finished, a connection will be opened back to the client at the IP address stored in the database, and a report will be sent back to them.

## 5 Deep Learning Image Classification

### 5.1 Overview

This section is dedicated to discussing the design of the image classifier used to identify seeds in a sample. Image classification can be split into two main parts: classifier training and seed identification/segmentation. In the case of grass seeds, the image classifiers will need to be trained with high accuracy in mind, as many seeds look nearly identical with the exception of a few small features.

#### *5.1.1 Viewpoint 1: Context*

Any individual looking to train the image classifier or modify it in any way will have to be familiar with Linux, the deep learning Caffe framework, Google's Inception-V3 neural networks, OpenCV, and C++. This part of the project is by far the most technically demanding, is it uses cutting-edge technology that has only recently been developed.

#### *5.1.2 Viewpoint 2: Interface*

Clients will not be directly interfacing with the image classifier, but rather indirectly through the mobile application. Due to the fact that the Jetson TX2 is very resource limited, Google's Inception-V3 neural networks will be used for the image classifiers. This network is known both for its speed and accuracy when it comes to classification problems. It uses up to 12 times less parameters than other leading network architectures such as AlexNet. Before diving deep into implementing this on the Jetson, NVIDIA has a graphical software called DetectNET that acts as a sandbox for testing training networks with large sets of data. This will become very useful in determining the quality and quantity of training data that we need in a timely manner. Before sample images are sent to the Jetson from the client, they must be pre-processed to account for various factors such as light conditions and orientation. OpenCV provides an excellent interface for doing just that. Once the networks have been properly trained, a second step is needed, which is responsible for localizing each individual seed in an image. This will be done using R-CNNs, which finds regions of the image that have high probability of containing seeds.

#### *5.1.3 Viewpoint 3: Structure*

The code developed for the Jetson will be written in C++ for speed purposes, and will be fully modular so that future developers can add to the image classifier if need be. Nearly all of the API calls will be through OpenCV or Caffe, which makes the whole structure of the back-end easier to understand due to the vast amount of documentation available for their respective API calls.

#### *5.1.4 Viewpoint 4: Interaction*

Once a request has been received from the web server, the Jetson will process it in a first-come-first-serve manner. In the case that the image pre-processing step takes a large chunk of time, it will send chronic updates of the progress to the user so that they know what is going on at all times. From a client standpoint, there will be no direct interaction with the Jetson TX2.