

C964: Computer Science Capstone: WispWatch

Task 2 parts A, B, and D

Trayer Harvey
001341276
July 2022

Table of Contents

Part A : Project Proposal for Business Executive	3 to 6
A.1 Letter of Transmittal	3
A.2 Project Recommendation	4
A.2.a Problem Summary	4
A.2.b Application Benefits	4
A.2.c Application Description	4
A.2.d Data Description	5
A.2.e Objectives and Hypothesis	5
A.2.f Methodology	5
A.2.g Funding Requirements	6
A.2.h Data Precautions	6
A.2.i Developer's Expertise	6
Part B : Project Proposal	7 to 11
B.1 Problem Statement	7
B.2 Client Summary	7
B.3 Existing System Analysis	7
B.4 Data	8
B.5 Project Methodology	8
B.6 Project Outcomes	9
B.7 Implementation Plan	9
B.8 Evaluation Plan	9
B.9 Resources and Costs	10
B.10 Timeline and Milestones	11
Part C : Application – List of Files	12
Part D : Post-Implementation Report	13 to 30
D.1 Business Vision	13
D.2 Datasets	14 - 15
D.3 Data Product Code	16 - 17
D.4 Hypothesis Conclusion	18
D.5 Effective Visualization and Reporting	18 - 22
D.6 Accuracy Analysis	23
D.7 Application Testing	23
D.8 Application Files	24
D.9 User Guide	25 - 32
D.10 Summation of Learning Experience	33
Part D : Sources	33

Part A: Project Proposal for Business Executives

A.1 Letter of Transmittal

July 15, 2022
Iam. T Boss, CEO
PrettyPerfectPrinting LLC
123 Layne St.
Metropolis City, Xy, 90006

Boss,

We at PPP have felt enormous pride in seeing this garage 3D printing startup grow into a veritable tour de force of the prints-on-demand industry. And as the head of the 3D Printing Mechanical Technicians Team, it's an amazing feeling to see our original pair of 3D printers grow into a whole fleet. But we are not without our growing pains, and a growing fleet means our technicians are struggling to keep up with all their duties while monitoring the printers for errors, shorts, or fires.

In an effort to combat this, my advice is two-fold:

- 1) Let the technicians focus on their more important duties, and
- 2) Leave printer monitoring for an automated system

This brings me to said automated system: WispWatch. A machine learning-powered image analyzer, designed to detect fire in pictures, videos, and live-streams. As of writing, monitoring for fire accounts for a large portion of our technicians attention, when fires rarely occur, wasting precious human resources. But our printers are already equipped with OctoPrint-enabled webcams so technicians can view the status of many at once. The benefit of WispWatch is that an instance of it can be run for every OctoPrint stream we have, all for a flat cost, without having to hire anymore technicians or purchase new hardware. You don't even have to worry about the back-end, I threw together a prototype using google cloud's vision API and trained a custom model based on over 1000 images, accessible anywhere with python and WiFi.

Ultimately, the cost of renting a week's worth of server time for google's model, which can then be downloaded and run edge-wise, totals to less than \$500. The app should be maintained by a dedicated software engineer once it grows big enough, costing between \$4000 and \$8000 a week. A single Technician costs around \$1000 a week, but we have over one hundred printers now, even ten technicians would be overwhelmed by the responsibilities of ten printers each.

Due to my background in computer science and software development, I'm qualified to expand my role to take on the responsibilities of Software Engineer for PPP if my compensation is adjusted fairly. My capstone in college was in machine learning so this task is well within my skill-set, unless the company opts to hire a new grad so I can focus on the mechanical tech.

Please take a look at my included files and let me know what you think,
-Trayer H, Technician Team Lead

Part A.2: Project Recommendation

A.2.a - Problem Summary

- Summarize the project.
- Describe the setting and why the project is needed.
- Briefly describe how the project meets the business' (or organization's) needs.
- Describe what will be delivered and achieved.

The project is an effort to take the hardware and software that we already have installed on our systems (OctoPrint and webcams) and couple it with a machine learning model that can take information from said systems and solve problems (like detecting when a printer has caught fire). Our printing lab currently houses over 100 printers, and it's a waste of technician resources to monitor them all with humans, especially at the rate we're growing. We need a scalable, affordable, automated solution to fit our business needs, and I believe WispWatch is that solution. This delivered app can be deployed for every printer, automating our monitoring and safety protocols so that we can continue to grow with the assurance that if a flaming emergency happens on any of our systems, we'll be alerted .

A.2.b - Application Benefits

- Describe (in more detail than above) how the project meets the business's (or organization's) needs.
- Describe how the business (or organization) will benefit from implementing the proposed solution.

Our organization faces a two-fold problem as we continue to grow out of bounds: safety and cost. Too much growth at too little cost is unsafe, and that's the territory we're in as we expand without matching our safety protocols. If we hired a dozen human technicians, that would be safe, but we'd no longer be turning a profit. The sweet spot is an automated vision solution. This vision solution is more affordable at scale than the manual alternative, and with high enough vision accuracy, the margin in safety can be even better than a living monitor (computers can't doze off). This will benefit PPP immensely as we can grow without worry of safety issues, and our profits will remain high due to the lack of human labor involved post-deployment.

A.2.c - Application Description

- Provide technical details on how the application will solve the problem.

The application consists of a python front-end and a Google Cloud Vision Model back-end. Google Cloud offers a machine learning platform that can be rented out affordably to train a model which can then be downloaded and deployed locally. With a large enough dataset, images of fire and not-fire, this model can be trained to address our needs. Namely, the front end (PySimpleGUI) can be used to grab our live video streams from OctoPrint, grab images periodically, submit those images to Google Cloud via their Python API, and get back a prediction on whether the image contains fire or not (along with an accuracy assessment). Using this prediction, we can control external hardware and trigger a cut to the printers power, and/or activate a fire extinguisher. (For the prototype the text just turns red)

A.2.d - Data Description

- Identify the origin of the raw data.
- Describe the type (nominal, quantitative, etc.) and structure of data.
- Identify dependent and independent variables.
- Describe any anomalies (e.g., outliers) and limitations.

A large majority of the data comes from a Kaggle image set posted by Ahmed Saied in 2020. This data is comprised of 755 images of outdoor fire and 244 images of non-fire. The data is marked qualitatively by being in separate folders. I added an additional 100 images of 3D printers (from Google Images, manually downloaded) to the non-fire data set, as well as a few outliers (a picture of a toy sword) to invoke randomness.

To process this data, the images had to be uploaded to google's cloud storage, where they were sorted based on their independent variables such as file names. I then manually assigned the dependent, nominal, "Label" variable to all images, ending up with 582 images of fire, and 344 images of non-fire.

A.2.e - Objectives and Hypothesis

- Identify and describe desired outcomes of the project.
- If applicable, state a hypothesis.
- If applicable, state the desired prediction accuracy.

The goal, ultimately, would be to demonstrate the efficacy of both the concept of a machine learning app as a safety tool, and our ability to program/train our model in-house. With the state of machine learning being nearly ubiquitous in the tech sphere, I hypothesize that there are enough resources (and our company possesses the requisite skill and capital) to develop such an app, and that it will prove itself a worthy safety solution, on par with cost-compared human solutions.

However as we scale, and 100 printers becomes 1000, we'll need to ever increase the accuracy of our model, which can be done by training the model further with the images we get from our webcams. This could affect the long term results of our hypothesis, if we fail to maintain funding and training, our machine learning-base solution will no longer work.

A.2.f - Methodology

- Identify the methodology used to develop and implement the project.
- Describe why the chosen methodology is appropriate for the project.
- Provide an outline of the project methodology describing each phase.

As with the rest of PPP, we'll be developing this product using Scrum methodology. We use scrum due to the application of sprints, with each sprint covering a different phase of the project. We use phase of of planning, implementation, deployment, and review, looping back around after feedback has been gathered during the "review" phase to strategize in a later "planning" phase. Planning involves laying out timelines and prepping data pipelines, implementation means bug fixes, model training, and feature additions. Deployment involves making sure the app works on all machines and not just the dev's and installing it. Reviews are conducted on the process as a whole and data can be gathered from the Google Cloud Model Dashboard to determine the direction to take training.

A.2.g - Funding Requirements

- Describe the project's funding requirements, including environment, personal, licensing, and tools.
- The funding amount should match the letter of transmittal.

There are a few options regarding funding depending on our choice of hosting. Google charges for storage and for cloud model deployment, so I propose we pay for google storage, but we only use the cloud model when we're training. Frequency can be argued, but maybe once a week. The remainder of the time, we can download and use an offline model post-training. The cost of hosting varies based on data amount, we'll start with their \$500 per month plan, and the cost of a software engineer to maintain the project will be around \$10,000. Due to the cloud-based nature of the training, we can use company laptops for development and save on tool fees (PyCharm is a free IDE software that can be used)

A.2.h - Data Precautions

- Identify any sensitive or protected data.
- If applicable, review the general guidelines related to working with that data.
- If applicable, describe necessary precautions which will be taken.
- If either of the above is not applicable, explain why (public datasets, such as those from Kaggle.com, have no such restrictions).

All of our data comes from public data sets, either Kaggle.com or google images, and as such, have no sensitivity restrictions

A.2.i - Developer's Expertise

- Describe the developer's (you) qualifications, e.g., academic training, professional expertise, experience, etc. Using future qualifications, such as your WGU degree in Computer Science, is acceptable.
- Relate the listed qualifications to the needs of the project.

To succeed, this project needs to be handled by an engineer who has familiarity with machine learning, and is good at problem solving on the fly under high-stress scenarios (the job involves fire, after all). As a computer scientist with capstone project relating directly to image-based machine learning, I'm capable of taking on the role should we not find anybody.

Part B: Project Proposal

B.1 - Problem Statement

- Describe the problem

As our company has scaled, our number of 3D printers has become unmanageable. Printers need to be monitored 24/7 while in operation for safety reasons, (e.g. in case there's a short and a fire) but our technicians have better things to do than watch a machine. As we grow even further, it's not feasible to continue spending human resources on every machine. The problem is that we need a cost effective, scale-able, accurate solution to our 3d printer safety monitoring issue.

B.2 - Client (or Customer) Summary

- Describe the client (or customers).
- Describe why your proposed application (a data product in the task directions) will resolve the problem successfully.

This product is not intended for consumer use (although that can be discussed later down the line), and as such our company will be the only clients. The users will be the technicians, or whomever we hire to help assist the technicians. Technicians already have access to live-stream video feeds of all the printers for quality monitoring, and we aim to use the existing infrastructure to support our software. Our machine learning-based data product is more cost-effective than human monitors, scale-able as long as we have funds, and our current data suggests a 99% accuracy on our current data set, thus satisfying our problem solution criteria

B.3 - Existing System Analysis

- Describe (if any) what application(s) or tool(s) the client currently uses.
- Describe the shortcomings of this current technological environment, i.e., what your solution is needed.

The proposed solution has a front end written in Python and was built on it's accompanying IDE PyCharm, both of which are open source and free. The back-end is hosted on Google Cloud and currently uses my credentials and funds to log in and function. At the moment this means our system is limited by our internet availability, as it has to access the cloud. This can be remedied in the future if need be by downloading the model we train off google's servers.

B.4 - Data

- This section should include (where applicable) descriptions of:
 - o The raw data set.
 - o How data will be collected, processed, and managed throughout the application development life cycle: design, development, maintenance, or others.
 - o How data anomalies, e.g., outliers, incomplete data, etc., will be handled.

Currently the model has been trained by uploading over 926 images from Kaggle and Google Images, which were subsequently labeled as “fire” or “nonfire”, and used as our data set. Currently the product is a demo, and as such, uses a model that has been trained and we don't intend to train further at this time. However, should we choose to harvest more images for training use, the application creates a file labeled “current_frame.jpg” for every image analyzed. If the model predicts a response which we, as humans, don't think lines up with our expected result, the user can access said image, and add it to our image set to be used with the next round of training.

B.5 – Project Methodology

- Describe an industry-standard methodology to be used to develop and (if applicable) deploy your application.
- Describe the planned development of your application in each phase of the methodology, e.g., analysis, design, etc.

As mentioned in A.2, we'll be developing this product using Scrum methodology. Our sprint phases consist of planning, implementation, deployment, and review, looping back around after feedback has been gathered during the “review” phase

1 - Planning

- Define success criteria
- Locate and Prepare Data set
- Secure funding if applicable
- Form Team
- Finalize Timeline

2 – Implementation

- Train model
- Build front-end/GUI
- Merge Systems
- Debug and test as needed

3 – Deployment

- Test app on end user device
- Install
- Brief user on instructions

4 – Review

- Check success criteria and determine success rate
- Gather data on model statistics from google cloud
- Use data gathered to inform requirements for next planning session

B.6 - Project Outcomes

- Provide descriptions of all deliverables. For example:
 - o The finished application.
 - o A user guide.

Our only deliverable is the final python application: WispWatch

B.7 - Implementation Plan

- Provide an outline of how the project will be implemented. This description might include:
 - o General strategy.
 - o Phases of the rollout.
 - o Dependencies.
 - o Details for testing and distribution.

Implementation involves two phases, due to the two platforms our solution is running on: software phase, and hardware phase. Things are rather straightforward from a software perspective: Once the program has gone through the scrum cycle and cleared all of its success criteria, we can push the final update to our main host so our technicians can download said update and install it on their machines.

For our hardware, we simply need to ensure that the user/technician can find the live-stream for any machine they need to view with our app. If errors occur, we may need to debug the cameras monitoring our printers, or check our connection lag if we're streaming from online. In order to maintain code and app quality, implementation should never be done unless specified criteria are checked and approved, otherwise safety issues may arise.

B.8 - Evaluation Plan

- Describe the verification method(s) to be used at each stage of development.
- Describe the validation method to be used upon completion of the project.

Verification happens at each individual portion of app development. The app consists of three features: the front-end, the back-end, and the GUI that ties the two. To test the front end, we must ensure that the python app can handle the three situations it's programmed for: images, videos, and live-streams. If any of these go wrong they must be bug fixed prior to moving forward with merging code.

The google cloud vision model is our back-end, and their online dashboard provides an interface to upload images to the google storage database for labeling and training, and they provide an average precision between 0 and 1. If the precision is lower than our acceptable value, the model needs further training before it can be used for predictions.

To ensure the GUI is functioning properly, we simply see if when we load an image source with fire in it, that image is uploaded to google's servers, analyzed, and the proper value is returned.

Once the project is complete, the entire app can be tested both in and out of production environments, ensuring that even edge cases where fire is visible are still detected, and no errors occur between the GUI and google's API. If the front end loads images properly, the back end predicts images accurately, and the GUI operates as intended to provide the user with a functional tool, then the app is validated

B.9 - Resources and Costs

- Itemize hardware and software costs.
- Itemize estimated labor time and costs.
- Itemize estimated environment costs of the application, e.g., deployment, hosting, maintenance, etc.

Hardware	Cost
Work PC	Free (Already owned)
Webcam for every machine	\$50.00 / unit
Software	
Python	Free (Open Source)
Labor	
Software Engineer, full time	\$4000 - \$8000 / month
Deployment	
Github, internal use	Free
Hosting	
Google Cloud Storage	\$500 / month
Maintenance	
*See Labor	--

B.10 - Timeline and Milestones

- Provide a projected timeline, including start dates and end dates for each milestone (a table is acceptable).

Sprint	Start Date	End Date	Tasks
1 – Planning	08/01/22	08/07/22	<ul style="list-style-type: none"> -Define success criteria -Locate and Prepare Data set -Secure funding if applicable -Form Team -Finalize Timeline
2 – Implementation	08/08/22	08/19/22	<ul style="list-style-type: none"> -Train model -Build front-end/GUI -Merge Systems -Debug and test as needed
3 – Deployment	08/22/22	08/23/22	<ul style="list-style-type: none"> -Test app on end user device -Install -Brief user on instructions
4 – Review	08/23/22	08/26/22	<ul style="list-style-type: none"> -Check success criteria and determine success rate -Gather data on model statistics from google cloud -Use data gathered to inform requirements for next planning session

Part C: Application

Part C is your submitted application. The document only needs to include a list of any submitted files or links.

See D.8 for list of all submission files beyond just the app

App:

python camera project : project folder

Venv : virtual enviroment folder
python.exe

main.py - APPLICATION
predict.py

google authentication json file

icon.jpg
current_frame.jpg

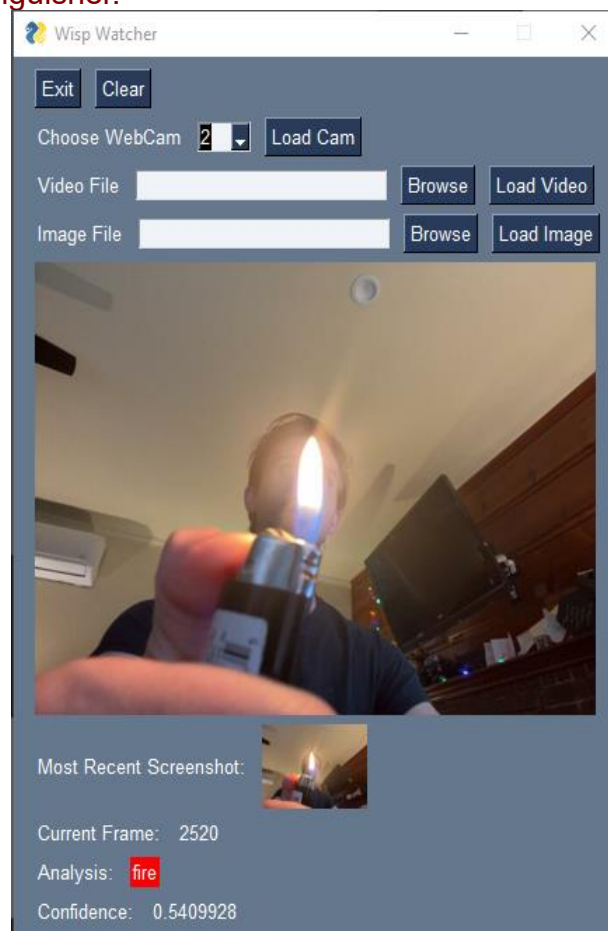
Part D: Post-implementation Report

D.1 - A Business (or Organization) Vision

- Describe the problem.
 - Describe how your application solves the problem.
 - Describe how the user can use your application to solve the problem.
- o Though not required, an example demonstrating how a user will apply the product is helpful for evaluators.

The problem has been elaborated on numerous times in this document, but to put it simply, we don't want to waste resources monitoring for safety hazards like fire when those resources could be put to better use, so we want a solution that is scale-able, cost effective, and accurate. The WispWatch application solves that problem by taking the human element out of stream monitoring, and substituting in a trained image-analyzing fire-detecting machine learning model to do the work.

To function, a 3D printing technician can open our app, select a webcam that is being used to view a 3D printer, and let the app worry about if it's on fire or not. Seeing as how this is a prototype, the app only notifies the user that there's a fire, but in a production scenario, the notification can instead be used to automatically cut the printer from power and activate a servo-controlled fire extinguisher.



The easiest way to test the intended functionality of the app, is to connect to the webcam and point it at fire, or hold a lighter up to your camera (carefully)

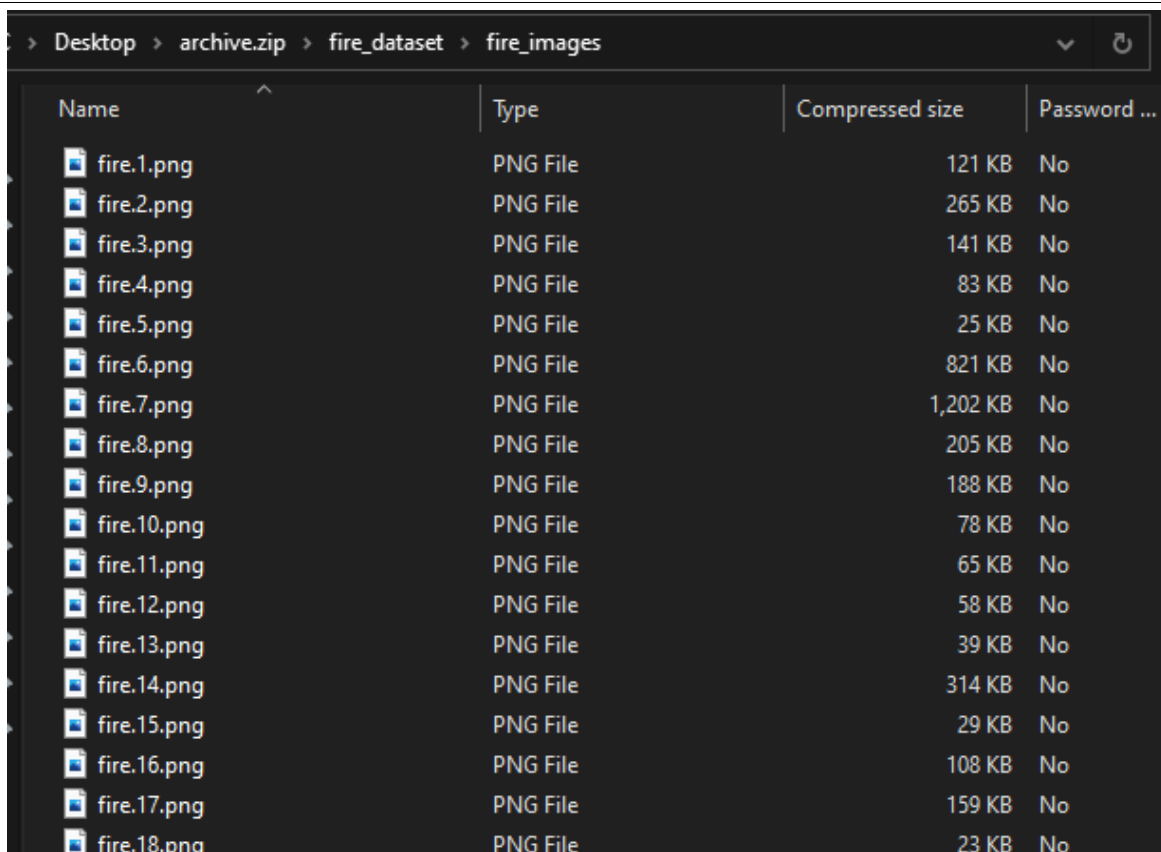
D.2 - Datasets

- Describe the raw and processed data.
- Provide an example (or examples) of the raw and processed data (if applicable).
- Provide access to any datasets used.
- Describe how the data's security (storage and maintenance) suits the needs of your project.

The data used to train this model was downloaded from Kaggle.com and google images, both of which are public sites and as such, face no data security issues. The data can be accessed in the data sets folder, with images being in the appropriately labeled folders (fire and not fire). The data did not require much processing other than uploading to Google Cloud Storage and subsequently adding a label of “fire” or “nonfire” manually.

As far as storage and maintenance, due to our images being cloud hosted, as long as we keep paying a hosting fee, Google will take care of all the maintenance for us, keeping things simple for the needs of our project

“Raw” Image Data



The screenshot shows a file explorer window with the path: Desktop > archive.zip > fire_dataset > fire_images. The window displays a list of 18 PNG files, each with its name, type, compressed size, and password status.

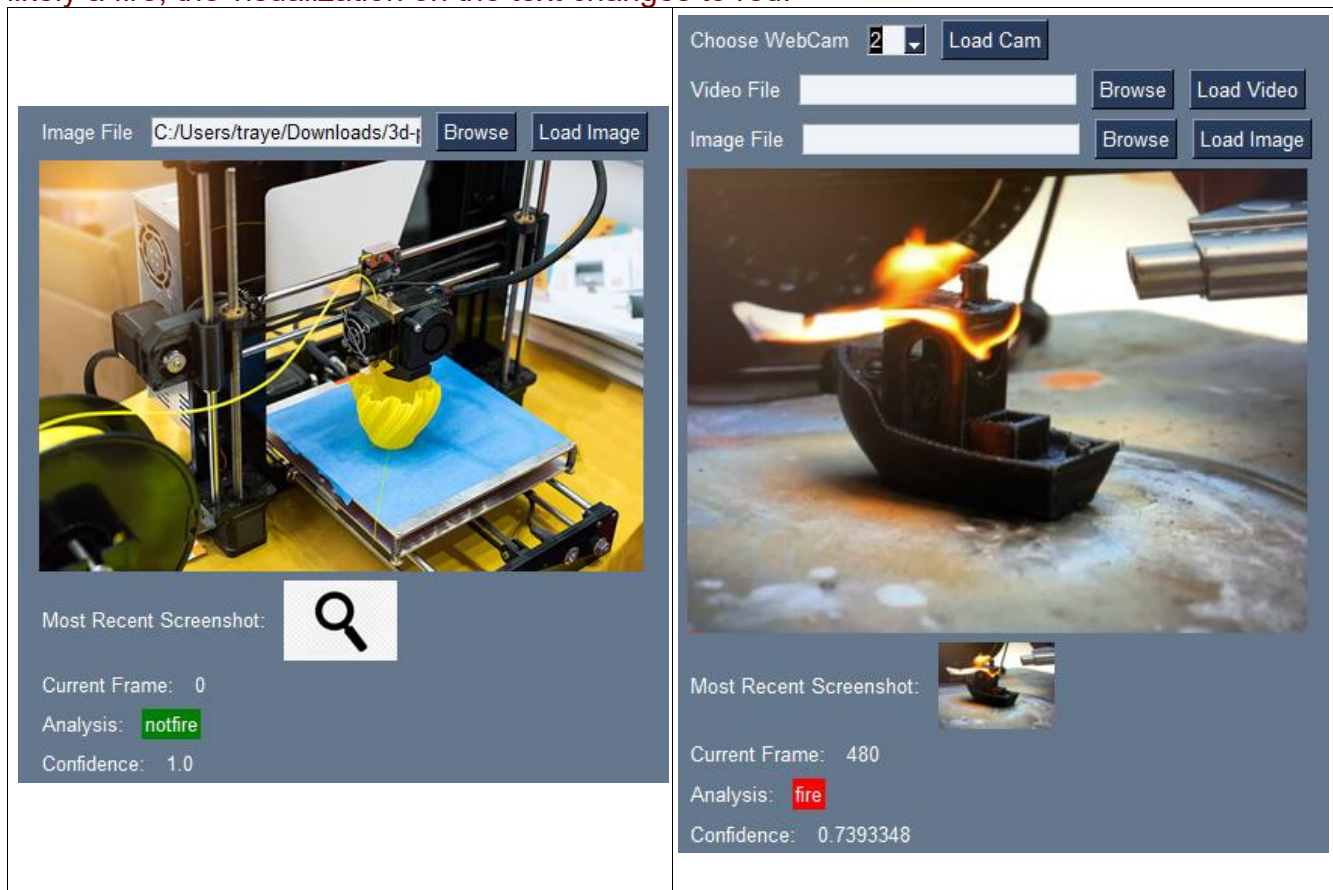
Name	Type	Compressed size	Password ...
fire.1.png	PNG File	121 KB	No
fire.2.png	PNG File	265 KB	No
fire.3.png	PNG File	141 KB	No
fire.4.png	PNG File	83 KB	No
fire.5.png	PNG File	25 KB	No
fire.6.png	PNG File	821 KB	No
fire.7.png	PNG File	1,202 KB	No
fire.8.png	PNG File	205 KB	No
fire.9.png	PNG File	188 KB	No
fire.10.png	PNG File	78 KB	No
fire.11.png	PNG File	65 KB	No
fire.12.png	PNG File	58 KB	No
fire.13.png	PNG File	39 KB	No
fire.14.png	PNG File	314 KB	No
fire.15.png	PNG File	29 KB	No
fire.16.png	PNG File	108 KB	No
fire.17.png	PNG File	159 KB	No
fire.18.png	PNG File	23 KB	No

D.3 - Data Product Code

- Review the functionality of the code used to perform the analysis and develop your application.
 - o Processing raw data (if not applicable, explain why).
 - o Descriptive methods(s) and visualizations.
 - o Non-descriptive method(s)
- For the non-descriptive portion, describe the following:
 - o The analytic methods
 - o Why these methods are appropriate for the project
 - o How they were developed, e.g., trained and tested.
- Where applicable, discuss how the data analysis supported the choosing and improving your descriptive and non-descriptive methods.
- Through direct submission or shared links, evaluators should be given access to all sources necessary for the development of your project.

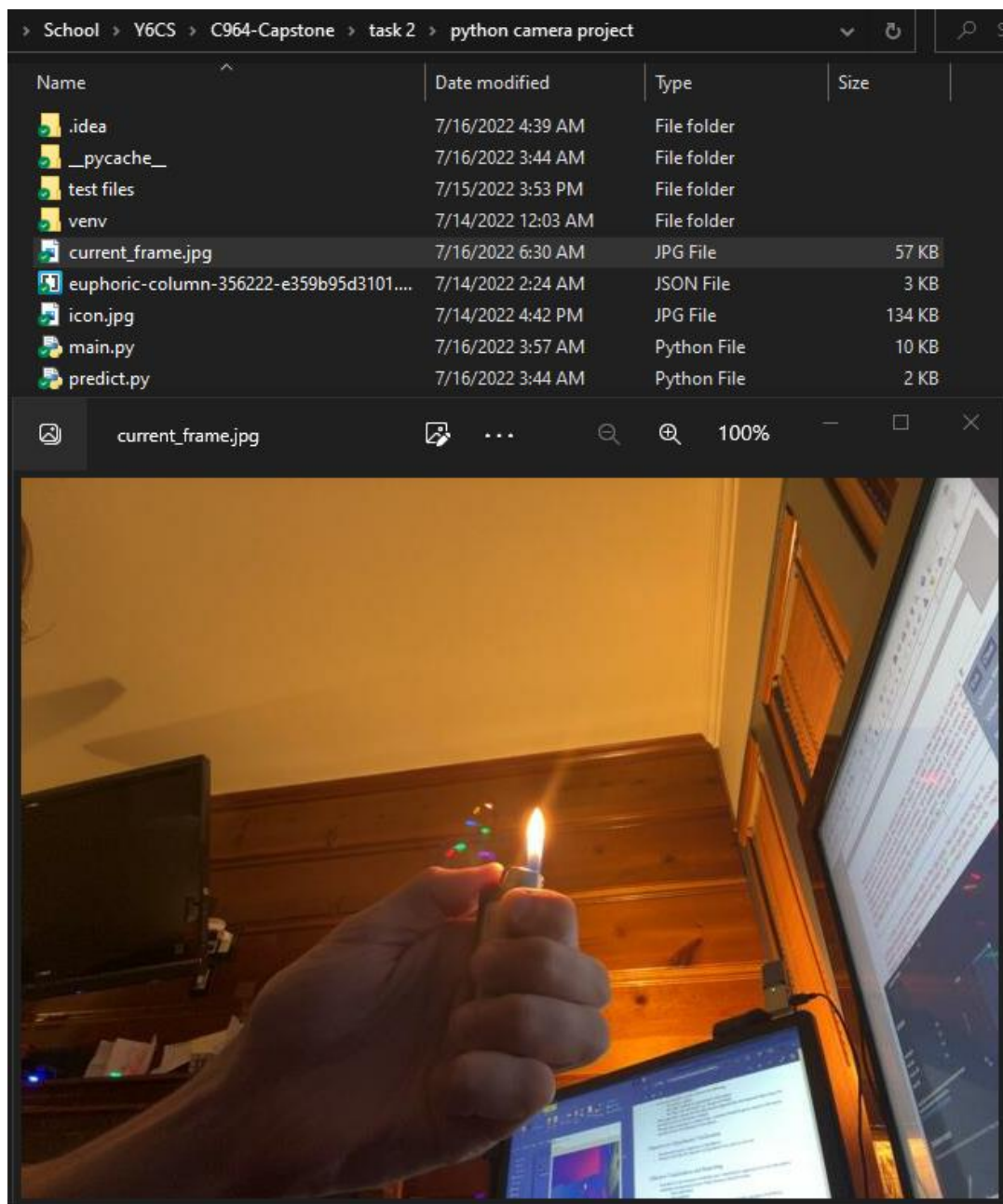
To simplify things, our application is run in two discrete pieces: an image processing and data parsing front-end, used to give the user a dashboard to interact with their files/streams and see the results, and the cloud-hosted back-end, which holds all of our training images and the model taught by them, ready to give predictive analysis when summoned.

Or python front end just had to get the necessary image at the proper time, display it (or loop through them if it's a video), send it to our google cloud predict() function, and parse the results. No problem if the predictive analysis engine comes back with nonfire, and the display will visualize the safety with green. But if the predictive engine determines there's likely a fire, the visualization on the text changes to red.



As for the back end, for this proof-of-concept, we used data that was relatively easy to wrangle. The titles and other attributes of the images didn't matter, as we applied their labels via dashboard software instead of through code. Thankfully we were able to locate a data set on Kaggle that fit our purposes nicely, saving us time on data processing. Our predictive analytic method-using model was developed by changing the dataset and modifying the confidence threshold until the results were favorable.

Together with the python front end, the app allows any user to determine the danger of fire present in any streamed or recorded video or picture, thanks to live video and predictive visualizations. If the user ever disagrees with the way the app is behaving (false positive or negative fire), then they can access that image in the Application folder as `current_frame.jpg`, to be uploaded to our cloud data set to strengthen our model if need be.



D.4 - Objective (or Hypothesis) Verification

- Describe the project's objective (or hypothesis).
- Discuss if and why the objective (or hypothesis) was met (or not met).

The problem is one of automated object recognition at scale and cost. The hypothesis, is that this problem can be solved better and cheaper with a machine learning-based application solution, rather than an expensive and litigious human. Furthermore, our company possesses the requisite skill and capital, to develop such an app, and that it will prove itself a worthy safety solution.

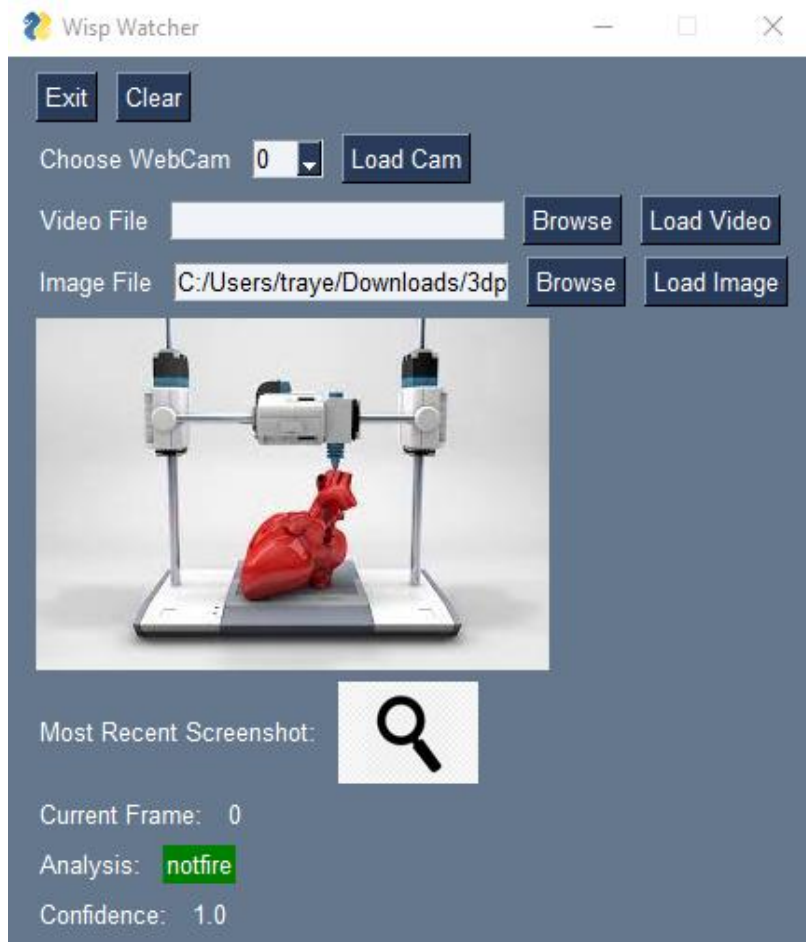
Post development and testing, in specific scenarios, the application has passed with over 99% image accuracy on the training set (at a cost of less than \$200 to train), leaving me convinced that machine learning is an effective, affordable solution. However, due to the high-risk nature of this application, extensive testing should be done prior to putting lives on the line. However I remain convinced that a machine learning solution is right for this problem.

D.5 - Effective Visualization and Reporting

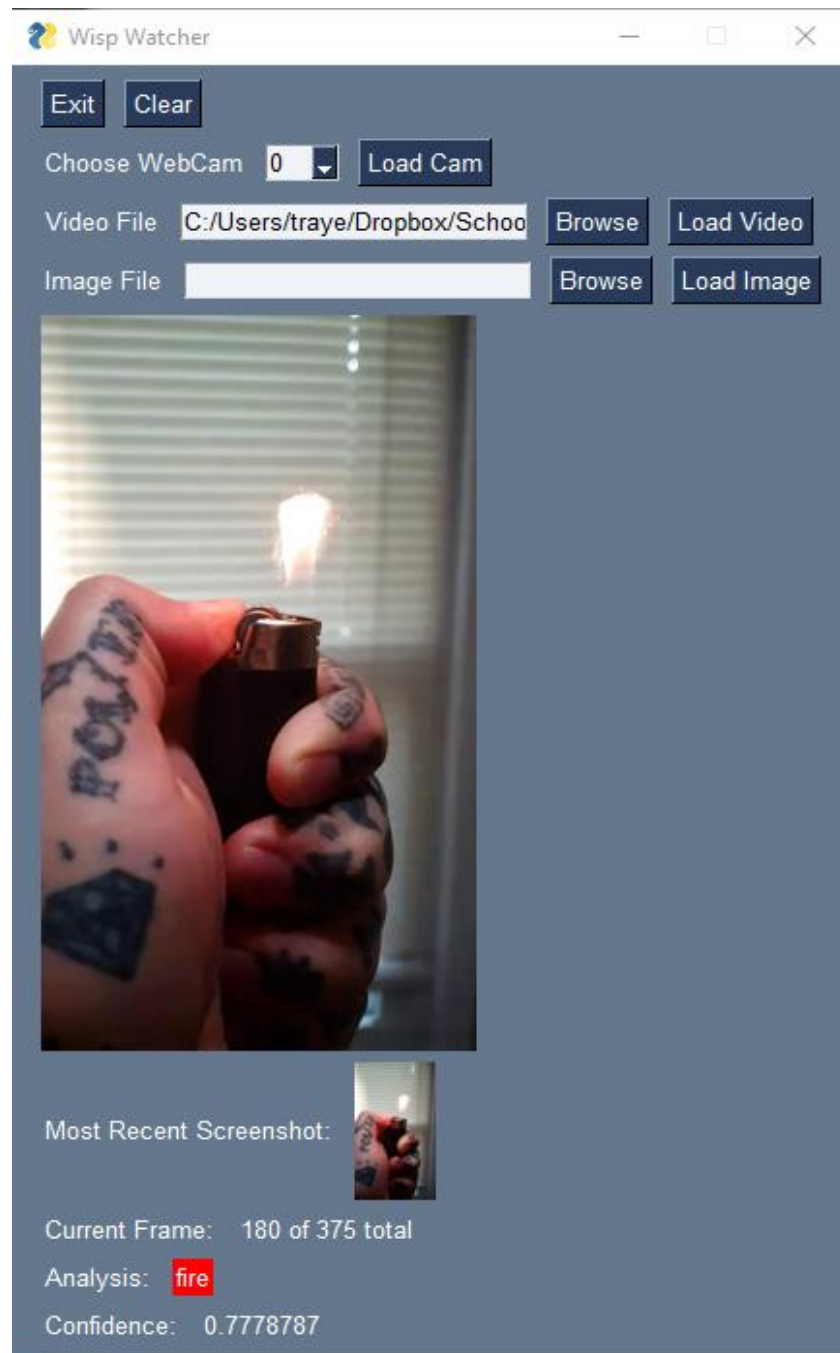
- Describe how the descriptive method(s) and visualizations supported your non-descriptive method(s) development process. Items discussed should include:
 - o Data exploration.
 - o Data analysis.
 - o Data summary.
 - o Analysis application of three visualizations (include examples of all three).

The primary function of our application is a kind of an image summarizer. Take an image, run it through a machine learning engine trained on hundreds of images of fire, and use that descriptive dataset to create a predictive result. To assist with showing that summary, the main screen consists of several visuals: A live feed of the current video source, a thumbnail of the most recently analyzed image, and the results of the analysis of said image. The results are colored to draw the attention to them. Our app can handle three kinds of inputs and as such, has three sets of behaviors and visuals. When an image is used, the image is loaded and sent for prediction immediately, very static. When a video is loaded, the frames are kept track of, and an image is only taken for analysis every 60 frames. With a video file instead of a video stream, the visual changes again, freezing up when a video loads an image of fire, instead of continuing to load more frame like the stream.

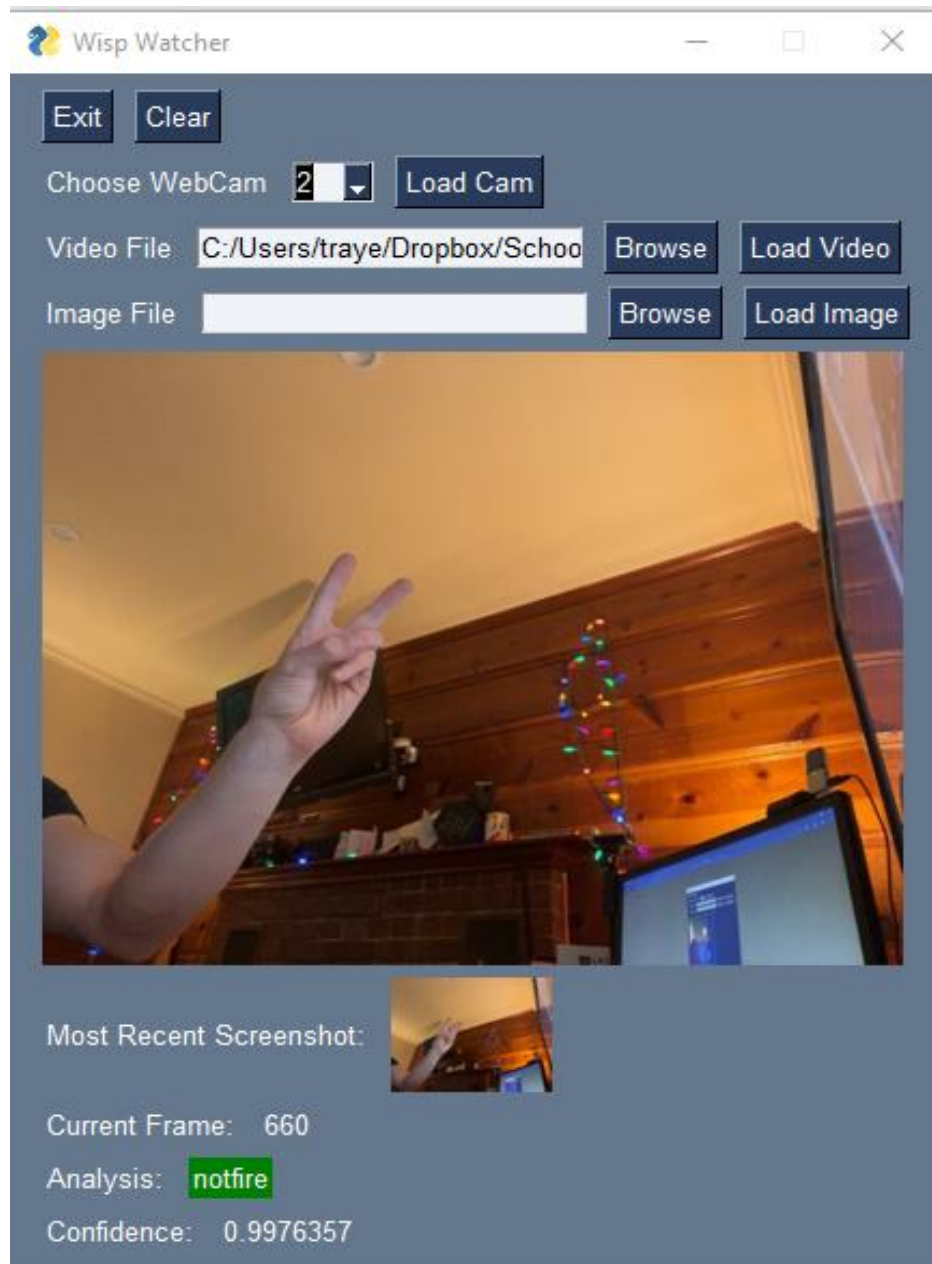
Image File Analysis:



Video File Analysis:

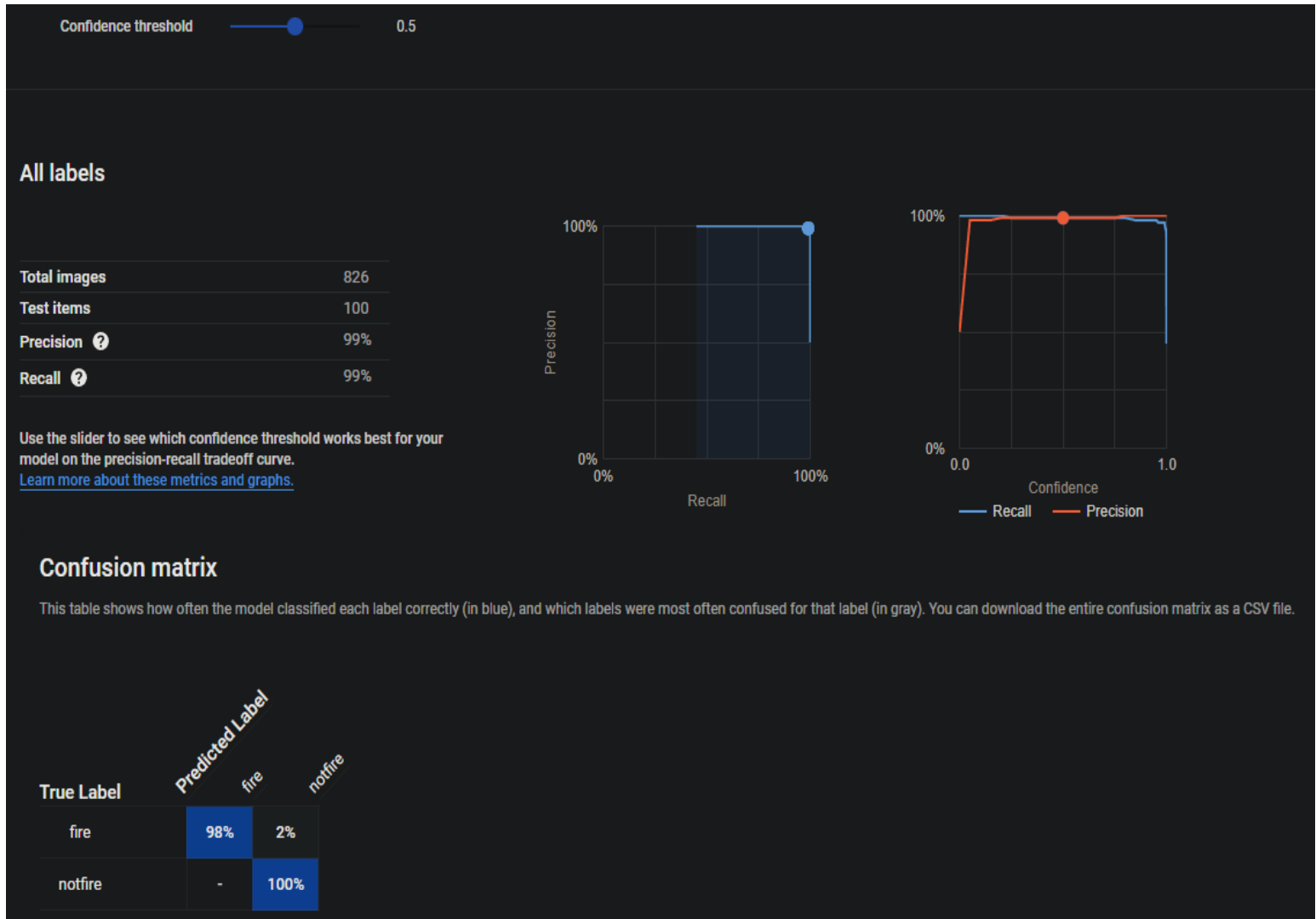


Web-Cam/Live-Stream Analysis



BACK - END

In order to view and manage data for the back-end, the user must log in using the instructions found in the user guide (**page 25**). Here the models are analyzed, and graphs are displayed showing the results of their precision tests. The labeled database can also be viewed.



D.6 - Accuracy Analysis

- Provide a description including objective metrics assessing the accuracy of your non-descriptive method.

o If not applicable, describe how future developments of the project could measure accuracy.

- Include an example demonstrating the non-descriptive method and discuss the accuracy.

For every image sent to our Google Cloud-hosted model, a few pieces of information come back from the query: a prediction on whether or not the image contains fire, and the confidence of that assessment between 0 and 1. To test our model, we trained it on ever more images. Our first model trained on 200 images had a 99% confidence rating, but mislabeled things constantly. We upped the training images to 500, but unfortunately, the increased images led to a lower avg precision of 74%, and a false positive rate of nearly 50%. Upon adding 700 images of fire and 250 images of non-fire led to the sweet spot, with an accuracy rating of 99% on all 900 images in our training-testing images.

After being deployed, accuracy of the app/model can be compared to the results reported by human technicians, to better compare safety per cost of both safety solutions.

D.7 - Application Testing

- Describe how the application was tested.
- Explain how the testing results were used to improve the application. If no modification was necessary, explain why.

As is evident by the code review and screenshots, the application has three main functions: image, video, and webcam/live-stream content analysis, with a back-end powered by a machine learning model. So during development testing was performed on both the front and back end until they performed as expected.

To Test the model, after training, I uploaded and tested an image that would trip up the last model. If it succeeded in predicting the results correctly (as judged by my human eyes), I would continue the process. I deemed it ready when I could no longer fool it into mislabeling images containing fire as nonfire.

To test the front end, I first made sure the application can communicate with google's API. Once a basic ui was set up, I gave the user the ability to upload an image, and verified that the values returned were the same as when I tested the image myself in the online dashboard. Subsequent time was spent expanding the GUI to handle images (.mp4 files), testing to ensure that fire is detected, that the proper frames are displayed, and that the video pauses when fire is shown. Finally, I added and debugged the web cam functionality, allowing the user to stream and send a prediction with any image their webcam could see. I held a lighter in the middle of the shot of my webcam and ensured it could detect it properly.

Further testing can be done of course but I deemed this adequate for a prototype.

D.8 - Application Files

- Provide a hierarchical list of files and libraries required to execute (or access) your application through a Windows 10 machine.
- Describe the organization of files in the submission.

Files for App are in:

python camera project : folder
main.py, predict.py

Image Files used to train and test Model are in:

fire_dataset : folder
fire_images : folder
non_fire_images : folder

Google Cloud Authentication is found in:

euphoric-column-356222-e359b95d3101.json

D.9 - User Guide

- Include an enumerated (steps 1, 2, 3, etc.) guide to execute and use your application.
 - o Include instructions for downloading and installing any necessary software or libraries.
 - o Your application will be considered “user-friendly” if the evaluator successfully executes and uses your application on a Windows 10 machine following your instructions.
- Though not required, an example demonstrating how a user will apply the product is helpful for evaluators.

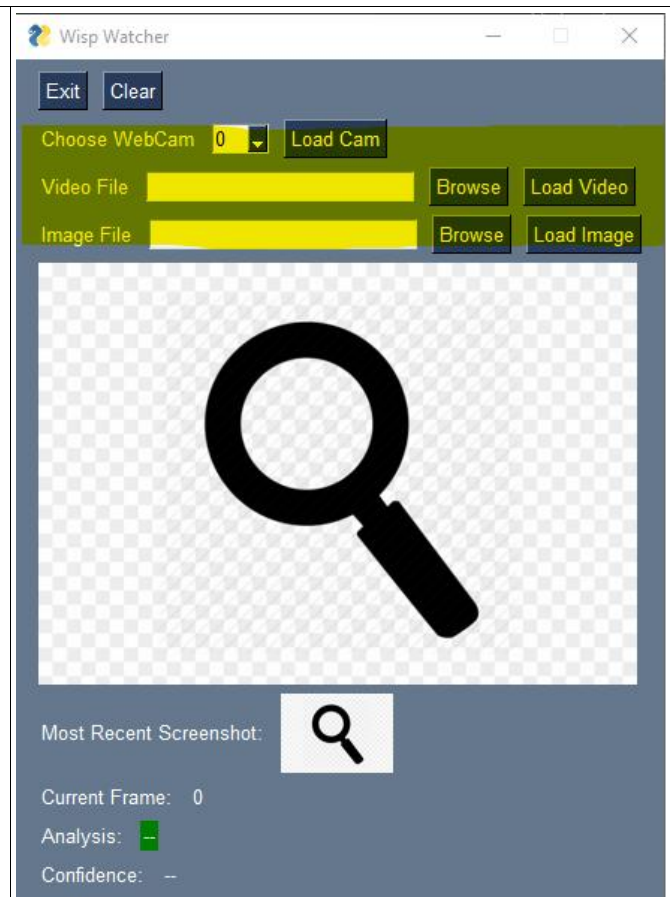
0 – Python Setup

- Download and Install PyCharm if you do not have it already
- Open the “Python Camera Project” folder as a project in PyCharm
- Install opencv if you don't have it, I recommend using 'pip install opencv-python'
- Ensure that the project interpreter is set to the python.exe file included in the venv folder in the project, alternatively use Python 3.9 (3.7 also seems to work in testing)
- Hit Play/Run when ready

1 – Option Selection

Upon first launching the application with PyCharm, the user is greeted with three possible inputs, accessible via different buttons:

- Choose WebCam | number drop-down → 4
- Video File | browse file → 3
- Image File | browse file → 2

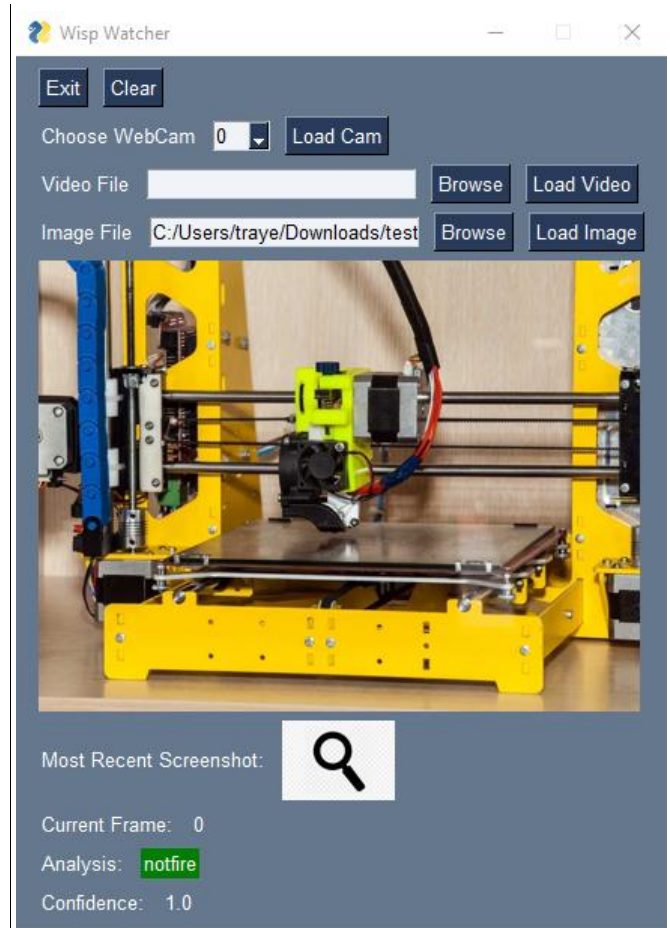


2 – Image Processing

After hitting browse, the user is shown their file explorer and encouraged to select a JPEG for analysis. Click Load to start the process, or Clear to start over → 1

Once loaded, the image is displayed while being uploaded to the model. Below the image there are a few values including a prediction back from the model on what the image represents (Analysis), and a number representing the Confidence in that prediction. If a predictive analysis indicates that the image contains fire, a visual is shown by way of a red background applied to the label

When done, hit Clear to load new data or Exit/X to close the app



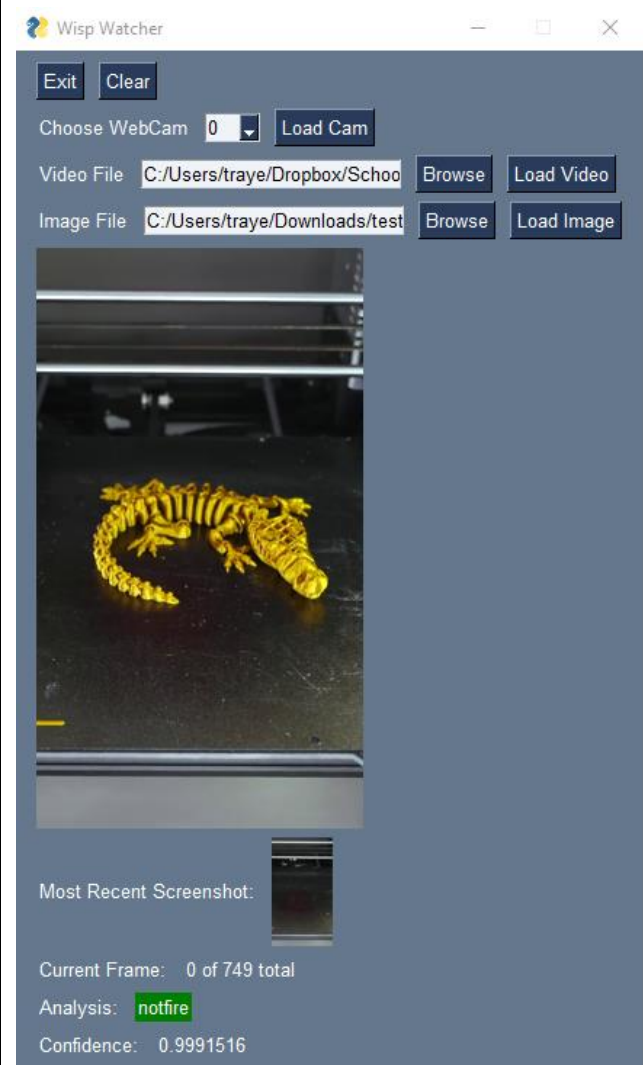
2 – Video Processing

After hitting browse, the user is shown their file explorer and encouraged to select a .mp4 for analysis. Click Load to start the process, or Clear to start over → 1

Once loaded, the video is played frame by frame while every 60 frames, an image is saved and uploaded to the model. Below the video, a thumbnail of the most recently analyzed image is displayed, and there are a few values including a prediction back from the model on what the image represents (Analysis), and a number representing the Confidence in that prediction

If a predictive analysis indicates that the image contains fire, a visual is shown by way of a red background applied to the label and the video playback is halted

When done, hit Clear to load new data or Exit/X to close the app



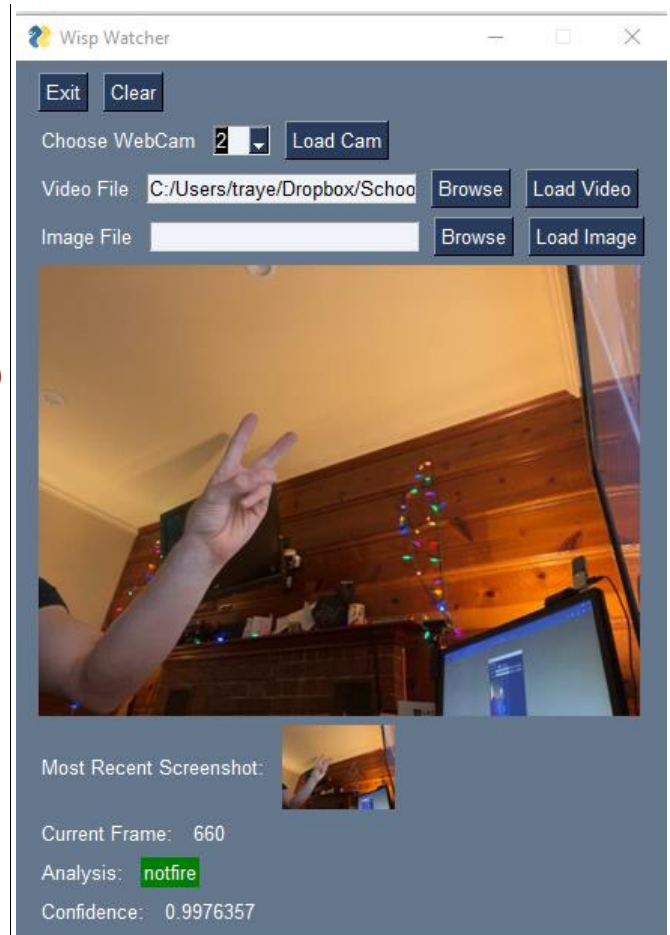
3 – Web Cam/Live Stream Processing

If a webcam is installed, it will usually be under “0”, however mileage may vary. If the user lacks a webcam but owns a smartphone, they can use the DroidCam iOS/Android app along with the accompanying windows app. This lets the user use their smartphone camera as a webcam (on port 1 and 2 for me)

Once loaded, the stream is played frame by frame while every 60 frames, an image is saved and uploaded to the model. Below the stream, a thumbnail of the most recently analyzed image is displayed, and there are a few values including a prediction back from the model on what the image represents (Analysis), and a number representing the Confidence in that prediction

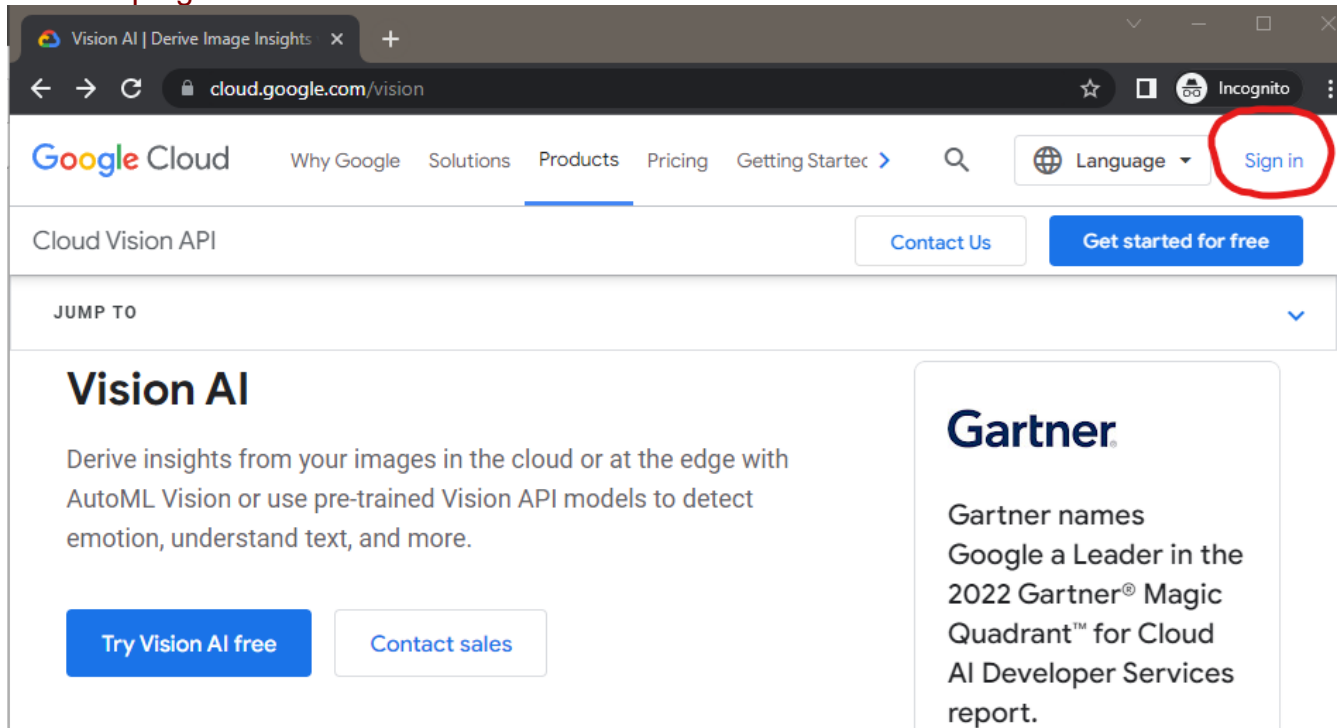
If a predictive analysis indicates that the image contains fire, a visual is shown by way of a red background applied to the label

When done, hit Clear to load new data or Exit/X to close the app



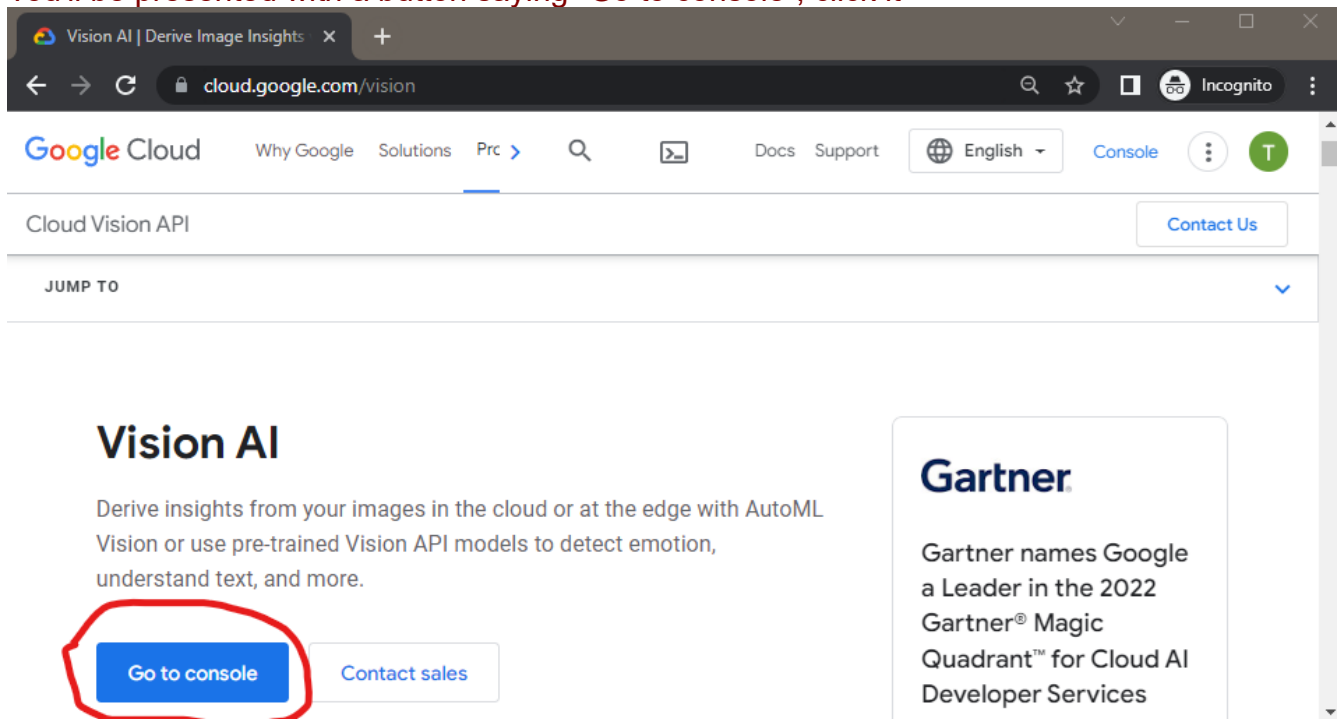
Back – End Access

In order to view the database and model as well as training and data visualizations, open a web browser of your choice and navigate to <https://cloud.google.com/vision>, then click Sign In on the top right

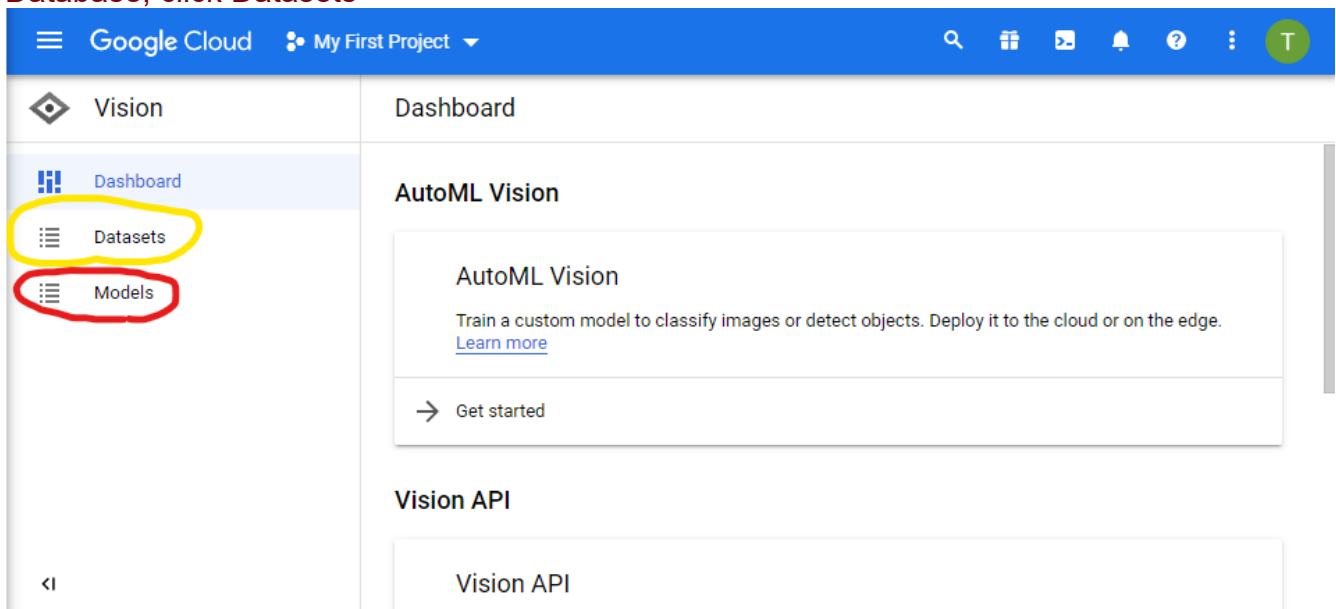


and log in with [REDACTED]. This will bring up a wgu sign in page. Sign in with [REDACTED], and password : [REDACTED]

You'll be presented with a button saying "Go to console", click it

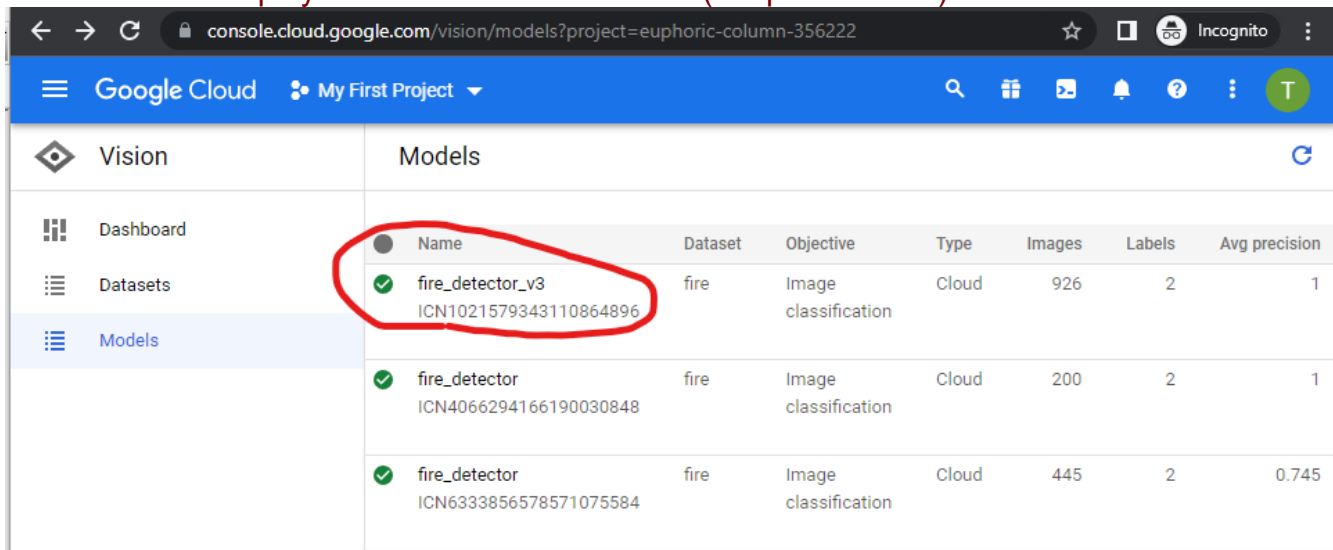


You should now see the console. To view the model, click Models on the left, to view the Database, click Datasets

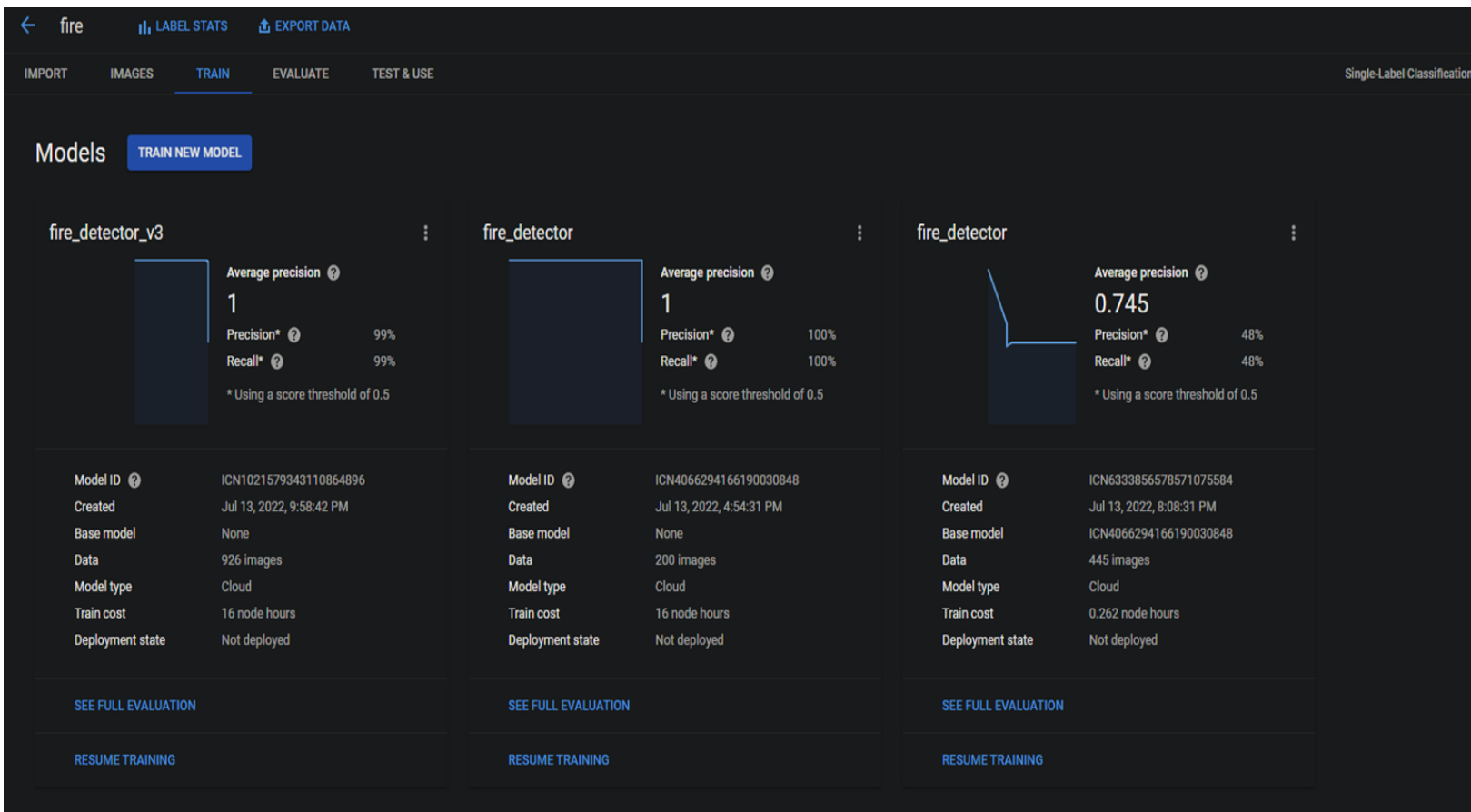


If viewing models:

There will be three models visible. Two were used during testing phases, with `fire_detector_v3` being the one ultimately best trained and currently deployed. Click it to view details such as those seen on **page 22**, as well as the Confidence Threshold and data labels. Models can be deployed and removed from here (but please don't).



Vision		Models						
Dashboard		Name	Dataset	Objective	Type	Images	Labels	Avg precision
Datasets		✓ fire_detector_v3 ICN1021579343110864896	fire	Image classification	Cloud	926	2	1
Models		✓ fire_detector ICN4066294166190030848	fire	Image classification	Cloud	200	2	1
		✓ fire_detector ICN6333856578571075584	fire	Image classification	Cloud	445	2	0.745



fire

LABEL STATS

EXPORT DATA

IMPORT IMAGES TRAIN EVALUATE TEST & USE

Single-Label Classification

Models TRAIN NEW MODEL

fire_detector_v3

Average precision ?

1

Precision* ? 99%

Recall* ? 99%

* Using a score threshold of 0.5

Model ID ?

ICN1021579343110864896

Created

Jul 13, 2022, 9:58:42 PM

Base model

None

Data

926 images

Model type

Cloud

Train cost

16 node hours

Deployment state

Not deployed

SEE FULL EVALUATION

RESUME TRAINING

fire_detector

Average precision ?

1

Precision* ? 100%

Recall* ? 100%

* Using a score threshold of 0.5

Model ID ?

ICN4066294166190030848

Created

Jul 13, 2022, 4:54:31 PM

Base model

None

Data

200 images

Model type

Cloud

Train cost

16 node hours

Deployment state

Not deployed

SEE FULL EVALUATION

RESUME TRAINING

fire_detector

Average precision ?

0.745

Precision* ? 48%

Recall* ? 48%

* Using a score threshold of 0.5

Model ID ?

ICN6333856578571075584

Created

Jul 13, 2022, 8:08:31 PM

Base model

ICN4066294166190030848

Data

445 images

Model type

Cloud

Train cost

0.262 node hours

Deployment state

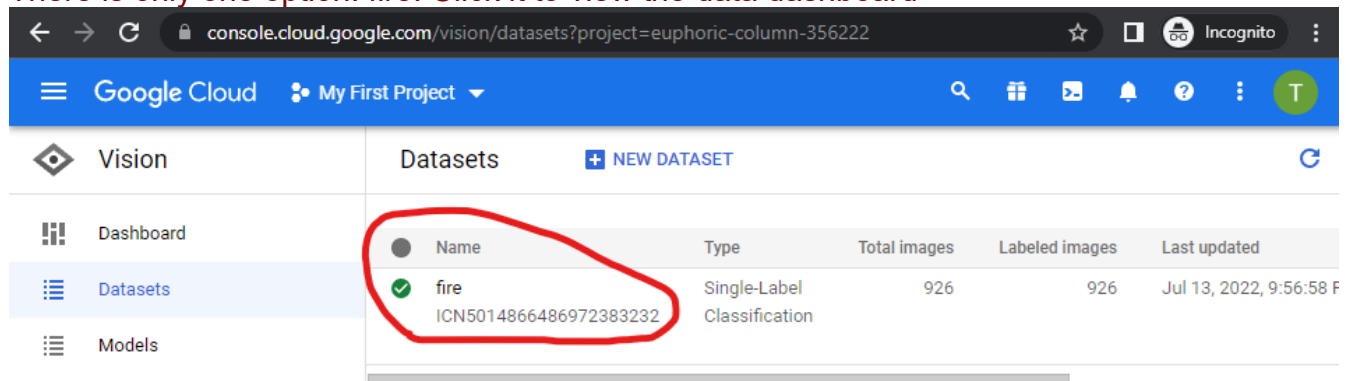
Not deployed

SEE FULL EVALUATION

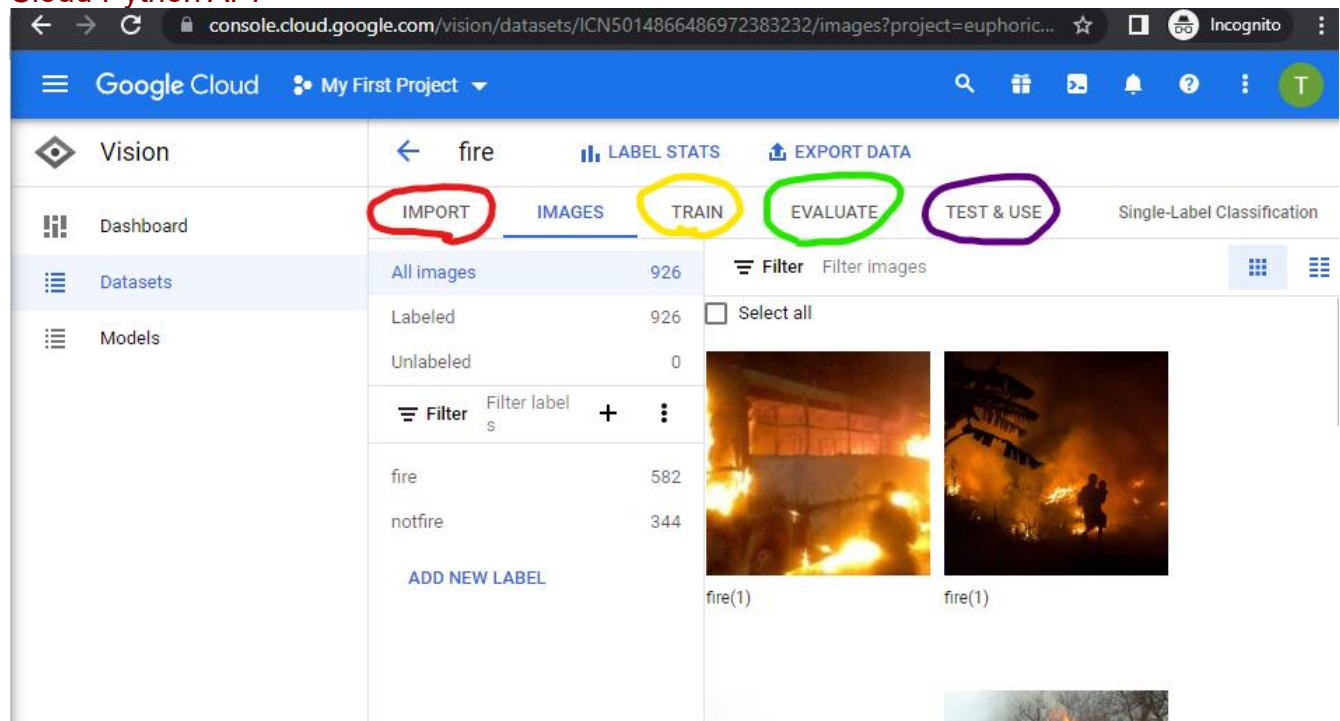
RESUME TRAINING

If viewing Datasets:

There is only one option: fire. Click it to view the data dashboard



Here you can see the labeled data set used to train the models, such as the example shown on **page 15**. Clicking the **Import** tab allows a user to add new images to the dataset, the **Train** tab allows the user to train a model for more time (and more money), and the **Evaluate** tab leads to the model trained and in use currently. **Test and Use** lets a user upload individual images for evaluation via the model, and gives instructions on how to connect via the Google Cloud Python API



D.10 - Summation of Learning Experience

- Describe how your previous experience (academic or professional) readied you for this project.
- Describe any additional learning or resources needed to complete this project.
- Describe how this experience contributed to your concept of lifelong learning.

This project was rightly a culmination of so many of the concepts I've learned over the past few semesters at WGU and beyond. I've been programming and coding since high school, but Software I at WGU was the first time I made an application with a GUI (JavaFX). That time made it so that doing the same thing in Python now wasn't that big a deal. The practice I got writing Python in Data Structures II was instrumental in my development for this capstone. All of the course material with this class was useful in determining what kind of machine learning I needed to apply to my problem, and how I might go about finding and training that kind of model. In order to use Google's API I drew on my own experience writing web-scraping apps for personal projects, and I did a lot of external research into Python UI libraries and kaggle datasets. Ultimately I was more daunted by the document.

Thankfully, Intro to Artificial Intelligence and even the Task 1 of this class helped give me more manageable project documents to get used to sinking my teeth into before dealing with this one. Completing this project has given me a confidence in my abilities to dissect and solve any software-based challenge that stands before me, and I'll take the lessons in problem solving and machine learning with me as I become a full fledged developer.

Part E: Sources

Saied, Ahmed (2020). FIRE Dataset. Retrieved July 10, 2022, from <https://www.kaggle.com/datasets/phylake1337/fire-dataset>.

Google. (n.d.). Test and Use: Predict.py. Google cloud platform. Retrieved July 16, 2022, from <https://console.cloud.google.com/>

3D Printers. (n.d.). Retrieved July 10, 2022, from <https://www.google.com/search?q=3d+printers>.