UAV Crack Detection and Classification

Client: Muhammad Ahmad Siddique

Advisor: Dr. Halil Ceylan

Team Members:

Lauren Arner – Project Manager

Benjamin Ferreira – Testing Lead

Madi Jacobsen – Data Lead

Ian Seal – Reporting Lead

John Schnoebelen – Software Developer

Jack Temple – Lead Software Developer

Team Email: sdmay20-13@iastate.edu

Team Website: https://Sdmay20-13.sd.ece.iastate.edu

Project Overview

- Infrastructure across the United States is in desperate need of repair
- Due to the amount of infrastructure across the country, identifying cracks and areas in need of repair can be very time consuming
- Programmatically identifying cracks will increase the speed of both research and maintenance

Engineering Standards and Design Practices

Software

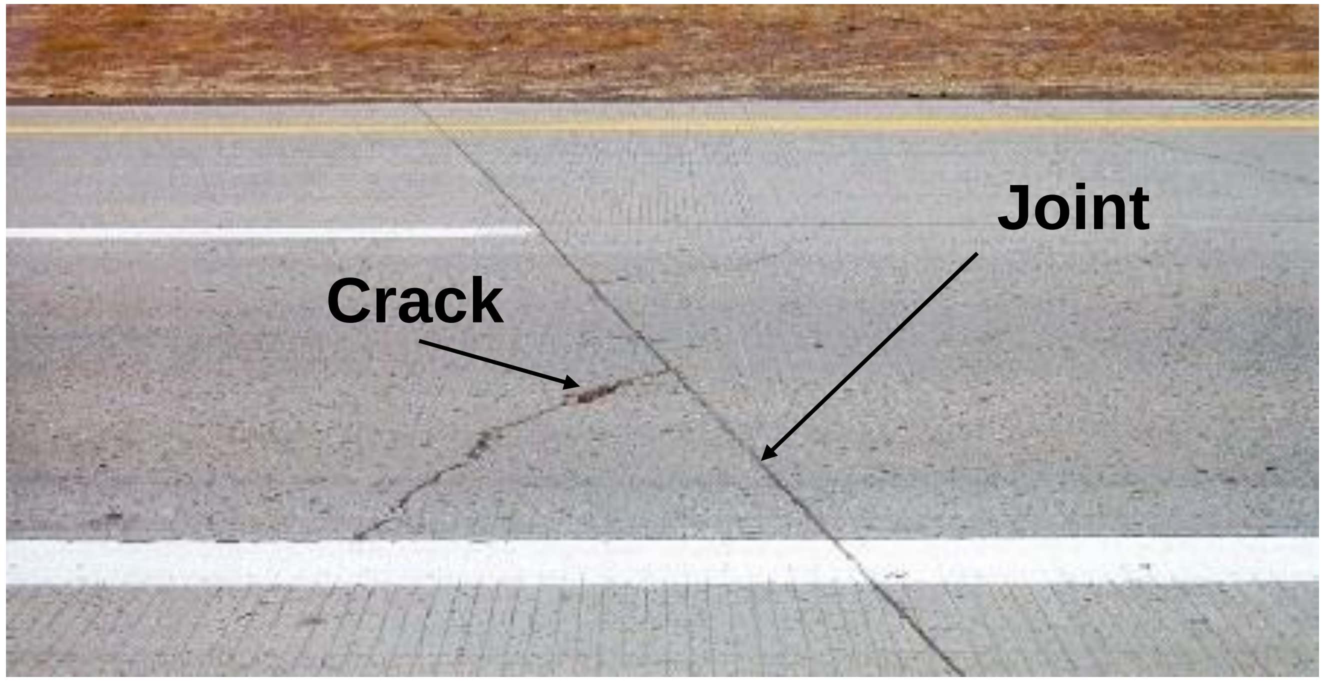
- PyTorch – Python library for machine learning
- TensorFlow – Python library for machine learning

Standards

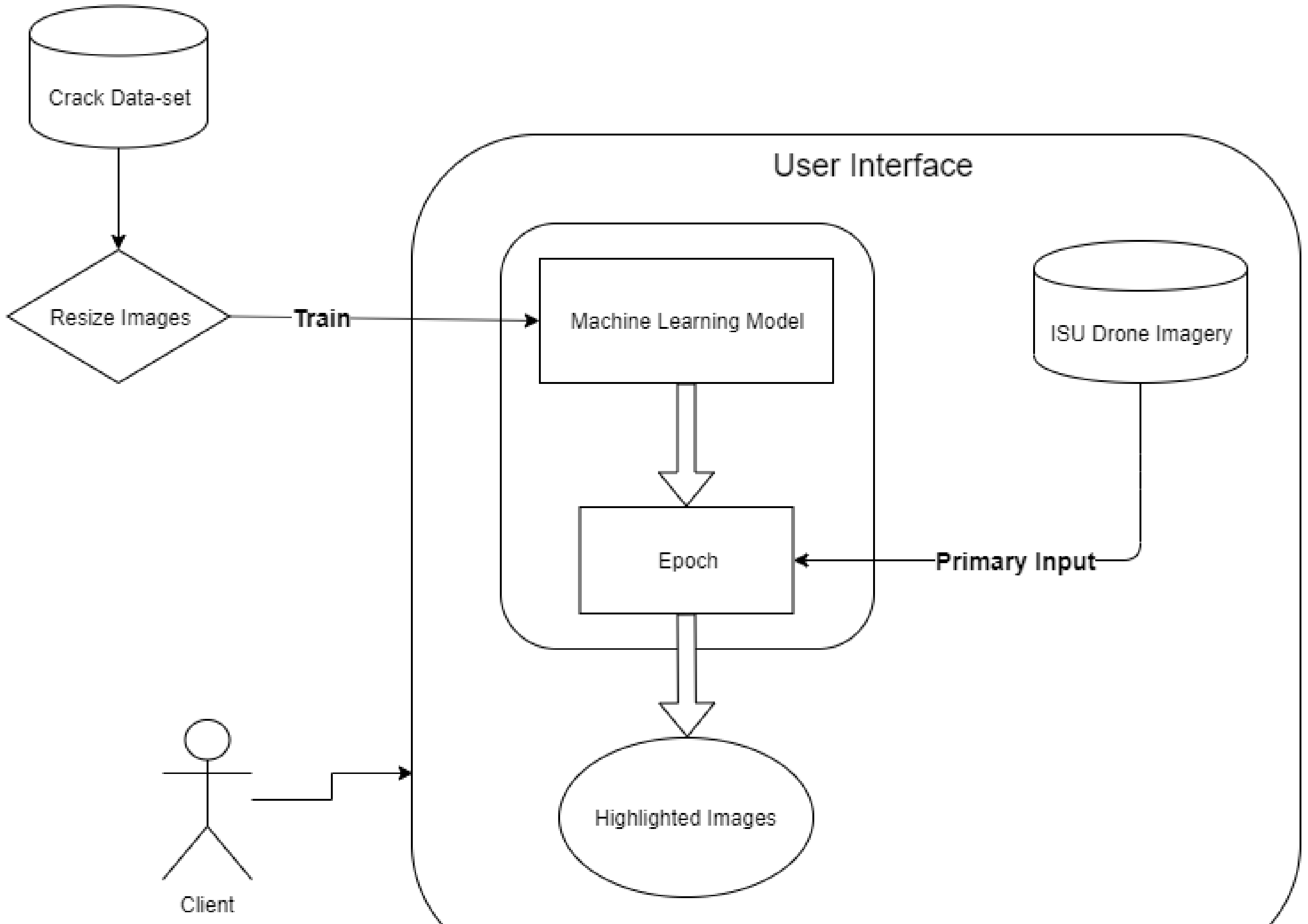
- IEEE 12207 - Software life-cycle processes
- IEEE 29119-2015 - ISO/IEC/IEEE International Standard - Software and systems engineering--Software testing

Functional Requirements

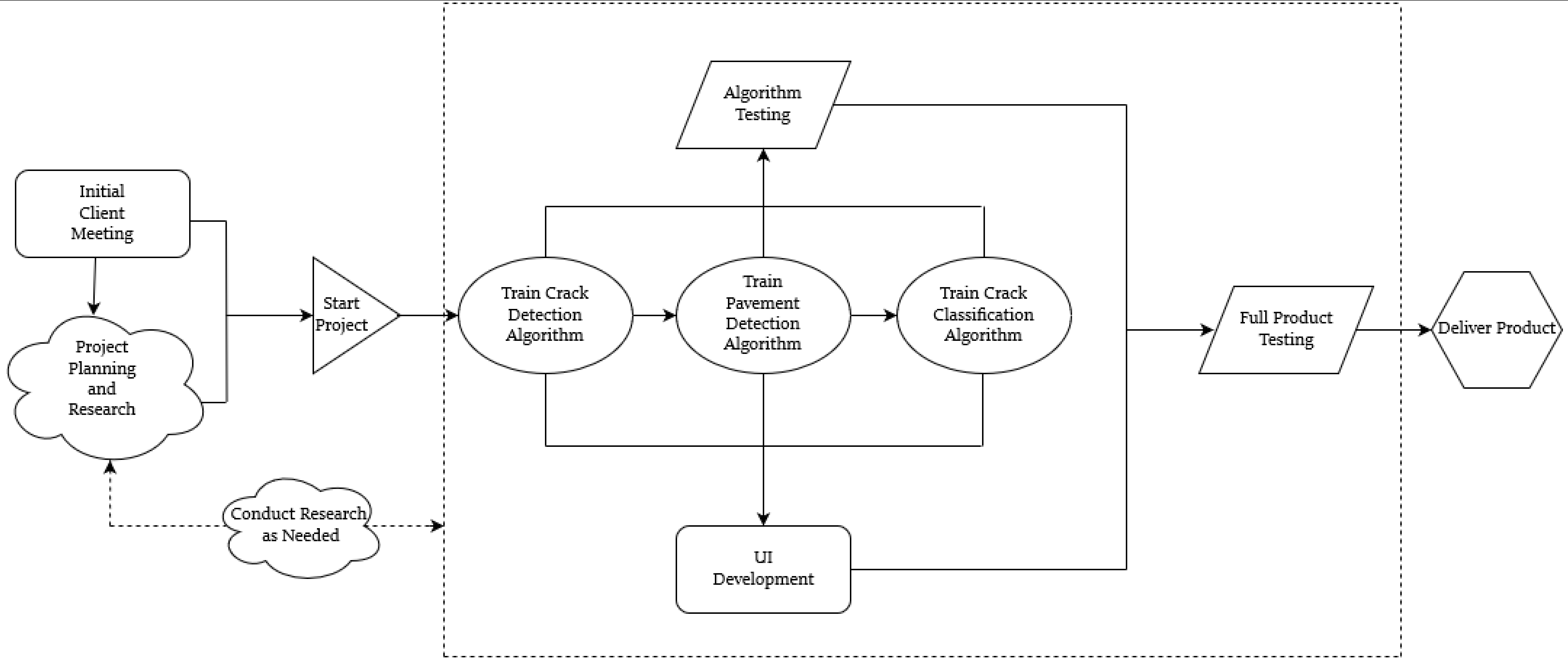
- Identify 80% of cracks accurately
- Ability to identify the difference between joints and cracks
- 80% correct classification of crack types
- Correct analyzation of cracks despite obstacles in images
- Ability to accept large quantities of images
- Output images (with accurately marked cracks)



Conceptual Sketch



Functional Decomposition


Initial Client Meeting
Project Planning and Research
Conduct Research as Needed
Start Project
Algorithm Testing
Train Crack Detection Algorithm
Train Pavement Detection Algorithm
Train Crack Classification Algorithm
UI Development
Full Product Testing
Deliver Product

Back End Design

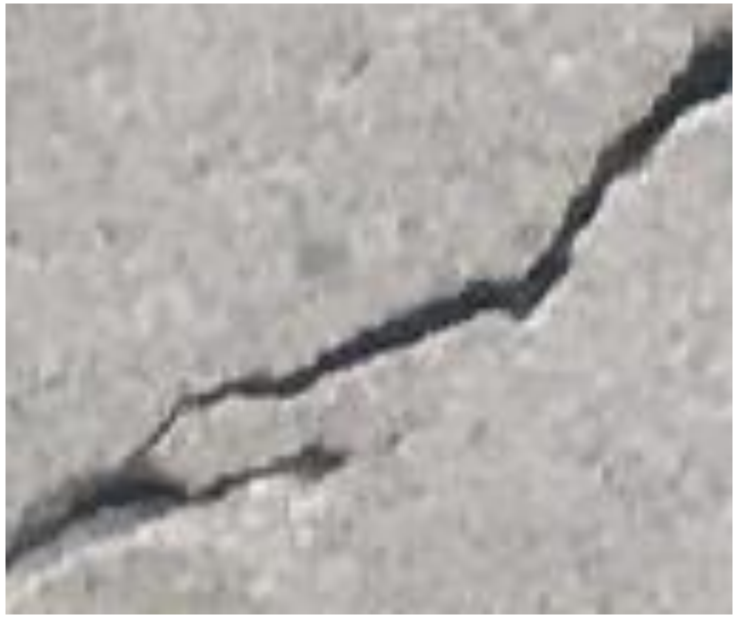
Based on research paper Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks

Breaks image down with each level

40,000 images in dataset

70% are used to train, 30% are used for model accuracy

Dataset Example



Back End Technology

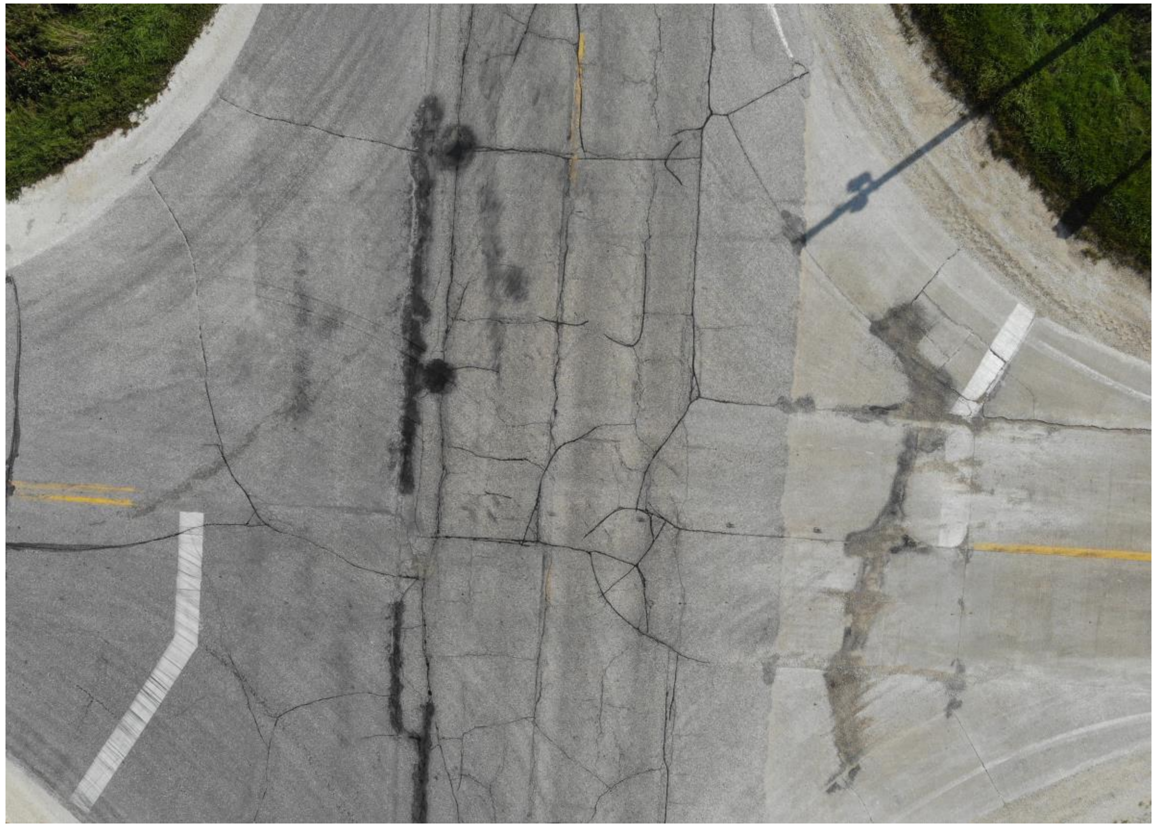
Originally used PyTorch for machine learning

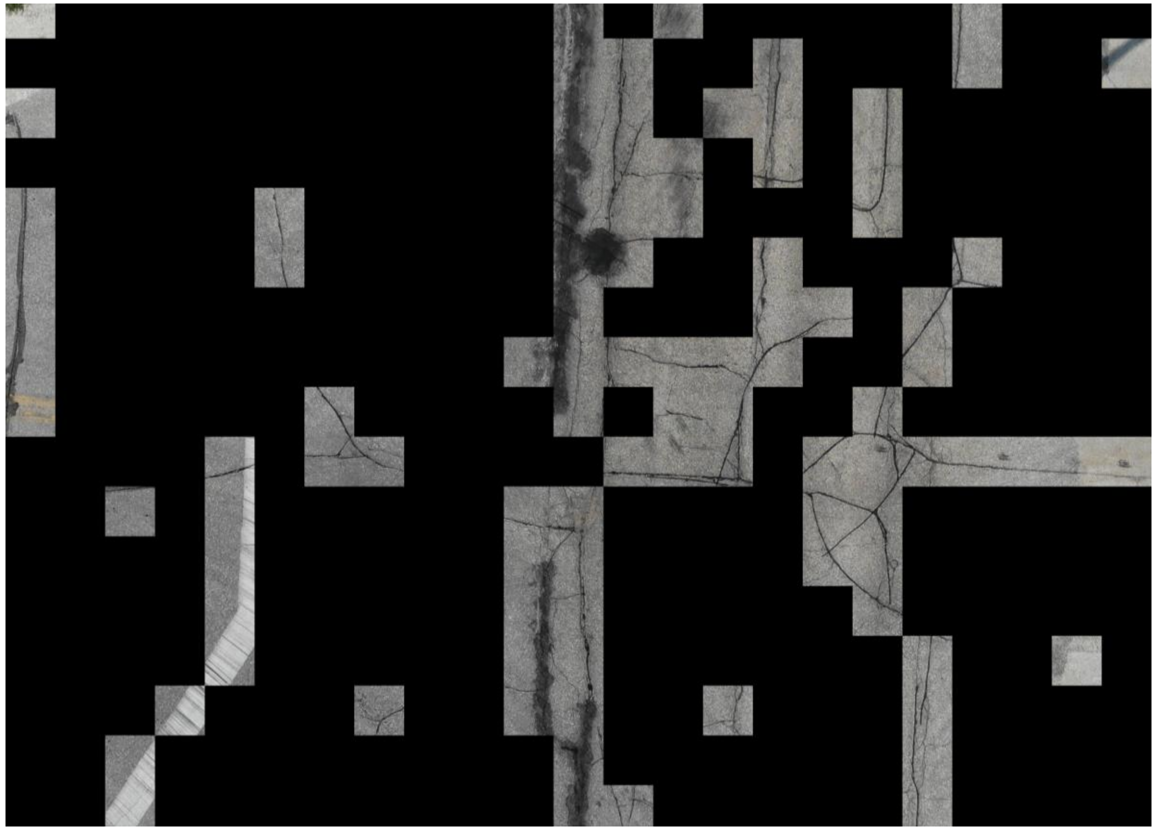
- Open source, lots of documentation
- Ran on HPC for first semester (491)
- Stopped training, HPC wouldn't run project around January

Switched to TensorFlow project

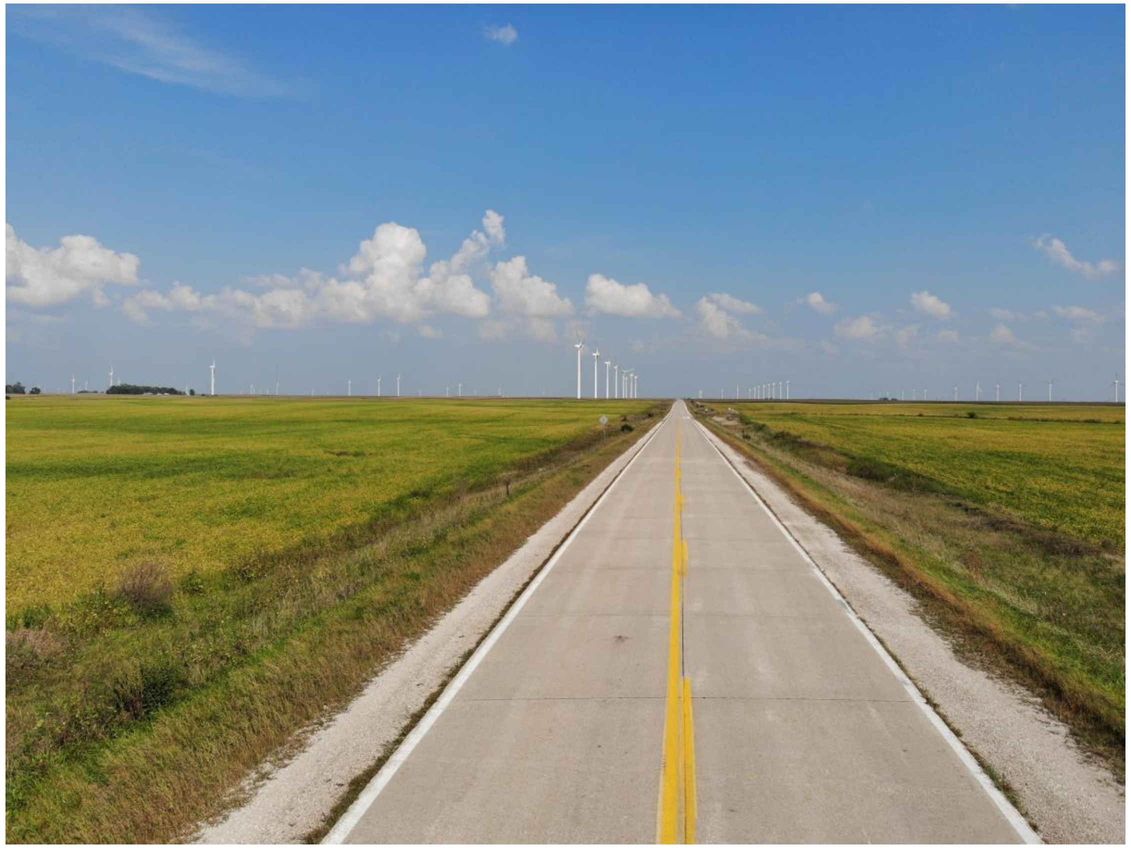
- Similar confidence level as PyTorch
- Project trained easier on personal computers
- All development was still in Python, just using a different library
- Added recording to .csv file

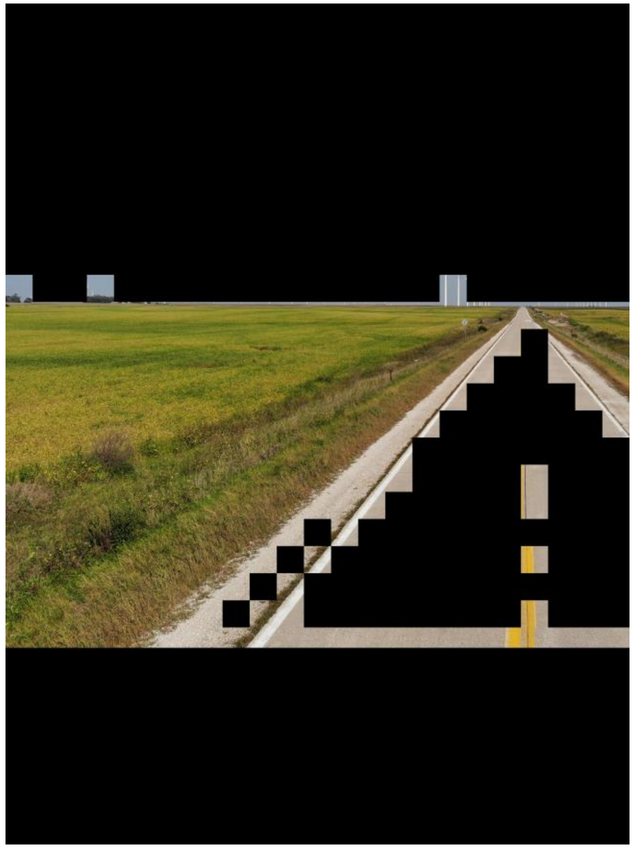
Example of Good Back End Output





Example of Poor Back End Output

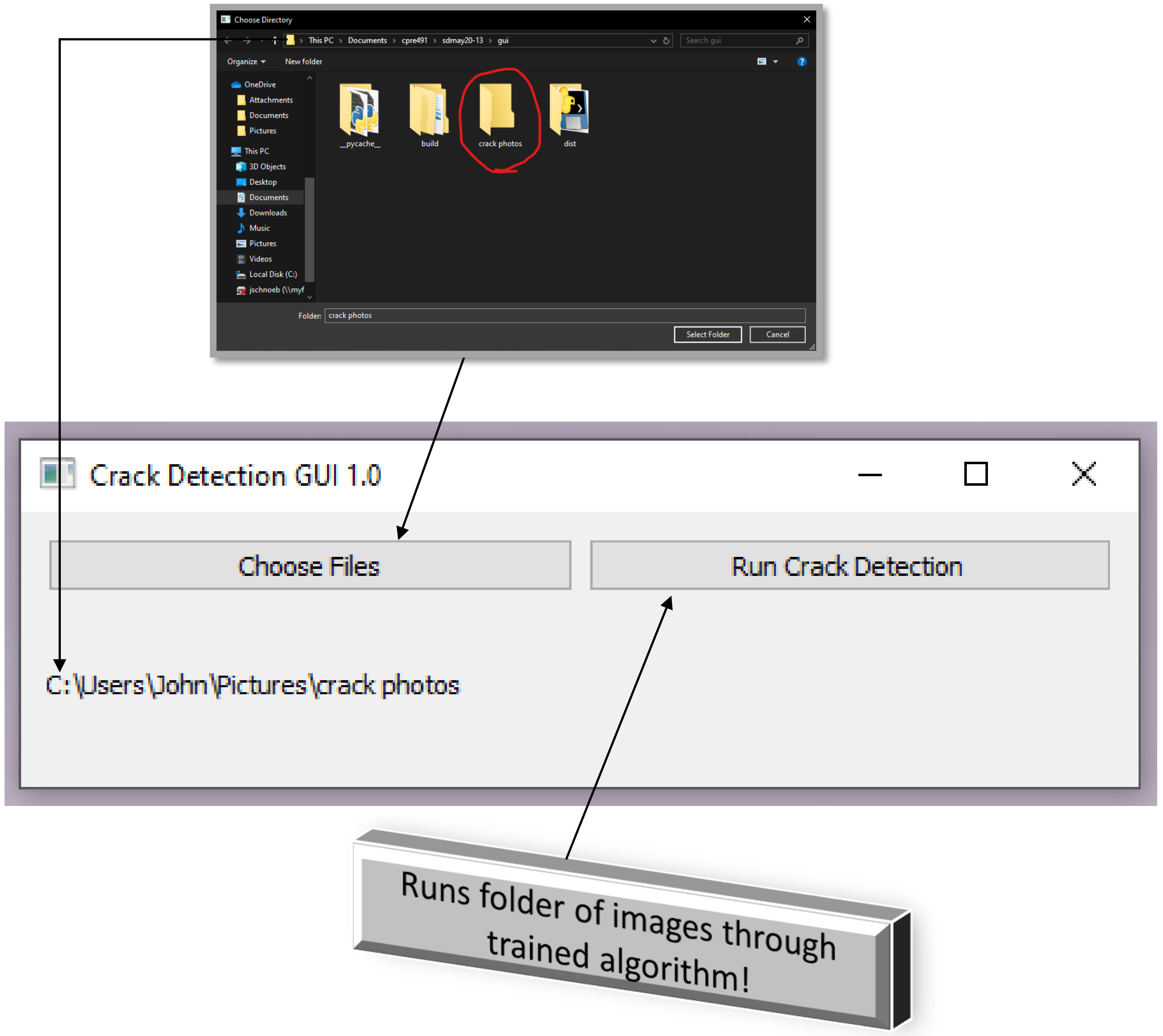




Front End Design

Functionality:

- Choose desired directory of images
- Run images through crack detection model
- Utilize crack detection software without running complicated shell commands
- Output new folder of images to a known location



Testing Process

Functional	Non-Functional
To test for accuracy of crack detection, crack identification, and pavement type identification: • Run program, take output images and compare them to human processed images, use the proper equation to determine true accuracy. • Input images with obstacles and record the sub-accuracy. • Modify program accordingly to achieve desired accuracy.	To test for testability, timeliness, reliability, stability, operability, and learnability: • Create a log and keep track of multiple variables each time the program is run, such as runtime, output success rate, actual accuracy rate, if the program output images, and if the program crashes • Use the log to calculate averages To test for maintainability, customizability, efficiency, and evolvability: • Make sure code is commented and understandable to other industry professionals

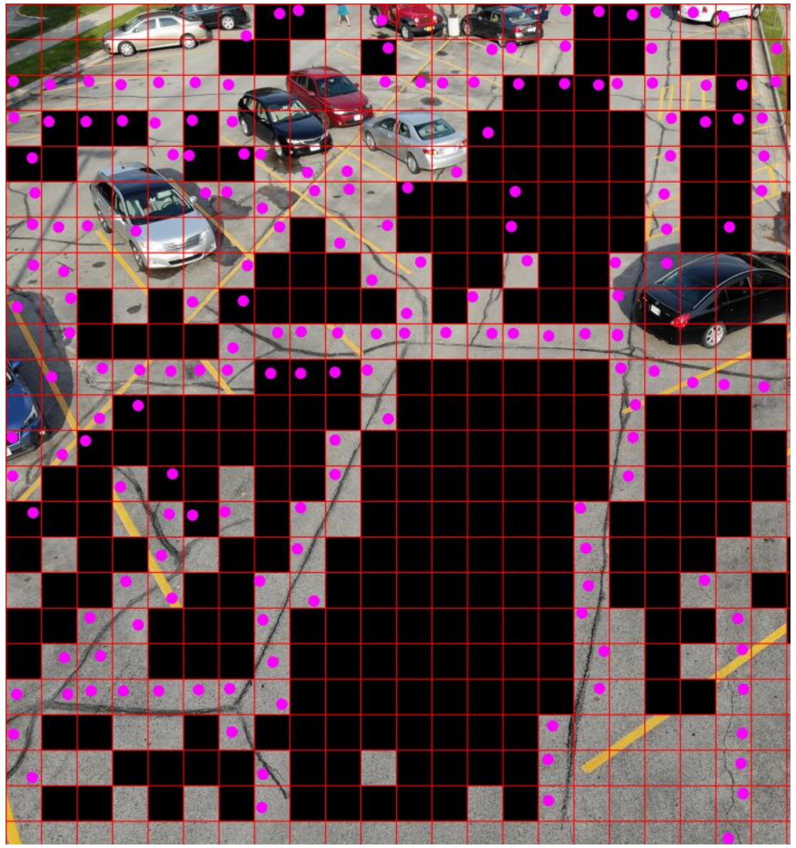
Data analysis

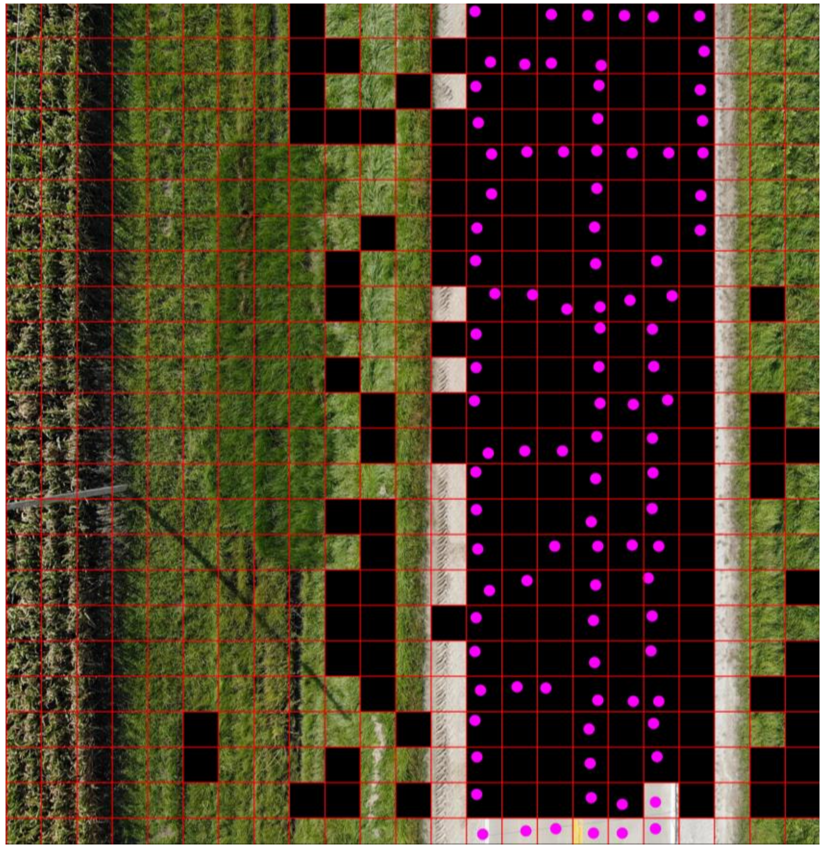
Crack Detection Accuracy (avg): 47.74%

Max Accuracy: 72.46%

Min Accuracy: 23.55%

Max Accuracy VS Min accuracy





Functional Results

Self-constrained requirements	**Actual Functionality**
Identify 80% of cracks accurately	47.74% average accuracy
Ability to identify the difference between joints and cracks	Functionality not added due to unreached basis of satisfactory crack detection accuracy
80% correct classification of road type (asphalt vs concrete)	Functionality not added due to unreached basis of satisfactory crack detection accuracy
Correct analyzation of cracks despite obstacles in images	Obstacles such as cars or poles are detected as false positive. This means we could continue to train the program with images of these obstacles to increase the actual accuracy rate.
Ability to accept large quantities of images	The program is able to process large quantities of images as long as it is being run on a system with proper specifications, like Iowa State University's High-Performance Computing (HPC) service.
Output images (with accurately marked cracks)	The program successfully outputs the images it has processed, and the image has markings indicating where a crack was not detected.

Non-Functional Results

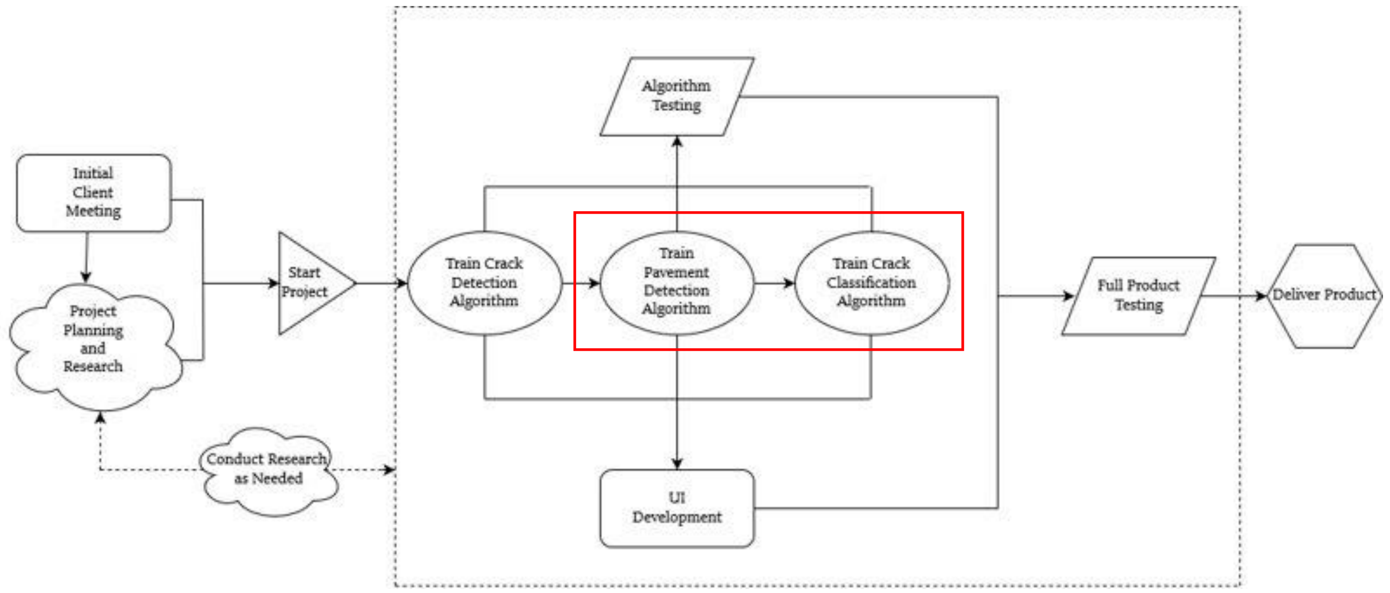
Self-constrained requirements	**Actual Functionality**
Testable Timely Reliable Stable	The program proves to be testable and timely but does struggle with reliability and stability.
Maintainable Customizability Efficient Evolvability	Code is commented and understandable to other industry professionals and able to be built onto.
Operable Learnable	GUI function allows non-technical users to run the model with little training.

Conclusions

- When an obstacle, such as a car or pole, is tested, it will typically come back as a false positive (preferred over false negative because of how the program is trained).
- One obstacle we did not foresee was grass being analyzed as a crack. This highly skews the accuracy rates in some images due to the altitude the images were taken. Altitude determines if the image is a majority grass or a majority road in many cases
- We expected the skew of an image due to the angle it was taken to have a dramatic impact, but numbers so far show very little difference when the image is straight on compared to askew. This is not the case for images taken down the horizon.
- The program tends to do dramatically better when identifying cracks on asphalt roads compared to concrete. A 56.61% accuracy rate compared to a 38.86% accuracy rate, respectively. Some concrete roads have a grooving throughout them that causes the program to return false positives.
- Overall, this program is not ready for use by the clients due to the low accuracy rate, but the UI makes it easily learnable for the client once the program detects cracks, crack type, and pavement type more accurately.

Future Prospect

- Train pavement detection
- Train crack classification
    - Identify a way to distinguish between crack types (joint and crack)
- Polish the User Interface
    - Deploy as an executable desktop application


Initial Client Meeting
Project Planning and Research
Conduct Research as Needed
Start Project
Algorithm Testing
Train Crack Detection Algorithm
Train Pavement Detection Algorithm
Train Crack Classification Algorithm
UI Development
Full Product Testing
Deliver Product

Team Member Contribution Breakdown

Team Member	**Contributions**
Lauren Arner	Project Manager, Data Analysis
Benjamin Ferreira	Front End
Madi Jacobsen	Data Analysis
Ian Seal	Back End
John Schnoebelen	Front End
Jack Temple	Back End