Trademark Analysis & Identification Tool (TRAIT)

Team 11 Sprint 15 Lane Keck, Logan Taggart, Caleb Stewart

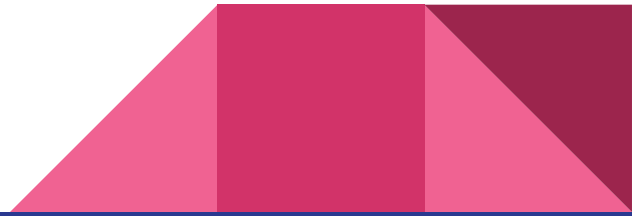
Lazy Loading

We have finished converting library imports from loading into memory at runtime to instead being loaded into memory as is called/needed by the functions

By lazy loading we were able to bring our initial load time from around 45 seconds to under 5 seconds

Majority of load time is deferred to first time a user goes to process image or video

This is important because when we launch our electron desktop application we want both the frontend and backend flask server to launch in unison

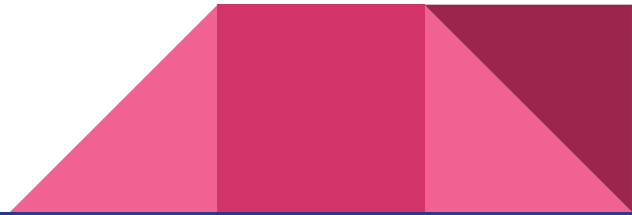


PyInstaller Executable

To build a single executable that users will be able to launch without the need to install any dependencies we are using the PyInstaller library

This will bundle all of backend Python files and libraries into a single file that can be launched to start the backend

Using the –windowed option also allows for us to run the executable without any terminal windows displaying

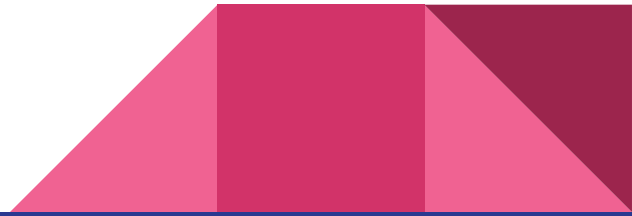


Github Actions

We have wrote a couple automations using Github Actions to accomplish two things:

- The first one is in our backend repository and is used to package our backend together with all required dependencies into a single executable
- The other one is in our frontend repository and retrieves the backend executable to package/build into our complete Electron application that users will be able to install and run to launch both frontend and backend from a single file

By using GitHub actions we are able to build versions for both Mac and for Windows



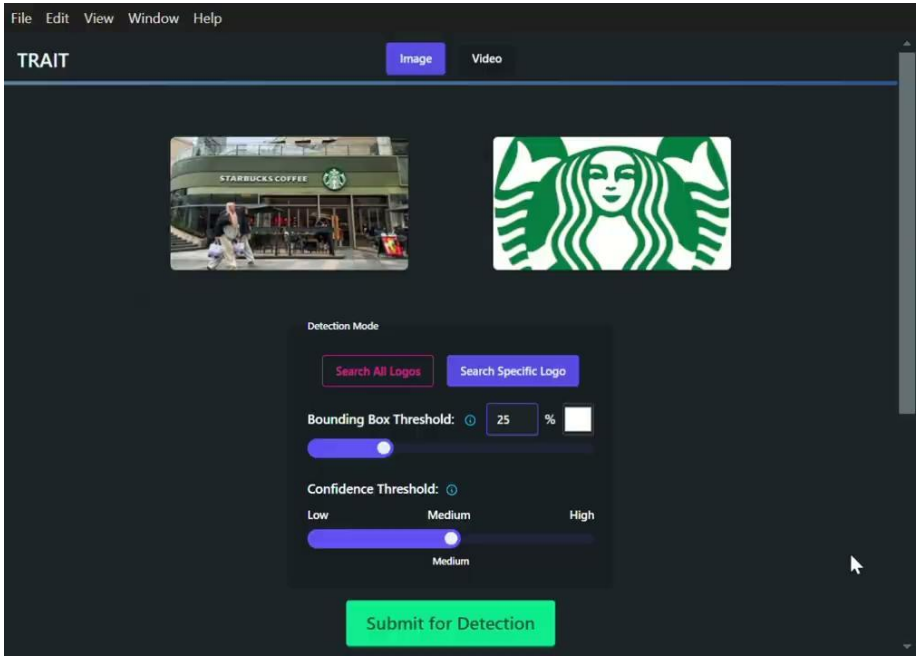
Cancel Process Button

When image or video is submitted the submit button switches to become a cancel button

Clicking this button will terminate the current process without shutting down the backend server

This is done by triggering a flag in the backend, which causes the long-running processing functions to exit early.

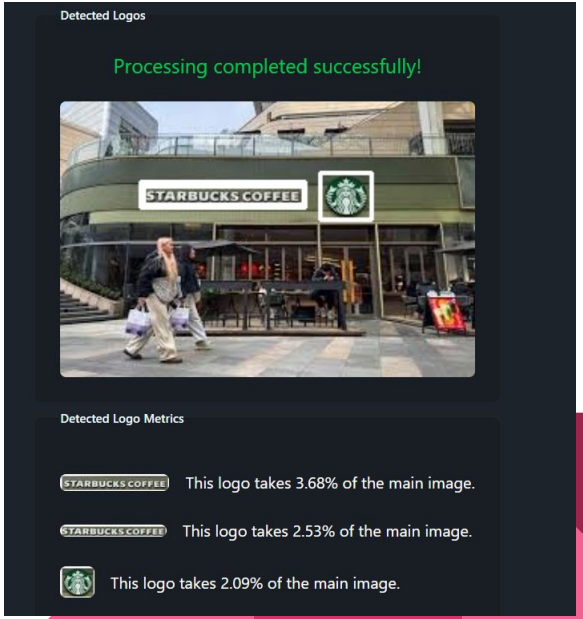
Cancel Button Demo



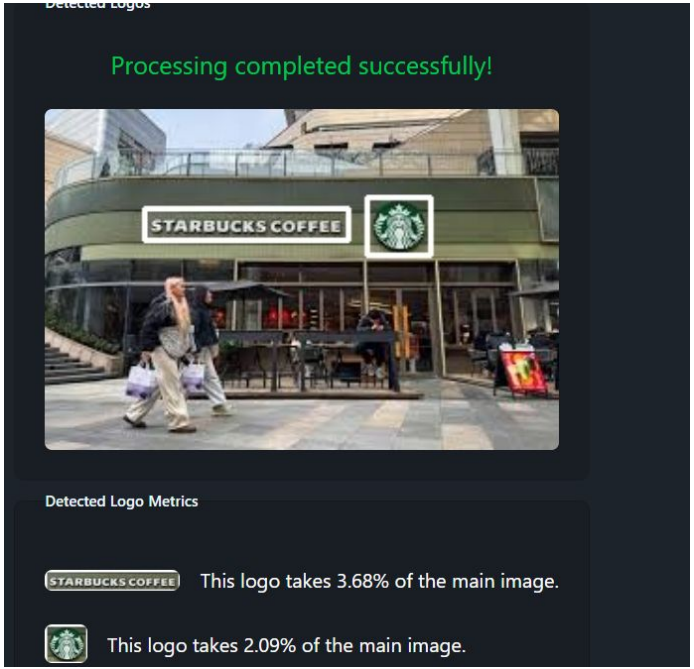
Overlapping Boundary Boxes - Lane

Our model sometimes detects the same logo twice, which creates this overlapping bounding box. We did some research and found that there is a parameter when we call our model that can help this.

Examples on next slide as well



Overlapping Boundary Boxes - Lane




```python
if isinstance(image, str):  # File path
    img = cv2.imread(image)
else:  # Assume it's already an ndarray
    img = image

if img is None:
    print("Error: Could not load image.")
    return [], [], None
print("Boundary box threshold:", bounding_box_threshold)
results = model(img,conf=bounding_box_threshold,iou=0.3)
```

iou stands for “intersection over union” , in this example, if logos overlap more than 30 percent, show the one with the highest confidence

What’s Left?

Fixing a few bugs that we’ve noticed

Format resulting metrics in a better way

Minor cosmetic changes

Use backend executable in Electron app to bundle/build into a single file that users can run to use whole application

Creating clear and detailed project documentation