Oregon State University
College of Engineering

CS Capstone Final Progress Report

June 12, 2018

CDK Data Stream AI

Prepared for

CDK Global

Chris Smith

Prepared by

Group 65

SigFind

Jacob Geddings

Inhyuk Lee

Juan Mugica

Abstract

CONTENTS

1 Introduction to Project

1.1 Who is our client

CDK Global currently operates with thousands of different dealerships across the world each of which using their own unique forms, servers, and verication methods for each transaction. The company is interested in providing new forward-thinking solutions to the various logistical problems that surface as a result of permitting such liberties within their various branches. Our direct correspondent and client from CDK Global is Chris Smith, who went on to provide an advisory role for the project as it progressed through the development lifecycle.

1.2 Defininf the problem

As mentioned, the current system has allowed a great level of liberty for each dealership but has made it a logistical nightmare to quickly coordinate documents and to ensure everything submitted is actually valid. In particular, their main concern is nding a way to quickly verify if a form has been signed. Beyond that core issue they are also interested in reigning in their various servers so that this verication process is done under one roof and with a singular database. To achieve this task, CDK Global does have a few ground rules regarding the development of any potential solutions. They are interested in an open-source design with no licensing requirements and to have it modular in nature to ensure upgrades can be carried out as needed should our solution be implemented.

1.3 Why this is important

The current model being used to tackle this problem is to have dedicated employees devote their entire workday to the manual scanning and verification of documents. This process is not only sluggish in nature but very accident prone with an extreme dependency on human error not occurring. Not only that but this occurs only after the fact when considering potential transactions. A worst-case scenario could be an auto dealership releasing a car to a potential buyer to only realize there was no legally binding signature proving the purchase of the vehicle. By implementing AI assistance, we can permit quick feedback on submitted documents as well as significantly reduce the cost to the company in verification by mitigating errors and reducing manpower necessary to being devoted to this fact-checking process.

1.4 Our team

Our team, Group 65, was created to tackle this problem. Within our team is three members: Jacob Geddings, Inhyuk Lee, and Juan Mugica. This team was assembled based on the common interest of wanting to explore what goes into AI development and found this project to be a perfect opportunity to work with these new concepts. While we ultimately named our project SigFind we also would use the term to refer to our group instead of the generic number classification. Overall, our group was expected to spend the Fall term working on research and documentation for the project with Winter representing the coding phase and Spring acting as our debugging and finalization of the project for code handoff. Jacob Geddings would function as team leader for the duration of this project. Most notable contributions can be found in documentation, client communication, and organization of the project. This meant that Fall and Spring was

particularly workload heavy for Geddings whereas Winter was a comparatively lighter workload for dedicated tasks. During the coding phase of Winter term, he contributed towards our file conversion module as his dedicated portion of the project. Given that the file conversion was a quick implementation he would spend the remainder of coding working on providing assistance with OCR Tesseract as well as constructing training sets for our Convolutional Network. Inhyuk Lee functioned as second in command for the team and, when able, provided a second set of notes on most interactions within the group and with our client. His primary task was to develop the OCR Tesseract component of our project. This meant creating a program that could correctly analyze any arbitrary page and locate potential signature lines. Once that could be detected he was then tasked with cropping those signature lines out to submit to our convolutional network. Lastly, Lee also provided assistance in submitting samples for the training set of our program. Juan Mugica was our dedicated convolutional network researcher and implementer. Due to the importance of this module it was deemed best to have at least one member be completely focused on this component and not be concerned with the other modules as much. During the Fall term, Juan explored potential AI platforms for our network. From then on his entire focus was towards the correct creation of the network and ensuring proper judgement capacity of the program when interpreting signature lines.

2 Requirmentrs Document

2.1 Introduction

2.1.1 Purpose

The purpose of this requirements document is to bring clarity to the project by indicating what the hard-line requirements are for the project to be considered a success. This will be done with the consideration that the desired end product may not be achievable and as a result will be treated as a form of research project or proof of concept.

Intended target audience for this document is our client and by extension CDK[1] given this is their idea and the success of the project directly assists them. In addition, our instructors and teaching assistant are extensively involved in this project and they will be using this as a grading guideline. The assumption is that the reader is familiar with general concepts within computer science including artificial intelligence, common programming languages, and coding environments.

2.1.2 Scope

The product being developed will be a variant of open source AI that will be tailored to scanning submitted forms by CDK. With the inclusion of stretch goals this platform may also analyze driver licenses, vehicle imaging, or name to vehicle identification.

This project will be capable of analyzing a submitted form to locate its signature box and check to see if it has been signed. It will also prompt the operator if it is not certain of what it is observing and if a cosigner box is also present. The format of these submissions will be in PDF format. Stretch goals include the capacity to detect expiration dates on

drivers licenses and determine what make and model of a vehicle is based on a submitted image. Additional file formats are also being considered.

Benefits from this product include reduced costs for CDK in error checking forms, rapid validation of a document, and a singular platform that can scan the various form styles that CDK possesses. Another goal of this design is the ability for it to expand upon itself and adapt to new forms that may be introduced by permitting operator confirmation or override on its assessment.

2.1.3 Definitions, Acronyms, and Abbreviations

Open source AI is a free to access platform in which the code can be used and implemented as seen fit by the operator. OpenCV[2] is a popular image processing example of this as well as TensorFlow[3] and DL4j[4]. Programming languages that may be used include C++[5], Python[6], and Java[7]. The use of an IDE is likely, which means Integrated Development Environment which provides a more convenient setup for coding. Examples of IDEs include Microsoft Visual Studio[8] and Eclipse[9]. Black box design means that the user does not need to understand the internal functions of the program, just what it wants as input and what the user will receive as the output.

2.1.4 References

[1] ”CDK Global” Internet: https://www.cdkglobal.com/, 2017 [Nov. 10, 2017].

[2] ”OpenCV” Internet: https://opencv.org/, Aug. 3, 2017 [Nov. 11, 2017].

[3] ”TensorFlow” Internet: https://www.tensorflow.org/, [Nov. 12, 2017].

[4] ”Deeplearning4j” Internet: https://deeplearning4j.org/, 2017 [Nov. 11, 2017].

[5] ”C++” Internet: http://www.cplusplus.com/, 2017 [Nov. 12, 2017].

[6] ”Python” Internet: https://www.python.org/, Oct. 17, 2017 [Nov. 12, 2017].

[7] ”Java” Internet: https://java.com/en/, Apr. 22, 2016 [Nov. 12, 2017].

[8] ”Microsoft Visual Studio” Internet: https://www.visualstudio.com/, Oct. 2016 [Nov. 10, 2017].

[9] ”Eclipse” Internet: https://www.eclipse.org/ide/, 2017 [Nov. 9, 2017].

[10] IEEE Software Engineering Standards Committee, IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications, June 25, 1998.

[11] ”GCC” Internet: https://gcc.gnu.org/, Nov. 3, 2017 [Nov. 9, 2017].

[12] ”CMake” Internet: https://cmake.org/, Nov. 10, 2017 [Nov. 9, 2017].

2.1.5 Overview

The remainder of the requirements document will contain a greater look into what precisely the project entails. This is going to involve looking at where the project stands for CDK and how it will be utilized as well as the products functions and characteristics for a user. Mention will also be made towards potential constraints placed upon the program.

The following section will be a general overview and description of the project, primary focus being placed on functions, characteristics, constraints, and assumptions/dependencies. After that there will be a section on specific requirements that will bring mention to potential interfacing concerns as well as performance metrics and standards compliance. The document is designed to follow the IEEE Software Engineering Standards [10].

2.2 Overall Description

2.2.1 Product Perspective

The core objective as well as the stretch goals all conclude that this product will be an independent platform. For this project, we will explore the potential of A.I. and create the foundation of CDKs own AI platform.

The signature detection and drivers license verification software must support various personal computers such as desktop, laptop, and tablet as an input source. This software will first receive a PDF file of a document as its input. It will then compute the data and give result or feedback to user. With that in mind, the software needs to be robust enough to compute large set of data. Similarly, within our stretch goals, vehicle model detection software must also support various personal computers and be robust enough to evaluate large data. However, it must also support portable devices with camera such as phone and tablet. Software should first take photo from camera then it evaluates the photo and returns model number.

2.2.2 Product Functions

Main goal The function of this software is detecting the location of the signature box and determine whether it is signed or unsigned. If document is unsigned: software must provide information of location and quantity of missing components. Also, the program must be capable of making a reasonable guess on whether a signature is sufficiently binding. This entails the AI must be capable of distinguishing between a simple mark in the signature window and an actual signature.

Stretch goal License validation software: This software must validate whether given photo is drivers license or not. First, it must validate the format of drivers license and check legal information such as expiration date.

Vehicle model detection software: Vehicle model detection software must be able to determine the specific model number from appearance of the vehicle. Software will first take photos or video of vehicle taken from different angle. It will determine vehicle model by narrowing down the information. For example, software will search sequentially for make, model, and year.

2.2.3 User Characteristics

The intended user of the software is vehicle retailers. We can assume a low degree of understanding on artificial intelligence. Therefore, software must be designed as black box model. To accompany this structure the user interface must be agreeable for them. Lastly, the program must not overburden the operator, it will take a single file as input to scan and return a conclusion to the user.

2.2.4 Constraints

There are few constraints on CDK A.I. data stream project. First, software must be standalone. The future goal of this project is to build CDKs own platform, so software must be independent. In addition, project must be kept in open source. The software must be made in such a way that, once our team has completed its work, the project be taken over by a new team within CDK. To meet this demand the need for thorough documentation of implementation is vital. This project will also have limits on testing due to the privacy issue surrounding actual consumer data. Since the data that signature detecting and license validation software compute contains personal data, testing data must be done on dummy documents and licenses. The last constraint is that product must function on its own without the additional use or maintenance of outside software. Since the users of the product are unlikely to be familiar with the internal workings the interface for them must sufficiently mask the technical workings of the program only provide a clean and simple menu.

2.2.5 Assumptions and Dependencies

All images provided by the client are sanitized and free of legal conflicts. Cloud based deployment services will be provided by the client. From a developing and testing standpoint, this project will be developped in C++ utilizing the OpenCV picture formatting library. This library assumes that the system is running GCC 4.4.x[11] or later as well as CMake 2.8.7[12] or higher.

2.3 Specific Requirements

2.3.1 External Interface Requirements

The focus of this project is building signature detecting artificial intelligence run on the CDK server. It must be designed as black box model, so user interface must be simple . Since software runs on the CDK server, communication interface must be secure and durable.

The structure of user interface is very simple. There are two functions in user interface. Software simply takes set of documents from the user, and it displays result of the signature detection to the user. Result of the signature will be any pages of the document where error is detected.

Communication between our software and the CDK server database is important; however, focus on this project is making accurate and reliable artificial intelligence. We are considering the use of docker and amazon web services to construct a reliable communication structure but this is a stretch goal of the project.

2.3.2 System Features

2.3.2.1 Signature detecting capability:

Introduction/Purpose of feature: The program must be able to detect if a document contains a signature regardless of the documents formatting.
Stimulus/Response sequence: The program will parse a PDF image of a document, or a document in PDF format, and output a percentage based estimation as to whether it contains a signature.
Functional Requirements: Signature detection must be functional for all types of document formats regardless of the type so long as they are presented in PDF file format.

2.3.2.2 Missing signature detection:

Introduction/Purpose of feature: The program must be able to detect if a document contains a space where there should be a signature regardless of the format of the form.
Stimulus/Response sequence: The program will parse a PDF image of a document , or a document in PDF format, and output a percentage based estimation as to whether it contains a space where a signature is missing.
Functional Requirements: Signature detection must be functional for all types of document formats regardless of the type so long as they are presented in PDF file format.

2.3.2.3 Multiple signature and multiple missing signature capability:

Introduction/Purpose of feature: The program must be able to perform feature 3.2.1 for every signature contained within the document. The program must also be able to perform feature 3.2.2 for every space that is missing a signature within the document.
Stimulus/Response sequence: The program will parse a PDF image of a document, or a document in PDF format, and output percent estimates for features 3.2.1 and 3.2.2 at every instance deemed applicable.
Functional Requirements: Multiple signature and multiple missing signature capabilities must be functional for all types of document formats regardless of the type so long as they are presented in PDF file format.

The following features are part of the groups stretch goals and are to be completed once the aforementioned features work to the satisfaction of the client.

2.3.2.4 Car detection within image:

Introduction/Purpose of feature: The program must detect if a given picture contains a car within it.
Stimulus/Response sequence: The program will parse a PDF image and output a percentage based estimate as to whether it contains a car.
Functional Requirements: The program must be able to detect cars within any picture presented in PDF format.

2.3.2.5 License and Permit detection within document:

Introduction/Purpose of feature: The program must detect if a document contains a license or a permit regardless of the format of the form.
Stimulus/Response sequence: The program will parse a PDF image of a document , or a document in PDF format, and output a percentage based estimation as to whether it contains a license, or a permit.
Functional Requirements: The program must be able to detect permits and licenses within any document presented in PDF format.

2.3.2.6 Parse license plate, license, and permit text into machine readable format:

Introduction/Purpose of feature: Once the program can perform features 3.2.3 and 3.2.4, it must then parse the contents of these forms into machine readable format. This will allow the user to utilize this information to validate a clients identity, ties to the vehicle, etc.. For this task, we will utilize Googles Vision API.
Stimulus/Response sequence: The program will parse a license, permit, or license plate image in PDF format and extract the information presented into a machine readable format.
Functional Requirements: The program must be able to read information within any license plate, license, or permit regardless of the origin so long as they are presented in a PDF format.

2.3.2.7 Damage recognition:

Introduction/Purpose of feature: The program must be able to parse a picture of a vehicle and detect whether this vehicle has sustained damage. Often CDK employees must go through pictures of vehicles in order to assess if they have been damaged. This feature aims at discarding all vehicles with no damage lessening the amount of pictures that must be assessed for damage.
Stimulus/Response sequence: The program will parse a picture of a car in PDF format and output a percentage estimate as to whether the car has been damaged.
Functional Requirements: The program must be able to detect damage within any picture of a car so long as it is submitted in PDF format.

2.3.2.8 Make and Model detection:

Introduction/Purpose of feature: The program will parse the picture of a car and attempt to discern the make and the model.

Stimulus/Response sequence: The program will parse an image of car in PDF format and output a number of percentages assessing what the possible make and model combination might be for the vehicle.

Functional Requirements: The program must be able to detect the make and model of a vehicle regardless of the make and model so long as the picture is of a car, and is submitted in PDF format.

2.3.3 Performance requirements

This program will utilize deep learning algorithms to achieve all features except for 3.2.5. To do so, the program must run continuously for extended periods of time in order to ensure optimal learning time frames. The program must be able to catch errors and continue running unless the error is fatal. Segmentation faults, memory depletion and hardware errors are all considered fatal, all other errors must be caught, handled, and notified to the user.

2.3.4 Design Constraints

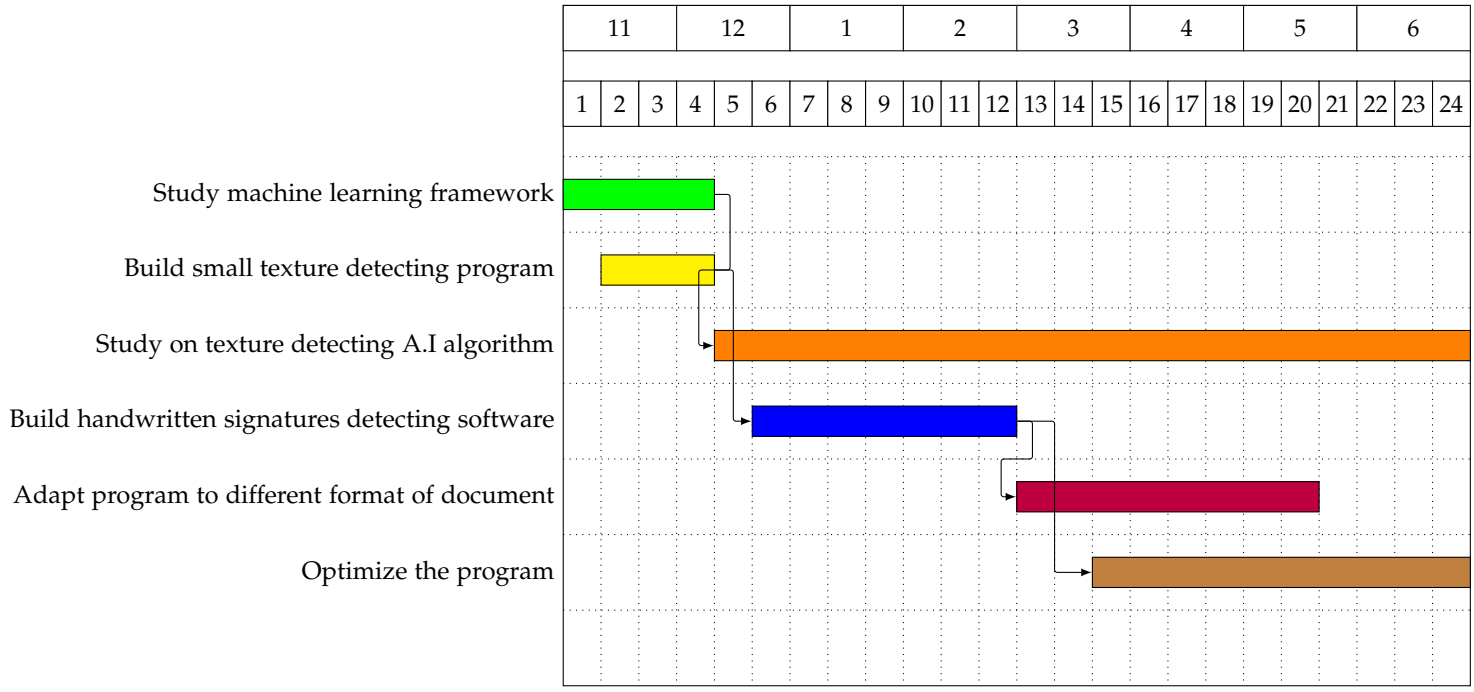
The program is free of design constraints so long as the interface ensures ease of access and interaction between the different components for all end users. This pertains to the ease of submission of pictures and documents to the software, as well as the presentation of all results in a readable and comprehensive manner.

2.3.5 Software system attributes

The program will require a platform in which to run uninterrupted for extended periods of time. The platform must be accessible to ensure that modification can be made whenever deemed necessary. The platform must be portable to ensure that the software can be deployed in a wide range of environments. Finally, the platform must be able to support heavy usage of memory resources to ensure the software does not encounter depletion of memory issues.

2.4 Appendix

This section details the groups schedule in order to complete the tasks per the school year of 2017. Included is a Gantt chart detailing what the specific task is and the time frame over which they're scheduled to take place.



Practice

- Study machine learning framework
- Build Small texture detecting program

Main: Build first prototype

- Study on texture detecting A.I. algorithm
- Build program that detects handwritten signatures from document
- Adapt program to accept ranging document formats
  - Data feeding
  - Check License information
  - Check permit information
- Optimize the program
  - Debugging

Stretch Goals (to be scheduled after completion of original goals)

- Detect the model of a vehicle

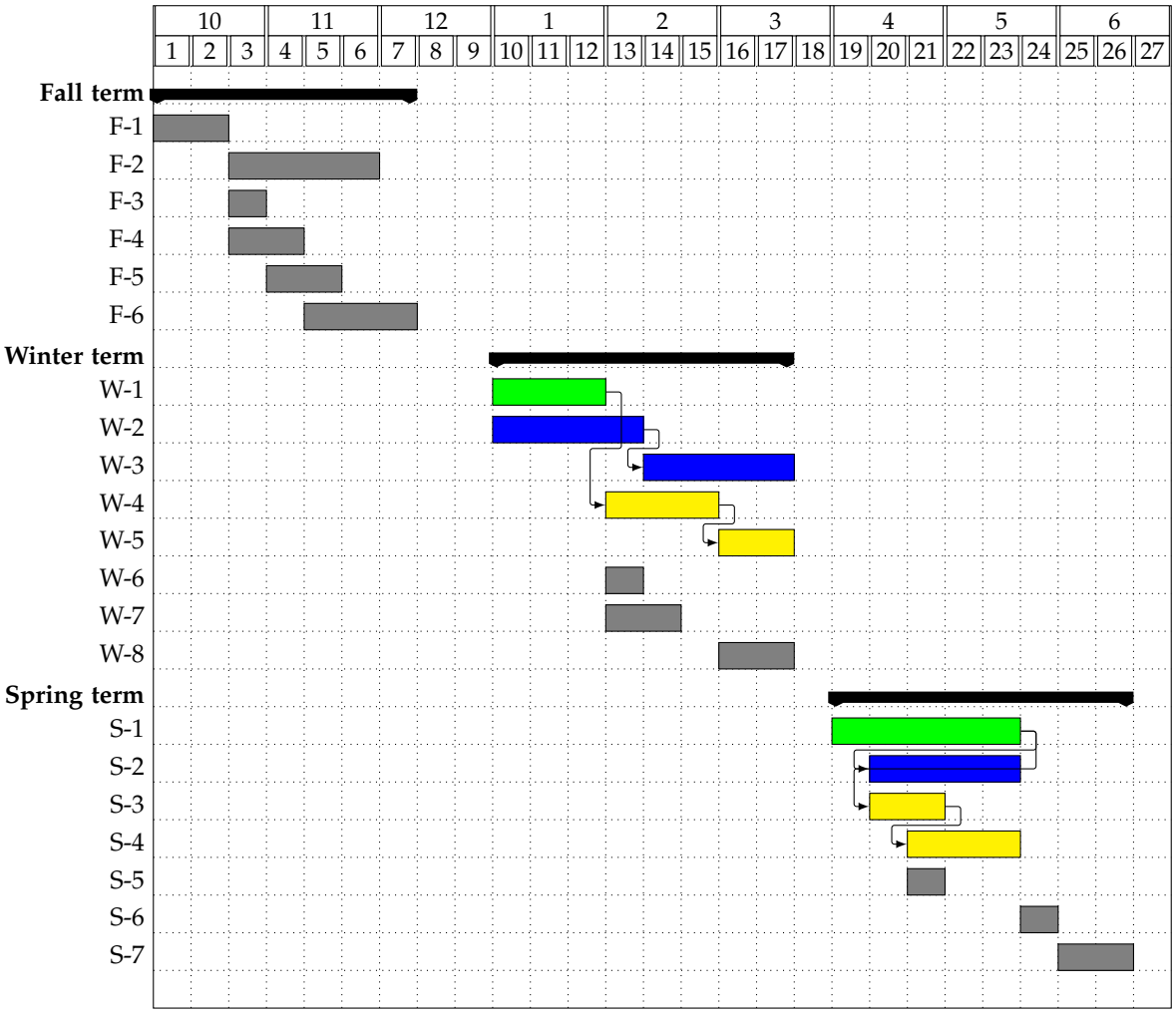
- Detect the make of a vehicle
- Detect/read license plate information
- Detect/read license and permit information
- Gather information of the condition of the vehicle (e.g.: dents, cracks and scratches)

2.5 Change of Requirments

There are no requirment changes made.

2.6 Final Gantt Chart

The following shows progresses on our project.



Fall term

1 Problem statement document

2 Research on Image learning API, and handwritten recognition algorithm

3 Formating github page
4 Requirments document
5 Technology review
6 Design document and Progress report

Winter term

1 OpenCV and Tesseract OCR setup
2 Convolutional Neural Network: Implimenting convolutional neural network
3 Convolutional Neural Network: Improving and training convolutional neural network
4 Tessract OCR: Locating the signature from the image
5 Tessract OCR: Cropping the signature box based on sign line
6 Poster design
7 Mid-progress report
8 Progress report

Spring term

1 Testing and debugging
2 Convolutional Neural Network: Debugging signature recognition algorithm
3 Tessract OCR: Debugging on line detection
4 Tessract OCR: Debugging on page segmentation
5 Final poster
6 Expo
7 Final report and video

3 Design Document

3.1 Overview

3.1.1 Scope

This document will cover nine pieces of the project including our chosen technology to accomplish that task and projected plans regarding that technology. This will include plans for if the technology fails to be what we anticipated as well as plans of dropping the piece down the line should it no longer be relevant to our needs. These pieces will each be discussed from a defined perspective or viewpoint that best defines the technology in question. Some examples of viewpoints include a dependency viewpoint, algorithmic viewpoint, and composition viewpoint.

3.1.2 Purpose and Background

Planning is vital to the success of this project. Each component should be considered and have backups in place should our predictions go awry. As such, this design document helps illustrate our projected plans and security precautions. This will help give clarity for our client as well as ourselves in seeing how we intend to progress and gives us clear

markers for when we have achieved a given task. While this document will not exhaustively cover every edge case, it does touch upon nine vital areas our group has been concerned about for completing the project.

3.1.3 Intended Audience

The primary audience for this document will be our instructors and client who have moderate to high levels of expertise within this field. This document will help in giving clarity regarding our intended actions in the following months as we enter the coding and development phase of the project. In addition, this document will operate as a checklist within our own group to ensure we are staying consistent with our documented plans and with what we actually produced.

3.1.4 Glossary

[1] "OpenCV Tutorial C++" Internet: https://opencv-srf.blogspot.com, [Nov. 21, 2017].

[2] "LearnOpenCV" Internet: https://www.learnopencv.com, Oct. 30, 2015 [Nov. 14, 2017].

[3] "CDK Global" Internet: https://www.cdkglobal.com/, 2017 [Nov. 10, 2017].

[4] "Python" Internet: https://www.python.org/, Oct. 17, 2017 [Nov. 12, 2017].

[5] "Smoothing Images" Internet: https://docs.opencv.org, [Nov. 21, 2017].

[6] "OpenCV Tutorial C++" Internet: https://opencv-srf.blogspot.com, [Nov. 21, 2017].

[7] "Filters B" Internet: http://www.bogotobogo.com, 2016 [Nov. 20, 2017].

[8] "Image Magick" Internet: https://www.imagemagick.org, 2017 [Nov. 21, 2017].

[9] "Ghostscript" Internet: https://www.ghostscript.com, 2016 [Nov. 20, 2017].

[10] "MuPDF" Internet: https://mupdf.com/, 2017 [Nov. 21, 2017].

[11] "OCR Optical Character Recognition Internet: http://www.recogniform.net/eng/ocr-optical-character-recognition.html, 2016 [Dec. 1, 2017].

[12] "System software: User interfaces" Internet: https://en.wikibooks.org/wiki/A-levelComputing/CIE/Computersystems,commu 2017 [Dec. 1, 2017].

3.2 Design Viewpoints

3.2.1 Introduction

The viewpoints model of depicting our various pieces for this project gives the opportunity to plan for that piece in regards to what role it must serve. The various viewpoints depicted below show what the intended function for that technology is for the group and how it will be used to serve the project as a whole. Each viewpoint carries its own distinct subset of rules as well to help clarify the particular function it needs to carry out.

3.2.2 Context Viewpoints

3.2.2.1 Design Concerns: Each piece of technology is outlined in its purpose as well as its potential shortcomings. Depending on the viewpoint being utilized for that given piece, the exact specifications of this section vary, but the core of it remains the same. We present the problem, the potential restrictions that exist, and our intended technology to remedy this while not violating the restrictions in place. In addition, it will include alternatives should our predictions be wrong.

3.2.2.2 Design Elements: In this section of a provided viewpoint the technology in question will be described in greater detail. Indications of what strengths or weaknesses an item may have are described here as well as the alternative technologies abilities. Beyond this portion, depending on the viewpoint being used there are additional potential subsections that may follow.

3.2.3 Languages of OpenCV(Composition)

3.2.3.1 Design Concerns: The largest concern in this design is the fact that OpenCV is primarily restricted to only a few languages. C++, Python, and Java are the three candidates worth considering, and there are significant differences between them. When choosing the language for our project, we have to consider two major factors: AI is a computationally heavy process, and we need to primarily prove that the task can be done (not necessarily in the most efficient manner). After consideration of these factors, we decided to go with C++ as our primary focus given its documentation and optimization [1], but a backup plan has been made for Python migration.

3.2.3.2 Design Elements: Design entities:
The primary entity is the language itself, C++, with its supported libraries as an extension of that. C++ is the leading language in extensive library support which helped lead to its design choice. Python is the backup here with a weaker overall support, but it maintains a competent package for AI usage [2]. In either case the environment for working on the project will not change; both function in the same space and will not require special equipment to complete the coding.

Design relationships:
The language will define how the entire project proceeds. It will be involved in all components of our project. Peripherals to this project such as Docker support and cloud storage are considered low priority; CDK can easily handle those portions on their own [3]. The program will be designed with modularity in mind, however should this program be expanded upon, we will need to reduce the difficulties involved with accomplishing that task.

Design attributes:
C++ operates in a predictable, expected manner. Maintaining an up-to-date grasp on its functionality is relatively difficult due to its memory allocation and syntax requirements, but it does result in some of the best performances at runtime. Python (as a backup) does carry some differences from what might be traditionally expected. For OpenCV to

function properly on Python, it will utilize C++ significantly just out of view of the programmer through the use of library packages like numpy [4].

3.2.3.3 Data Attributes: The internal functions of C++ are very flexible by nature despite the syntactic difficulties of writing in it. A fair bit of this program will be dynamic in nature; the AI will be learning from a constantly changing database of images and as such must be flexible in its capacity to interpret the data. There will be some static elements to this program as well, including image conversion and image filtering. All received files will be in PDF format and will be text documents that will need to be scanned. This allows for the aforementioned components to remain static.

In the event Python is incorporated into the project, presumably due to complications with C++, the attributes will be mostly the same as well. The process will be identical, but the functions used will have to be different. If this direction is taken with the project there will be a notable slowdown in performance regarding all aspects of our project due to the nature of Python.

3.2.4 Image Filtering(Algorithm)

3.2.4.1 Design Concerns: The leading concern for this project is whether the task assigned to us can be completed. As such, the focus for image filtering is placed on how effective it is at assisting the program in finding the signature windows. It is not a primary concern at present to have the most optimal algorithm used for the smoothing of our images, just one that will be most likely to succeed. Should we be incorrect in our assumptions for what would work best, there are another two functions that could be used as substitute. Should the project be proven possible, consideration will be made for using more efficient functions without losing accuracy on program readouts. Median will be the second choice due to the fact that it carries slightly stronger properties at shape retention through the blurring process. If both preceding options fail then Gaussian is unlikely to improve the situation but will be attempted regardless [5].

3.2.4.2 Design Elements: For this portion of the project we are interested in improving readability for the AI in the event the submitted images are filled with too much noise and it cannot correctly interpret what is being fed to it. To accomplish this, there exist several different functions that seek to smooth out a provided image at varying computational costs. The leading choice for our needs is bilateral filtering, a method that retains edges very well while blurring everything else. Fallback options in the event of failure include median and Gaussian which are cheaper to run on a system but provide a significantly stronger blur that can disrupt the original shape of an object within the image. Median still makes an effort to preserve object form while removing static-like noise in an image, but it doesnt make shape retention a high priority. Gaussian by itself is a very harsh method of blurring for our purposes in that it does not care about original form of the image; it effectively passes a thick brush across the image and that is it [6].

3.2.4.3 Processing Attribute: Bilateral filtering can effectively remove noise whilst retaining the object within the image. It is a costly function call however; bilateral filtering functions via calling two unique gaussian filters and designating one to sweep through the space of the image and search for sharp changes in pixel intensity which indicates the edge of an object. The other filter will move through the image and blur whatever does not show a steep change in pixel intensity. For this process to work in a reasonable manner, it will be implemented with the same language as the AI and be handed the image conversion of our PDFs. The output should enable the AI program to more readily see the distinct objects on the image and, most importantly, determine the locations of signature windows. In the event that bilateral

does not accomplish the task, the use of a median filter is our next best option. Median functions under a relatively new method for OpenCV in which it processes all channels of the image input independently. This process as a result can frequently blur any salt-and-pepper like noise on an image without breaking up object form [7].

3.2.5 Computer Vision Library(Composition)

3.2.5.1 Design Concerns: Choosing computer vision library is critical for designing signature detecting software. Using a robust library will improve the functionality of the software and reduce the development time. To select good quality of library, three common computer vision libraries are compared. These libraries, TensorFlow, Deep Learning 4 Java(DL4J), and OpenCV are evaluated based on: flexibility of environment and language, variety of functions, weight of the functions, and the size of the community. When considering these factors, OpenCV library was chosen for this project as the strongest candidate.

3.2.5.2 Design Elements: Design entities:
OpenCV is an Open Source Computer Vision Library that provides various functions related to digital imagery and video. Since the main goal of this project is recognizing signatures on a document, this library will be vital in correctly analyzing any submitted form. OpenCV supports many languages such as C++, C, Python and Java but it is optimized primarily for C and C++. Also, it is executable on both desktop (Windows, Linux, Android, MacOS, FreeBSD, OpenBSD) and mobile (Android, Maemo, iOS) [10].

Design relationships:
Computer vision library have high relationship with design and efficiency of the algorithm. Each computer vision library supports different functions to solve the same problem. Based on what library that software is using, design of the algorithm will change. In addition, since each library supports different functions, the performance of each functions varies. Therefore, choosing a good library will increase the performance of the entire software. The interaction between environment and library is also important. Since different libraries supports different languages and operating systems, our choice will significantly impact our options when developing the project. OpenCV has high flexibility on environment. It supports four languages and runnable on nine operating systems.

3.2.5.3 Design Attributes: Unlike other computer vision libraries, OpenCV been functioning for years with continued support. Because of this, OpenCV carries with it a large and diverse community of supporters. According to the OpenCV website, there are 47 thousand people in the user community, and more than 14 million downloads [10]. Since there is such a large community, learning OpenCV and fixing issues will likely be easier than with less documented or supported formats. OpenCV is very fast at runtime when compared to other AI platforms. This is because OpenCV library is written in C and C++. It also supports 500 improved algorithms to accelerate the original algorithms [10]. OpenCV do this by using CUDA which is modern GPU accelerators that increase GPU performance, so an algorithm can be computed in a short period of time.

3.2.6 Text Recognition Method(Algorithm)

3.2.6.1 Design Concerns: Choosing computer vision library is critical for designing signature detecting software. Using a robust library will improve the functionality of the software and reduce the development time. To select good quality of library, three common computer vision libraries are compared. These libraries, TensorFlow, Deep Learning 4 Java(DL4J), and OpenCV are evaluated based on: flexibility of environment and language, variety of functions, weight of the functions, and the size of the community. When considering these factors, OpenCV library was chosen for this project as the strongest candidate.

3.2.6.2 Design Elements: Design entities:
Intelligent Character Recognition(ICR) is a text recognition algorithm. It converts text images to electronical text documents by taking three steps. It first analyzes the image, recognizes the paragraphs, text lines, and words from the image. Then, it splits the images into individual characters. On the last step, it compares image characters with characters in the dictionary, and converts the characters [11].

Design relationships:
ICR directly impacts the performance and speed of the software. Since ICR is the algorithm that is used for detecting the signature box, the higher accuracy of the character recognition will lead to the higher accuracy for detecting the signature box. It also impacts the speed of the software. If the algorithm is heavy, speed of the software will go down. ICR algorithm does have comparatively high accuracy and high-speed when looking at its competitors.

3.2.6.3 Design Attributes: For detecting characters, ICR has three features to increase the accuracy of the function. First it compares the image character with character in dictionary. Second, it evaluates the characteristics of the character such as number of lines and angles between lines to determine which character is it. Last feature is the self-learning feature. ICR learns through its experience to increase accuracy. Also, since it can learn different font and style, it can determine handwritten texts from the image.

3.2.7 User Interface Type(Logical Viewpoint)

3.2.7.1 Design Concerns: Selecting proper user interface is important to draw full functionality of the signature detection software by user. To choose the best user interface for the software, characteristics of the user and characteristics of the software were considered. Determining who is the user and characteristics of them is the one of the most important factor. Based on their knowledge and experience, efficiency of the user interface will change dramatically. Also, software characteristics such as input constraints and hardware constraints also influence user interface. Based on the two characteristics, menu driven interface has been chosen among command line and form based interface.

3.2.7.2 Design Elements: Design entities:
Menu Driven interface interacts with user by providing simple selectable menus to user. It will show a small number of selectable menus to the user, and when user select the menu, it will perform the task that was written on the menu or gives the other selectable menus [12]. Menu driven interface is similar as tree structure. First menus act as a root of the

tree, and on each menu, they may or may not have branch menus.

Design relationships:
As it mentioned on design concerns section, selecting proper user interface will increase the chance of user to perform pull functionality of the software, and considering characteristics of user and software will make higher chance for user to do this. Potential user of the signature detection software is employee of the car dealers. They have little or no knowledge on artificial intelligence or image recognition process. Therefore, user interface must be simple to use without background knowledge of image recognition. Signature detection software need to take the image document and give result of the signature detection, so user interface does not need to provide unnecessary ability to user. This will only draw confusion to user, and it might lead to malicious code.

Design constraints:
Since menu driven interface limits the users action, it also constraints the functionality of the software. However, signature recognition software has simple functionality, so limiting software control will not affect the efficiency of this software.

3.2.7.3 Design Attributes: The structure of the menu driven interface is simple and intuitive. User simply have to click on the menu that is provided, so it doesnt require any background knowledge. In addition, this interface constraints the ability of the user by giving limited option to user. This will reduce the confusion of the user and avoid malicious code.

3.2.8 Image Converting (Dependency)

3.2.8.1 Design Concerns: The program will need to begin operations using PDF files, which are not a valid file format when operating with OpenCV. To resolve this issue, the use of an external image conversion program will be necessary. Functionality is the priority in design and consideration for efficiency will be made after we can confirm the project is doable.

3.2.8.2 Design Elements: The leading candidate for image conversion is ImageMagick, which operates as a standalone image converter within C++ that is open source. With command line support, the program can convert our PDFs into JPEG which OpenCV can then correctly interpret [8]. It does require more computer resources to function compared to some competitors such as GhostScript and MuPDF[9][10]. However, it is important to note that ImageMagick will do the entire conversion in one go unlike MuPDF [10], and it is free to use unlike GhostScript[9]. As it stands, the existence of an image converter is necessary for us when handling the various example inputs provided to us, but the longevity of this portion in our project is questionable. It is possible that our client will want to incorporate their own image conversion process so plans to make this section of our project easily removable have been made.

3.2.8.3 Dependency attributes: In the final stages of this project, the dependencies on file conversion may go away when handing the AI program off to our client. However, it is vital until we reach that point. Without this software, we

are unable to correctly build a library from which our project can read to make its assessments. As previously mentioned as well, OpenCV does not read PDFs in any capacity, and so to correctly analyze a document it must first be provided in a legible format for the program.

3.2.9 Runtime environment and Neural Net (Algorithm)

3.2.9.1 Design Concerns: The project being designed must be hosted in a runtime environment that is continuous as well as reliable. Given that we are taking an unsupervised learning approach to complete our task, it is imperative that it be able to run concurrently with instances of itself. The program must be designed in such a way that the ideal environment can host it and run it for extended periods of time. When taking this design paradigm into account, the algorithm running on this environment presents itself as the biggest design constraint. It must be optimized to be as noninvasive to the runtime environment as possible, while also efficiently making progress towards the desired outcome.

3.2.9.2 Design Elements: Design entities:
The first thing to take into account is what specific runtime environments can be chosen to perform this task. Careful consideration has landed our group to opt for OSUs flip server as opposed to other paid servers or personal machines. The second design entity is the algorithm itself. For this specific project we will be implementing a Convolutional Neural Network algorithm specifically suited for the processing of images, and qualities within them. Design wise, the neural net must be instantiated to run continuously, but in such a manner that does not exhaust our student resources on the OSU flip server. (e.g. utilize our allotted amount of memory) This will prove to be a key challenge since CNNs are extremely memory intensive algorithms, often handled by Graphical Processing Units as opposed to normal Computer Processing architectures.

Design relationships:
How often we can deploy our algorithm, as well as how efficiently we can allow it to run on the given environment, presents itself as a critical constraint in regards to the final outcome of our efforts. If either of these aspects are handled sub-optimally it will inevitably affect how well our algorithm learns, as well as its performance as an end product.

Design attributes:
OSUs flip server is monitored 24/7 as well as available 24/7. It has high memory constraints for students and excellent support if there were to be unexpected problems. Convolutional Neural Nets are excellent at discerning patterns within pixel image data. The various layers can be dissected as levels of depth regarding the images attributes. It requires high memory usage

3.2.9.3 Data Attributes: Data Attributes:
The data being fed into the algorithm is of numerical 3-dimensional nature. This is essentially what an image is when processed in its raw form. The convolutional network that will be instantiated will handle numbers ranging from 1-255 per channel, (RGB), per pixel. The handling and processing of all these different ranges is the culprit for convolutional

networks extremely heavy memory load. Our design must combine the processing of these numbers with the right amount of hidden neural layers so that the network functions properly, but does not overload the system.

3.3 Conclusion

For each piece of the project mentioned, the problem has been defined and a technology to remedy that problem has been identified. In addition, alternatives have been listed in the event we misjudged a given technology. Also, thanks to the viewpoints model being used, we now have greater clarity on how these various problems play in the bigger picture of the project and how best to approach them. While there are likely more parts to this project of which we are currently unaware, this provides a strong enough foundation to move forward into the development and coding phase of the project.

3.4 Design Changes

There is no design changes on our project

4 Tech Review

4.1 Language of OpenCV

4.1.1 Overview

This portion of the document will go into a technical review of three potential options for OpenCV supporting languages. The options are C++, Python, and Java; the pros and cons of each will be noted. Once all options are considered, there will be a discussion that directly compares these three choices. Lastly, a conclusion will be drawn from these three choices regarding the leader that we will utilize going forward.

4.1.2 Criteria

Our project has a hard requirement that the AI be open sourced. As such, we are restricted to utilizing established AI platforms. With this in mind, OpenCV is the leading candidate primarily due to its flexibility and popularity when wanting to handle image processing. When factoring in that OpenCV is our AI of choice, there are only three language options that are viable: C++, Python, and Java. Thus at present our only feasible options for supporting the AI are to choose between these three programming languages.

4.1.3 Options

4.1.3.1 C++: C++ is the primary language used by OpenCV and the most robust in general. C++ is widely regarded as one of the most efficient languages when it comes to runtime resources. It also carries the largest selection of tutorials

and guides to use for picking up and using OpenCV. The library supported under C++ is massive when compared to most other languages. Lastly, the community backing it is extensive. This option has been available for years, and thanks to its status as free-to-use and its efficiency, many users have taken to working with it. As a result, many questions have been answered already regarding the usage of OpenCV within a C++ language. Some issues do arise with the C++ variant with the largest being poor documentation. Another complaint is that while it supports a strong library, its actual pool of machine learning components is rather limited by comparison. This results in difficulties when attempting to use multiple different learning routines on a program. Another issue is the challenge of interpreting code and the debugging process. While this is a normal issue for any programming language, for C++ it is compounded when attempting to make use of machine learning algorithms. If the programmer needs to write any code from scratch, this method becomes a nightmare for comprehension.

4.1.3.2 Python: Python is the leading rival of C++ in language preferences. The most popular strength of Python is the simple ease of use associated with it. The language is straightforward and easy to write in. Another interesting fact about Python is that its often regarded as a language of scientific computing, with support for packages such as OpenCV, numpy, scipy, scikit-learn, and matplotlib[1]. In direct contrast to C++, the debugging and visualization of Python is considered very good which is primarily the result of its previously mentioned simple-to-understand syntax. Also, portability is something that should be considered with our program, and as such popularity and use of Python means that most systems already have support for this language. However, the documentation is lacking even more than the C++ option. There are concerns including difficulty discerning what a function does on a deeper level, what the various parameters do to impact the program, and relatively weak guides in the usage of this version.

4.1.3.3 Java: Java is the final language being considered for use within OpenCV. This language as a choice is a bit different from others being considered in that, while all lack documentation, this one takes it to the furthest extreme. Having not existed until version 2.4.4 of OpenCV, Javas introduction is the latest of the bunch. Its primary focus is on its usage in Android devices[2]. When attempting to research this language in particular, there was very little mention of it at all, with tutorials and guides being few and far between. While Java itself is a reliable enough language, it has failed to gain traction when placed in an OpenCV environment primarily due to its middle-of-the-ground nature. What this means is that its runtime performance is generally in-between C++ and Python. This means if performance was a concern, then C++ is the better option. When considering ease of coding, then Python wins, with Java falling into second place for either category. This is only made worse by the previously mentioned lack of documentation, which means it is extremely difficult to make proper use of this language. While other languages also suffer from this, they at least have several years of exposure in which the documentation has been improved or added. Java on the other hand has had minimal additions.

4.1.4 Discussion

These three choices are largely different from one another with the only considerable common ground being that they all have poor documentation in general. The discrepancy however, is that of the three options C++ does carry the strongest online support and Java is at the bottom by a significant margin. Next for consideration is ease of use for coding purposes. In this aspect, Python is the strongest candidate and Java and C++ are the ones trailing behind. When considering code efficiency at runtime, then C++ is the clear winner.

4.1.5 Conclusion

Given what is known about the three choices at present, the current leader that will be selected is C++. There is also the possibility that since Python utilizes C++ under the hood, should we need to try and move to a different language due to library restrictions the migration wouldnt be too taxing as opposed to involving C++ or going from Python to C++.

4.2 Filter Algorithm

4.2.1 Overview

This section will be focused on the consideration of image blurring techniques in OpenCV to assist in dealing with potential noise that may exist on the documents well be scanning. To combat this issue and make the image more readable for the program, well need to make use of some form of blurring technique in which there are three major choices.

4.2.2 Criteria

Whatever technique is being utilized must be functioning on OpenCV. The blur in question must also be safe enough as to not disrupt the natural look or flow of the document, should it become too extreme text bodies will appear completely connected. Lastly, the technique used must be efficient enough as to not impact overall performance to a point where the program cannot keep up with potential inputs.

4.2.3 Options

4.2.3.1 Gaussian: Gaussian is one of the most common implementation methods for blurring an image. This method takes a defined kernel and slides it across the image to smooth it out [3]. This kernel needs to be defined by the programmer with the parameters sigmaX and sigmaY. These values define their respective x and y scale. In addition, should the parameter for y be left undefined, then it will inherit the value of x creating a square shape [4].

4.2.3.2 Median: Median filtering is another common selection for its ability to carry out limited edge detection to help preserve the form of a given shape. While not perfect at this task it is often sufficient at preserving shape while also removing noise [5]. This method also carries a relatively new function built in to OpenCV that processes all channels of the image input independently [3]. This process is best at tackling the issue of images with heavy salt-and-pepper noise in which the central median value is always replaced by some pixel image, unlike most other filter methods [4].

4.2.3.3 Bilateral: Bilateral filtering is very effective at removing noise within an image while still retaining the edges to everything. This comes at the price of significantly slower runtime to complete when compared with competitors. The procedure is done through the use of two gaussian filters, one for space to handle nearby pixels and the other for intensity differences. This allows it to blur everything but areas with high differences which naturally indicate an edge in the image [4].

4.2.4 Discussion

When comparing these three filter choices the areas that need to be considered are runtime performance and the importance of keeping the edge to an image. If the only concern is performance then Gaussian would be the clear choice, but since were dealing with the scanning of text documents, the situation becomes a bit more challenging. At present my assumption is that to correctly detect potential signatures, well need to be able to properly see the relative shape of words which means that the edge will be vital to accomplishing this task.

4.2.5 Conclusion

Since our greatest concern is determining if it is even possible to accurately detect if a page has a signature section and if its been properly signed then performance will be taking a back seat to simply completing the task. As such, our best bet is to use bilateral given it effectively removes noise while still maintaining the form of the document to its most original state.

4.3 File Conversion Software

4.3.1 Overview

CDK stores all submitted documents as PDF formats which is not a readable file type with OpenCV. As a result, we will need to consider ways of converting the PDF into something compatible. Several resources already exist in this department that show potential use for our purposes, but determining which to select is the current problem.

4.3.2 Criteria

It is established that the documents well be dealing with are of the PDF format. Beyond this, the team is free to select however we see fit to implement this portion. Our current preferences indicate the use of C++ and OpenCV so well need to choose based on what best conforms to the language and platform.

4.3.3 Options

4.3.3.1 ImageMagick: ImageMagick is a free open source option for image conversion and editing. It carries the functionality to read and write over 200 formats including PDF, JPEG, and PNG. The software carries command line support and works with all three considered language options that our team is considering. It also functions on Linux, Windows, OSX, iOS, and Android devices [6]. With all of this functionality comes a major drawback however: it is a very heavy program to use. While it would be nice to have all these options available to us, we only plan a single PDF to image conversion and that would be wasting the majority of what makes this program so taxing to operate.

4.3.3.2 Ghostscript: Ghostscript is a copyrighted software package that can convert PDF files as well. It can function as an interpreter for PDFs and turn them into PostScript files [7]. To utilize this PostScript, we would then need to search for an addition software package for converting it into a suitable format for OpenCV. As such, while a lightweight program on its own, it will require a daisy chain of conversions to make it ready for manipulation on our end.

4.3.3.3 MuPDF: MuPDF is an open source software package that is written in C with the intent of doing PDF conversions. This program is an extremely lightweight PDF viewer and editor with the ability to convert the file into various formats such as HTML, SVG, and CBZ. In addition to C, there is support for Java and Android functionality with this program [8]. Unfortunately, to complete the conversion there will need to be an additional conversion of SVG into a format that is accepted by OpenCV.

4.3.4 Discussion

Out of the three options provided one of them is not open source which is cause for concern given the project aims to be an open source product. As such, to be safe in our decisions the choice falls between MuPDF and ImageMagick. ImageMagick is the all-in-one package with the support for what we need (and then some), whereas MuPDF would only get us closer to the proper image format that is required. While ImageMagick achieves our goal immediately, its added features make it a significantly more taxing program to use, a factor that we must consider.

4.3.5 Conclusion

While efficiency is a large concern for us given the scale of the project and how computationally heavy artificial intelligence is, we do need to consider that we primarily just want to prove this project can be done. As a result, ImageMagick is the software we will be going with since it stays safe in open source and reduces the amount of work required for us to chain several software together to get an input we can actively use within OpenCV.

4.4 References

[1] ”LearnOpenCV” Internet: https://www.learnopencv.com, Oct. 30, 2015 [Nov. 14, 2017].

[2] ”Introduction to Java Development” Internet: https://docs.opencv.org, Nov. 13, 2017 [Nov. 14, 2017].

[3] ”OpenCV Tutorial C++” Internet: https://opencv-srf.blogspot.com, [Nov. 21, 2017].

[4] ”Smoothing Images” Internet: https://docs.opencv.org, [Nov. 21, 2017].

[5] ”Filters B” Internet: http://www.bogotobogo.com, 2016 [Nov. 20, 2017].

[6] ”Image Magick” Internet: https://www.imagemagick.org, 2017 [Nov. 21, 2017].

[7] ”Ghostscript” Internet: https://www.ghostscript.com, 2016 [Nov. 20, 2017].

[8] ”MuPDF” Internet: https://mupdf.com/, 2017 [Nov. 21, 2017].

4.5 The algorithm

4.5.1 Overview

The main goal of this project is to develop an algorithm that can discern whether a document contains signatures, needs to be signed, or does not contain sign-able spaces; based on the analysis of a large number of already existing

documents. Given that the task is to utilize an already existing pool of samples to discern key patterns relating to our three criteria while keeping scalability in mind, this problem is best catered to by a neural network algorithm. Neural networks have the unique property of discerning patterns they were never explicitly trained to discern. This property makes them extremely unique when exploring image recognition problems, since these are based on recognizing a wide number of very specific features no one person could ever possibly specifically outline.

4.5.2 Criteria

When looking at neural nets tasked with image processing, were looking at thousands of pixel related inputs that must be correlated in order to make sense of a certain picture. To land on a particular neural network paradigm we will be looking at the following categories. Efficiency of data propagation: how quickly can the net make sense of a given data set. Learning proficiency: how quickly is the neural network projected to draw relationships between given inputs in comparison to other neural networks. Memory efficiency: what level of memory usage is required to efficiently run this network in comparison to other neural networks.

4.5.3 FeedForward Neural Networks

Feed forward neural networks are artificial neural networks where connections between units do not form a cycle. In essence, we are talking about neural networks in which data only flows forward, that means it never reassesses data as it passes through, rather it relies on existing architecture to make a prediction based on its current state. These neural nets learn by analyzing the correctness of their estimate against known estimates, which classifies their learning as a form of gradient descent, a first order iterative optimization algorithm for finding the minimum of a function[1]. Feedforward neural networks can provide very fast, resource mindful solutions for simple input data sets, e.g. : guessing a student's overall grade while only being given hours of homework and hours studied. Feedforward neural nets can become flustered when the input data set becomes more complex and thus are to be utilized for scaled down versions of complex problems.[2]

4.5.4 Convolutional Neural Networks

Convolutional neural networks utilize a variety of different methods in order to discern key patterns about a given data set. The most important method a convolutional neural net utilizes is called, quite fittingly, convolution. Convolving consists of picking out subsets of a certain data set and merging them into one meaningful value. This is done throughout the entirety of the data set ensuring that data subsets have a certain percentage of overlap. These values are then remapped into their own data set creating a feature map representative of the original data set. By repeating this process, our neural net can start discerning more and more particular features about a data set. A convolutional neural network learns by backpropagation, short for "backward propagation of errors" [3] , it is uniquely suited to discern patterns within an image, but it must be trained, which in turn means it could be under-trained or over-trained at the point of assessment. Convolutional neural nets are extremely resource heavy and tend to require dedicated GPUs to run efficiently.[4]

4.5.5 Genetic algorithms

While most neural networks utilize some kind of learning function in order reassess the weights of it?s connections, this can often be a lengthy and resource heavy process. Another approach to learning is the genetic algorithm approach, where we exploit evolutionary concepts in order to create better and smarter nets at each iteration. Genetic neural network algorithms, like all neural network algorithms, start with a neural net with randomized weights that is fully incapable of completing the task that it was given, or rather, a collection of them. A single neural net never learns anything in a genetic algorithm, at its time of birth it was either destined to succeed and breed, or to fail. Among all these neural nets, which are all randomized differently, there will be a subset deemed the most ?fit?. Fitness is the way we assess how well a neural net performs its purpose within a genetic algorithm. The usefulness of the ?breeding? technique is that it is straightforward and simple, and thus creates learning from randomization rather than mathematical assessment. Genetic algorithm neural nets are resource mindful, but can get stuck in certain iterations and/or take a very long (uncertain) amount of time to reach peak performance.[5]

4.5.6 Discussion

We explored three different neural network paradigms that each highlighted specific components of neural network capabilities. The first was the straightforward feedforward network. A network that learns by optimizing a cost function (cost = —correct answer - wrong answer—), and is only concerned with feeding information through its neurons. It can perform simple tasks very quickly, but its simplistic nature can find itself flustered by complex problem spaces. We explored the convolutional neural network, a network suited for discerning patterns within a data set by iterating recursively over key feature maps, allowing it to discern more complex patterns at each iteration. These neural networks are suited for image detection and are the industry standard in terms of image recognition. While extremely efficient in performing its task, these are resource heavy and rely on powerful GPUs to work correctly. Lastly, we explored the genetic algorithm learning approach, where we were not concerned in teaching any single network to perform our task, but rather only concerned with ?breeding? the networks that are most successful at performing it. This approach is resource mindful, but can get stuck in certain iterations and never learn to perform our task satisfactorily or take an unreasonable amount of time to do so.

4.5.7 Conclusion

While the feedforward approach is efficient, it does not seem to be powerful enough to tackle the problem space that we will be dealing with. The convolutional network is basically the obvious choice but it is fair to take genetic mutation and learning into consideration if our simulation encounters points where backpropagation performs underwhelmingly.

4.5.8 References

[1] ”Wikipedia Gradient Descent” Internet: https://en.wikipedia.org/wiki/Gradient_descent, [Nov.15,2017].

[2] ”CS Stanford, Neural Networks” Internet: https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Architecture/feedforward.html, [Nov.15,2017].

[3] ”Brilliant : Backpropagation” Internet: https://brilliant.org/wiki/backpropagation/, [Nov.15,2017].

[4] ”Adit Deshpande: A Begginer’s Guide to Understanding Convolutional Neural Networks” Internet: https://adeshpande3.github.i
Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/, [Nov.15,2017].

[5] ”Matt Harvey: Let’s evolve a neural network with a genetic algorithm” Internet: https://blog.coast.ai/lets-evolve-a-neural-network-with-a-genetic-algorithm-code-included-8809bece164, [Nov.15,2017].

4.6 Container Software

4.6.1 Overview

One of the most important features outlined throughout our project is the ability for it to be used from any machine. The project is to act as a standalone application that is easy to launch and operate regardless of the machine we are utilizing. In order to do so it must be deployed utilizing containerizing software, which addresses this problem at its core.

4.6.2 Criteria

One of the most important features outlined throughout our project is the ability for it to be used from any machine. The project is to act as a standalone application that is easy to launch and operate regardless of the machine we?re utilizing. In order to do so it must be deployed utilizing containerizing software, which addresses this problem at its core.

4.6.3 LXC

LXC is specifically designed for Linux users and aims to emulate the Linux standard installation without the need for a separate kernel. LCX?s API supports a number of languages for deployment including Python 3, Lua, ruby, and Go. LXC does not support being deployed in c++ which could create extra constraints if the native code of our application runs on c++. LXC support, being a Linux based application, depends on stable Linux releases and the efforts of their creators to cater to it with specific distributions. Documentation for its API is existent but often difficult to find, and almost always requiring disambiguation for specific tasks. This is mainly due to problems being Linux distribution specific.

4.6.4 LXD

LXD is also designed for Linux distributions and offers an experience similar to virtual machines but utilizes Linux containers instead. LXD is an image based containerizing software available for a wide range of Linux distributions that supports all the languages LXC does. LXD main pros are it is intuitive and simple API, its security driven design, and image based deployments. Unlike LXC whom relies on singular developers to tend to it, LXD is developed and distributed by Canonical LTD. Canonical LTD offers support and documentation for LXD which is lacking for LXC. LXD builds on top of LXC and acts as a friendlier version of the LXC framework.

4.6.5 Docker

Docker is the current industry preferred choice when it comes to container software. Docker is faster than other container options, is extremely well documented and has its own public repositories to obtain the latest installations. The Docker Hub hosts thousands of container images as well as support for the download of ready to use apps, something that tends to be hard, if not impossible to find on other repository websites such as GitHub. Even though Docker is advertised as supporting Windows and Mac OSX, it utilizes virtual machines on these which in essence defeats the point of the container approach. Docker is written in the Go programming language, something which the team must familiarize itself before utilizing it [1].

4.6.6 Discussion

Each of our three choices contains its own set of pros and setbacks. LXD pertains to more bare-bones approach where we utilize the basics of the containerizing paradigm at our own complete discretion. This could save memory, but due to its poor documentation and support, will need extra time to get accustomed to in order for the team to become proficient in its usage. LXD is well documented and image based, both of which will help save time and effort when containerizing the application. Even though there is extensive support for LXD, it builds LXC, and it could potentially fall into the same undocumented setbacks and LXC. Docker is widely regarded as the best container software across the industry and was mentioned by our client as a choice when the project was first discussed. It is well documented and supported, but boasts cross platform implementations even though it utilizes virtual machines to do so, which can be done to utilize LXC and LXD. Docker also will need that the team familiarize itself with the Go programming language [2].

4.6.7 Conclusion

While LXC could be a very powerful resource efficient approach, utilizing LXD will provide most if not all LXC features in more time sensible manner. While LXD is well documented and industry standard ready, Docker seems to be widely regarded as superior due to the aforementioned features. While not having utilized either they might seem like equals, its popularity across many developer opinions warrants that Docker be our main choice, with LXD being only pertinent if certain required features prove untimely, or inefficient, on Docker [3].

4.6.8 References

[1] ”Linux Containers, LXC” Internet: https://linuxcontainers.org/lxc/introduction/, [Nov. 18, 2017]

[2] ”Linux Containers, LXD” Internet: https://linuxcontainers.org/lxd/introduction/, [Nov. 19, 2017]

[3] ”3 Pros and 3 cons of Working with Docker Containers” Internet: https://sweetcode.io/3-pros-3-cons-working-docker-containers/, [Nov. 20, 2017]

4.7 Development Platform

4.7.1 Overview

In order for a neural network algorithm to run it must run continuously and at a high end processing power. It takes many iterations of a neural network before it is optimized to perform the task that it is given. For this specific task we will explore three different continuous run-time environment options that can host our application, our home pcs, the OSU flip server, other resources / paid services.

4.7.2 Criteria

The chosen technology must allow a constant run-time environment of at least 24 hours at a time. This number is rather arbitrary but optimal so that instances can be checked in the mornings and left unsupervised throughout the day. It must be able to run Linux and must have enough computing power to efficiently run our algorithm uninterrupted.

4.7.3 Home-PC

Developing the algorithm on our home computers allows us ultimate accessibility especially when executing certain permissions necessary for the proper functionality of our various frameworks. It also allows unrestricted access to installing any third party software we deem appropriate to test and maintain our application. Utilizing our home pcs is free of charge, but home pcs might not have the capability to efficiently run our algorithm, and will require constant attendance to ensure the continuous run-time environment necessary to achieve optimal learning.

4.7.4 OSU flip server

The OSU flip server guarantees a constant run-time environment monitored by OSU computing facilities staff. OSUs computing resources are vast and allow our algorithm to run efficiently and uninterrupted with minor restrictions on memory usage. They are free of charge but are restricting when it comes to installing permissions.

4.7.5 Other resources / paid services

Our last option would be to find a paid service that allows us to code free of restrictions and in a continuous run-time environment. These services would combine the pros of both OSU?s flip server and our home computers, allowing us full control of our environment as well as a supervised server unlikely to go down at any critical point. Utilizing one of these services would come at a cost that would have to be paid by the members of this group, putting an economic constraint on the project.

4.7.6 Discussion

Each one of these options is fully capable and viable in terms of our existing criteria. The choice will likely come down to a number of factors for each of our options. For home pcs we must figure out if each member’s option has

the computing power to run our algorithm. This basically comes down to a powerful gaming oriented GPU. For paid services it will be decided on whether the group can accommodate the expense of renting / utilizing a pay as you go server. If neither of these options is viable, the OSU flip server must be utilized and its restrictions addressed to ensure the proper functionality of our various frameworks.

4.7.7 Conclusion

Given that the OSU flip server has both the computing capability, as well as constant availability, it is the most likely candidate for the development of our application. In the unlikely event that a critical function cannot be executed due to a lack of permissions non addressable as students, one of the other choices must be picked and our strategy reassessed.

4.8 Deep Learning library

4.8.1 Overview:

On this section, most commonly used deep learning library will be compared. The three libraries that will be compared are TensorFlow, Deep Learning4 Java(DL4J), and OpenCV. The best library that fit into our project need will be selected.

4.8.2 Criteria:

Our main goal of the project is to build the signature recognition program by using image detecting API. To accomplish this task, it is essential to find a suitable machine learning library for signature recognition. The machine learning library must be runnable in different operating system, support many programming languages, provide handwriting recognition algorithms or functions, have many community, and fast enough.

4.8.3 Potential Choices:

4.8.3.1 TensorFlow: TensorFlow has a short history. However, large communities use TensorFlow, and there are more than 6000 open source repositories online [1]. TensorFlow has special feature on numerical computation which makes it a powerful tool. Unlike other machine learning library, TensorFlow uses data flow graphs that contains vectors or matrixes called tensor, and this data flow graphs allow developers to compute the calculation on one or more CPUs or GPUs [1].

The other strength of TensorFlow is flexibility. It supports both desktop operating systems and mobile operating systems such as 64-bit Linux, MacOS, Windows, Android, and IOS [1]. Also, TensorFlow Application Program Interface (API) supports in Python, Java, C, and Go [1].

There are also disadvantages using TensorFlow. The main problem with TensorFlow is the speed. It has dramatically slower speed compare of other frameworks. Also, TensorFlow is heavier than other machine learning libraries. Since TensorFlow force to use set of vector or matrix for flow graph data, copying large matrices cause slowness of the

program [2]. The last problem of the TensorFlow is the optimization of Java API. It is an experimental step supporting Java API, so Java API for TensorFlow is not stable.

4.8.3.2 Deep Learning for Java(DL4J): Deep Learning for Java (DL4J) is the machine learning library supports on Java virtual machine (JVM). It supports languages such as Java, Scala, Clojure, and Kotlin, but Java is the main language [2]. The strength of the DL4J is the speed. There are several characteristics that makes DL4J run faster. First characteristic is the DL4J has linear algebra computations [2]. It is much simpler than other complicated computation algorithms. The second characteristic is that DL4J can run in parallel, and this parallel setting is automatic. Since the user does not have to create parallel setting, the system runs much faster [2]. The last characteristic is that DL4J relies on JavaCPP which parse C/C++ header files to Java interface files [2]. This allows DL4J run faster since most of the high cost program are written in C or C++.

Although the DL4J API language supports lack of diversity, it won't cause much problem since Java is one of the largest programming language. Almost 10 million developers use this language, and many large companies highly rely on Java or a JVM based system [2]. In addition, Java is supported in many platforms without converting the language. Once developers write the code, they simply have to pass their code in JVM to run on different operating system.

There are two main disadvantages using this machine learning library. Firstly, DL4J does not support many languages. Although Java can support different environments by using JVM, different programming languages have strength on different circumstances. Supporting more programming language will to give diversity to developers. The second problem a small community compared to other machine learning libraries because DL4J has a short history.

4.8.3.3 OpenCV: Unlike the previous two machine learning libraries, OpenCV has a long history, so it has a large community. According to OpenCV website, there are 47 thousand people in the user community, and more than 14 million downloads [3]. Additionally, our teaching assistance have experience on OpenCV. This is highly advantageous since our group can get advice directly from our TA.

OpenCV is more flexible then TensorFlow. It runs on both desktop (Windows, Linux, Android, MacOS, FreeBSD, OpenBSD) and mobile (Android, Maemo, iOS), and supports C++, C, Python and Java languages [3].

Speed is another benefit of using OpenCV. Since OpenCV library is written in C/C++, it is fast. Also, OpenCV currently supports about 500 CUDA and OpenCL algorithms to accelerate the original algorithms [3]. CUDA and OpenCL are the modern GPU accelerators [3]. These GPU accelerators increase GPU performance, so computation time for algorithms shrinks.

One of the disadvantages of the OpenCV is that the library is not suitable for Java language yet. Unlike other two libraries, OpenCV supported Java language recently. Therefore, it has lack of implementation examples with Java, and problems with using Java language.

4.8.4 Discussion, compare and contrast:

As it mentioned on criteria section, machine learning library needs to support enough algorithms, language, and operating system. Also, it need to have fast computation speed, and large community. Following table shows the

compare and contrast based on these requirements.

	TensorFlow	DL4J	OpenCV
handwriting recognition algorithms	Decent algorithms	Decent algorithms	Decent algorithms
Language support	Many	very few	Many
Operating system support	Enough	Enough	Many
Community	Many	Few	Abound
Speed	Slow	Fast	Fast

TABLE 1
Compare and Contrast

4.8.5 Conclusion:

From the discussion, compare and contrast section, it is clear that OpenCV is the best choice for CDK Data Stream AI project. It has enough handwriting recognition algorithm, supports various language and operating systems, has a large community, and fast speed.

4.9 Text recognition method for detecting signature box

4.9.1 Overview:

On this section, three text recognition method that is used to detect signature box will be compared. Finding signature box is important part of detecting signature. Since format of the documents vary from each company, signature box must be found before recognizing signature. To locate signature box, word relate to signature box such as “signature” must be found from the document to locate the signature box, and text recognition method help software to find certain words from text images. These three methods are optical character recognition, intelligent character recognition, and intelligent word recognition.

4.9.2 Criteria:

The method must be able to detect words relate to signature box such as “signature”, “company signature”, “cosigner line”. Accuracy is the most important thing to considered.

4.9.3 Potential Choices:

4.9.3.1 Optical Character Recognition (OCR): OCR is a one of the earliest method to convert text images or handwritten texts to editable electronical text document. To do this, OCR takes three steps. According to the Recogniform Technology, it first analyzes the layout of the text image to identify the columns, paragraphs, text lines and words from the text. Then, OCR splits the image to individual characters. On last step, it identifies characters and converts characters to electronic character [4]. OCR takes two methods to recognize the characters which are pattern recognition and feature detection. Pattern recognition determines the character by matching the pattern of the character. It compares the splitted character with the list of the character that stored in the program. If splitted character is similar with stored character,

it converts splitted character into electronic character . However, printed text doesn't have identical different font, and it gets much more complicated for handwritten text. To solve this problem, OCR uses feature detection method. This method determines character by evaluating the characteristics of each character. OCR program has list of features for each character such as angles, number of lines, and such. By using this list of features, OCR can recognize the character even if it has different font [4].

4.9.3.2 Intelligent Character Recognition (ICR): ICR is an advanced version of OCT. It has self-learning features that allow ICR to learn through its experience [5]. Since ICR learns different fonts and styles from the text images or handwritten texts, unlike OCR, it can determine handwritten texts [5]. Also, accuracy of the ICR increase as it converts text. ICR has many advantages, but ICR programs are heavier than OCR programs.

4.9.3.3 Intelligent Word Recognition (IWR): Unlike both OCR and ICR, IWR detects words from the text images instead of characters. It is very similar to ICR, but IWR has dictionary that makes possible to detect the words from text [6]. For example, while ICR extract "hello" as individual letters like h, e, l, l, and o, IWR matches pattern from dictionary to recognize and extract exact word.

4.9.4 Discussion, compare and contrast:

As it mentioned on criteria section, accuracy is the most important feature to be considered. In addition, self-learnability and size of the algorithm need to be considered for software aspect. Three methods are compared based on accuracy, weight, and self-learnability.

	ORC	ICR	IWR
Accuracy	weak	medium	strong
Weight of the algorithm	light	medium	heavy
Self learnability	no	yes	yes

TABLE 2
Compare and Contrast

4.9.5 Conclusion:

For our software, ICR is the best choice among three methods since it has medium accuracy and weight, and it also has self-learnability. Although, IWR has the highest accuracy, it has unnecessary features which makes the software heavy. Finding signature does not need to understand the word. It just need to find the location of the signature box.

4.10 User interface

4.10.1 Overview:

On this section, three types of user interface, command line interface, menu driven interface, and form based interface, will be compared and selected. It is important to select proper type of user interface for software. To select proper user interface, user of the software must be estimated and evaluated. User of the signature detecting software are the

employee of the car dealers. They have little or no knowledge of artificial intelligent or computer science. Therefore, simple and easy to learn user interface must be build it from the proper type of user interface.

4.10.2 Criteria:

Signature detection software must provide black box form of user interface since user doesn't have to know about detail process. Also, it must be simple to use, and only necessary option must be provided.

4.10.3 Potential Choices:

4.10.3.1 Command Line: A command line interfaces simplest interface that allow user to interact with the software or hardware by typing command on command line [7]. It is oldest interface among three interfaces, and designed for old computers. Since old computers are used by the professionals, interface of it was not user friendly. To use this interface, user must memorize a lot of different commands, so it takes time to get familiarized.

4.10.3.2 Menu Driven: A menu driven interface allow user to interact with software by providing simple selectable menus [7]. It is commonly used on cash machines, ticket machines and information kiosks [7]. Each menu has instruction that describes what action it will took by selecting the menu. To interact with the software, user just need to select the menu, and software will show sub menus or perform corresponding action. This interface is very intuitive and user friendly. It does not require prior knowledge to interact with the software.

4.10.3.3 Form Based: A form Based interface collects data from the user and send it to the system. It collects data by using drop-down menus, check boxes, text boxes, radio boxes, text areas, and buttons [7]. It is commonly used for software that collects data or requires data to perform the function [7]. This interface can be seen on survey website and mathematical software.

4.10.4 Discussion, compare and contrast:

A command line interface is very simple to design and easy for user to learn how this interface works. However, it takes time for user to get familiarized with the interface because user have to memorize different command that is used for different situation. This interface is recommended for software that is designed for software developer.

For software that has simple functions, menu driven interface is recommended. It is intuitive and simple to learn how to use. User doesn't require any knowledge to use software. If software needs to provide many functions, this is not good interface to use.

A form based interface is recommended for software that needs to take a lot of data from user to perform the function. If software does not need to take many data at a time, this interface is not suitable for that software.

4.10.5 Conclusion:

Since signature detecting software interface need to be black box module and simple to use, menu driven interface and form based interface matches for the software. However, signature detection software does not need many input at a time, so menu driven interface is the best interface for our project.

4.11 References

[1] ”TensorFlow website” Internet: https://deeplearning4j.org/, [Nov. 21, 2017].

[2] ”Deeplearning4j website” Internet: https://deeplearning4j.org/, 2016 [Nov. 21, 2017].

[3] ”OpenCV library website” Internet: https://opencv.org/, [Nov. 21, 2017].

[4] ”OCR - Optical Character Recognition” Internet: http://www.recogniform.net/eng/ocr-optical-character-recognition.html, [Nov. 21, 2017].

[5] ”The Difference Between OCR and ICR and Why It Matters for Organizations Using DMS” Internet: https://www.efilecabinet.com difference-between-ocr-and-icr-and-why-it-matters-for-organizations-using-dms/, [Nov. 21, 2017].

[6] ”Intelligent Word Recognition” Internet: http://captricity.com/intelligent-word-recognition/, 2016 [Nov. 21, 2017].

[7] ”System software: User interfaces” Internet: https://en.wikibooks.org/wiki/A-levelComputing/CIE/Computersystems,commun 2017 [Nov. 21, 2017].