Oregon State University
College of Engineering

CS Capstone Progress Report

February 16, 2018

CDK Data Stream AI

Prepared for

CDK Global

Chris Smith

Prepared by

Group 65

SigFind

Inhyuk Lee

Jacob Geddings

Juan Mugica

Abstract

This report is the cumulative summary of our groups progress throughout the winter term. We will be discussing the progress made for the term and how we handled them as well as touching upon client interaction and group decisions moving forward. This report will also talk about the three primary components of our project and why we de-constructed our project this way. At the end there will also be a week-by-week breakdown of events. Moving forward, we will be entering beta phase for the goal of the project as well as tackling potential stretch goals.

CONTENTS

1 Introduction

Our group fell into the demographic that did not receive the general guideline and as such our group contributions are a bit more scattered than desired. The report breakdown for individual contribution goes as such: sections one through five as well as seven are written by Jacob Geddings. Section six was done by Inhyuk Lee and lastly, Juan completed section seven. As will be noted in section four, ImageMagick was not as difficult as we initially thought and integrated very well with what we decided to use OCR Tesseract for. As a result, while ImageMagick does have a unique section it will also be mentioned within the OCR component.

2 Project Purposes and Goals

Our project, CDK Data Stream AI, is concerned with the construction of a program that can arbitrarily detect if a document has a signature line and if it has been signed. To accomplish this primary task the program must make use of an open source AI platform, function as a black box, and be modular enough that parts can be added or removed from the program. Accompanying this primary goal is a series of stretch goals that, should our primary concern be easy to implement, become our focus for the remainder of the coding section. These stretch goals include license validation, data security, image categorizing, and vehicle image processing. Should these be accomplished, additional interests have been stated in regards to automatic construction of consumer portfolios based on submitted forms.

A major concern for our client is that we do not get caught up on peripheral issues or stretch goals before the core task is completed. Weve been explicitly requested not to worry about container software, cloud operation, or user interface in particular. Another issue of which we must remain mindful is data sensitivity; any documents or forms were permitted to use for testing purposes cannot be made public. They are sanitized for us to legally view, but the forms are not open to the public. Lastly, CDK Global is aware that this project may not be feasible and in lieu of a completed product, they will accept a report indicating why we were unable to complete the project.

3 Recap of Last Term and Progress

All required documentation for fall term has been completed and submitted. All documents requiring client verification have been cleared now including the design document which was pushed into winter term. Regular meetings have been scheduled with our TA every week for 20 minutes as well as 30 minute meetings with our client. Communication has been setup for intergroup contact through the use of group text/chat services, and contact with our client has been established through the university-provided emails.

The program itself has also made some progress throughout the term. We have selected OpenCV as our launching platform for the project and will be using its C++ variant as our leading choice. Our group has successfully completed setup of OpenCV on our personal machines and student FLIP server. This includes adding the ability to view images from PUTTY. Lastly, our most recent development for OpenCV is constructing an image window for any submitted image. Plans have been made to ensure progress continues throughout winter with the intent to have PDF to PNG conversion included into our program as well as the ability to construct reduced windows in OCR Tesseract to pinpoint where a signature line is located.

4 Projected Goals of Winter Term

Winter is the primary window for coding to be completed in and as such, our lead goal for the term is to have the primary objective of this project, detecting if a signature line was signed, to be completed by the end of the season. To accomplish this task there are several subsections we constructed towards reaching this goal to help facilitate workflow. The three core breaks came in the form of establishing a file conversion software, construction of an OCR program to isolate and crop images, and a neural network to then judge the provided image. While we allotted the entirety of the term to completing this core function we also made tentative plans for setbacks as well as stretch goals. Some problems we anticipated encountering included: insufficient language for OpenCV, neural network instantiation issues, and inability to accurately crop images down. To protect against these problems, we readied a plan to transition to Python if necessary as well as numerous resources for starting up networks as well as different ways of isolating the image without the use of OCR.

5 ImageMagick

For the file conversion software, we opted into using ImageMagick for our PDF to PNG needs. Ultimately, on its own this portion of the project was straightforward and shares a close bond with the OCR Tesseract portion. As a result, ImageMagick will be discussed briefly on its own for this section and in the following OCR section there will be more discussion on how OCR handles its output. This open-source program was immediately compatible with our school servers and is running on all three of our machines. While this program is exhaustive in all the functions it can carry out we only needed it for its conversion capabilities. We required the ability to convert a given PDF into a upscaled PNG image to have passed off to OCR Tesseract. As it currently stands, the program is both running and accurately producing upscaled images to then pass off to OCR. One problem we currently have is finding a more elegant way of directly merging ImageMagick and OCR given their dependency on one another. For now, we have wrapped both functions into a bash script to enable a onetime call to prove they can work together.

6 OCR Tesseract

6.1 Software and Technique

6.1.1 ImageMagick

ImageMagic is free and open source software that can create, edit, compose, or convert bitmap images. It supports PNG, JPEG, GIF, HEIC, TIFF, DPX, EXR, WebP, Postscript, PDF, and SVG files. It can be utilized from the command line or program. It supports various types of programming languages such as c, c++, Java, Python, and many more.

6.1.2 OCR Tesseract

Optical character recognition(OCR) is a technique of converting images to machine encoded text. Generally, most of the OCR has two part which are pre-processing, and character recognition. On pre-processing, it edits the given images

to raise the accuracy of the character recognition rate. To do this, OCR binarize the image which convert images from color to black and white. This process will make easier for program to separate the text from the background. After binarization, OCR takes segmentation process. First, it separates the page into different sections such as photo, table, and text box. After segmenting the page, it locates where the text is and segment it to line, word, then character. When preprocessing is done, OCR recognize the characters and converts to machine encoded text. To do this, OCR look for pattern and features of given image. For example, when image of the character given, OCR finds similar character from library, and extracts features such as number of lines and angle between lines. After evaluating given image, it gives the closes machine encoded character.

6.2 Why This Was Chosen

Our group decided to use both ImageMagic and OCR Tesseract to build signature locating program. The basic algorithm of the program is like this. When pdf document is given, it converts the given pdf document to machine encoded text. Then, it searches for certain words such as Signature to locate the signature box. Lastly, program save the signature box and pass it to our convolutional neural network to detect signature.

The main reason of using ImageMagic and OCR Tesseract is that our group have limited time. We could build software that can convert pdf to image file and recognize the characters, but it will take a lot of time. Since our client wanted us to focus on making accurate signature detection AI, our group wanted to focus on building decent AI. To do this, we decided to use free open source software for locating signature. The other reason that ImageMagic and OCR Tesseract is suits to build signature locating program. ImageMagic is good at dealing with bitmap images, so it can be used to convert pdf to image file. OCR Tesseract converts printed text to machine encoded text well. It also provides different types of page segmentation method. Since all the documents in the workplace have different format, having decent page segmentation is critical for locating where the text is. In addition, OCR Tesseract can recognize more than 100 different languages. For this project, we only focus on dealing with English document, but for later update, it is good to support different languages.

6.3 What It Can Do

The alpha version of our signature locating program can convert pdf to png, do page analysis, convert image of text to machine encoded text, search CUSTOMER SIGNATURE, and save the signature box as png image file. Since this is alpha version, there are two major issue. First problem is the search method. For locating the signature box, we search word CUSTOMER SIGNATURE from converted document by tesseract OCR. However, since many documents have different signature box format, we need to come up with the search word list to work on different types of documents. Also, when signature is signed over the search word, program have difficulties locating the signature box.

The second limitation for our alpha version program is that it provides rough location of signature box. Since we locate the signature box by searching word, we do not know exact location and size of the signature box. Therefor, we manually set variable for calculating location and size of the signature box. It locates the signature box with many noises.

6.4 Possible Solutions

6.4.1 Search Method

It is difficult to locate the signature by searching certain word in the document. The performance of the program can be improved by making decent search list. Also, when program searches the word, rather than finding exact word, find the similar word in the document. For example, when we search for CUSTOMER SIGNATURE, we can take word such as CUST@MER S!GNATURE as valid input.

6.4.2 Finding Accurate Location

The possible solution for this problem is searching location of the horizontal line. Signature box is consisted with two parts which are text and line. If we are able to search word and locate it, we can search for closest horizontal lines to location of searched word. After locating the signature sign line, software can cut the slightly below and above the sign line to locate more accurate signature box.

7 CONVOLUTIONAL NETWORK

In order to assess whether a certain region contains a signature or not, our team decided to utilize a convolutional neural network. Convolutional neural networks are a specific type of neural network which take thousands of inputs relating to specific objects, and start learning specific features contained within these. Convolutional neural architectures are specifically designed to cater to object detection, which made them a perfect match for our given task of recognizing signatures.

The first thing our team had to do was familiarize ourselves with what exactly a convolutional neural network was. Convolutional neural networks take an image as an input, and return a percentage estimation of what objects are contained within that image. They do this through a series of convolutional blocks, which are typically comprised of four layers: a convolutional layer, a pooling layer, a ReLu (rectified linear units) layer, and a softmax regression layer. The convolutional layer applies a learned feature filter to the image. This feature filter will generate higher values when applied to an area that is similar to the feature learned while training. The pooling layer looks at smaller regions of around 2-4 pixels and selects the highest value, discarding the others. Neural networks are usually fully connected, but connecting every pixel in an image to every neuron in an another layer, is what many define as computational explosion. The convolutional layer plus the pooling layer allow the network to be much less connected while still being able to achieve satisfactory performance. The ReLu layer makes any negative value into a 0. Many times after training a network will develop negative synapses, also called weights. These negative synapses can be turned into 0s deeming that area of that specific filter irrelevant. By doing this there is no loss in performance but there is a gain in computational productivity. The softmax regression layer applies a function to the value calculated through the other layers, and computes a percentage estimation related to the labels the objects were given during training. Convolutional neural networks like many other neural networks learn through back propagation. The idea is to craft a cost function evaluating how wrong or how far away we are from a correct answer. One this relation is modeled it

is evaluated towards a minima and the weights in the network adjusted appropriately depending on whether certain areas activating generated a correct answer or an incorrect answer.

Once we were familiar with the architecture of convolutional neural networks the next step was initializing our own. There are many templates out there for CNNs that cater to a wide ranging complexity of tasks. In order to evaluate signatures efficiently a convolutional neural network does not need to be very complex. The neural architecture we chose consisted of one convolutional layer, one pooling layer, one ReLu layer and 2 softmax regression layers. Originally this convolutional took inputs of type ubyte, since it was modeled to learn to read handwritten digits, and used the MNIST database to do so. After correctly implementing it and testing it with this subset, the next step was back engineer its input system to be fit our subset. Also this network was designed to be trained and tested without being able to load specific kernels after running the program. In order to fix this, since our network must be able to use trained kernels even after the program is done, we fashioned a kernel loading module and integrated it into the already existing program structure.

In the current version of our program, the network is able to be trained, save the results of the training, and then be ran with a single picture returning an assessment of whether a certain image is a signature or not. Given that we have a rather small subset of signature to train with, it is not yet up the standard of correctness that we would like it to be. It currently functions at 52 percent to 56 percent accuracy with a training set of 300 signatures, and we would like to raise those numbers to 70 percent to 75 percent within this next week. Our final goal is for the network to have over 90 percent accuracy, which will require a subset of 2000 - 2500 signatures.

8 Weekly Summary of Winter

8.1 Week 1

8.1.1 Activities

For our first week back, the primary focus was establishing what our new schedules looked like as well as determining what meeting times needed to be adjusted. In addition, we received instruction from our teachers that the usual physical meetings for the class would be largely removed to facilitate more time to work on coding. As of this point in time we already have file conversion functioning and isolated potential problems with the other two components. There was a very short update at the end of the week regarding the difficulties being had for OCR Tesseract setup and that became a focus for the team.

8.1.2 Problems

The primary source of conflict for this week was the change in scheduling. Where before we had largely similar time commitments we now have a very small window of available time to do meetings. This not only meant making inter-group meetings difficult but also meant we needed to change our meeting times with both our client and teaching assistant. The OCR issue stems from the fact our normal workplace being the school servers lacked the required installations to sufficiently run the program we needed to make.

8.1.3 Solutions

For this first week we managed to re-establish communication with each other and determine future meeting times. This included the changing of our meeting times with our client to Thursdays at 2:30pm. Our TAs schedule also had to change so that portion of the scheduling had to wait until next week to receive word of his new times. Work also started on finding alternative methods of installation for OCR Tesseract that focused on the creation of a local virtual machine to run it.

8.2 Week 2

8.2.1 Activities

Week two was largely focused on finalizing the schedule changes that had occurred, this also meant we solidified our meeting time with the TA for 1:00pm. To handle the lack of physical meeting times for this term we constructed a new schedule for Monday, Wednesday, and Friday from 3:00-4:00pm to have our group meet to keep on track. Due to scheduling errors we did not have time to go over the neural network this week, but work did continue on setting up OCR.

8.2.2 Problems

The primary problem for this week came in the form of OCR setup again. Efforts to complete setup through a personal VM ended in failure, several version mismatches as well as limitations on the VMs we chose meant it did not function correctly. Another method of installation that was underway included dual booting our machines into Ubuntu and installing it from there. Problems did surface when attempting this however as a result of BIOS conflicting issues that prevented Ubuntu from launching correctly.

8.2.3 Solutions

The virtual machine concept was ultimately placed on hold, two different variants of VMs had been tested and both received problems at differing stages. The dual boot ultimately worked out, currently requiring changes in BIOS settings on startup but was able to launch correctly. From there we learned that the most current version of OCR has severe setup issues and a previous version would have to be used for our project.

8.3 Week 3

8.3.1 Activities

This was one of the more productive weeks for our group, neural network was successfully instantiated, and OCR Tesseract was now up and running. It was determined that a traditional network would be insufficient for what we needed and that a convolutional one would be more effective. This new network structure was able to correctly handle several test samples and return a strong accuracy rating. From here the primary concern with the network was adjusting

its data types to handle our images and then providing it sufficiently large data sets to train off of. As for OCR it was now correctly running and able to run basic word recognition on sample texts. With that accomplished the goal for OCR was to now correctly detect where a signature is located on a form.

8.4 Week 4

8.4.1 Activities

The network was successfully shared and functioning across all machines at this point in time. We have correctly isolated an issue regarding OCRs reading capacity when subjected to forms with extensive use of lines to segment, OCR will try and read them. Setup for OCR has been postponed for this week in an effort to prioritize more working components to showcase to our client. Weve also established that the difference in OCRs use of Leptonica completely isolates it from what OpenCV is doing. This will assist in modularizing the program but also means that linking them may be more difficult than we originally intended.

8.4.2 Problems

OCR will require modifications to permit the reading of forms with lines present within them as well as the changing fonts and line spacing. In addition, the modularity of the program may be too extreme, concerns have been expressed towards our ability to correctly link the OCR and network together correctly. Lastly, while it was postponed for more pressing concerns, the need for OCR to be running on more than one machine is critical.

8.5 Week 5

8.5.1 Activities

Most of this week was focused on researching how the network operates as well as determining a workaround to the problems previously noted for OCR. The network has been constructed and can handle a given data set but we are not sure yet on how the labels should be constructed and how to convert the images to a readable format for the network. Plans have been mapped out for preparing our poster and progress report, will be receiving input from client to confirm nothing they want secret is revealed. It was also decided that our GitHub repository needed to be restructured to help with organization and legibility.

8.5.2 Problems

With the alpha deadline approaching our primary concern was ensuring the link between OCR and the network could be established and that OCR can read the forms with sufficient accuracy to call an alpha state. In addition, communication with client needed to be maintained to ensure that our poster was acceptable and to the standard that CDK Global expects.

8.5.3 Solutions

While not fully tested, OCR workarounds have been found for reading the document correctly. Effort was put into determining the difficulty of removing the lines, which was deemed to great, and the use of a new function call we had previously missed. The new function call provides a line segmentation that can circumvent the previous problems and manage to isolate the signature window. This does require a few manual inputted values at present so that is now a new concern.