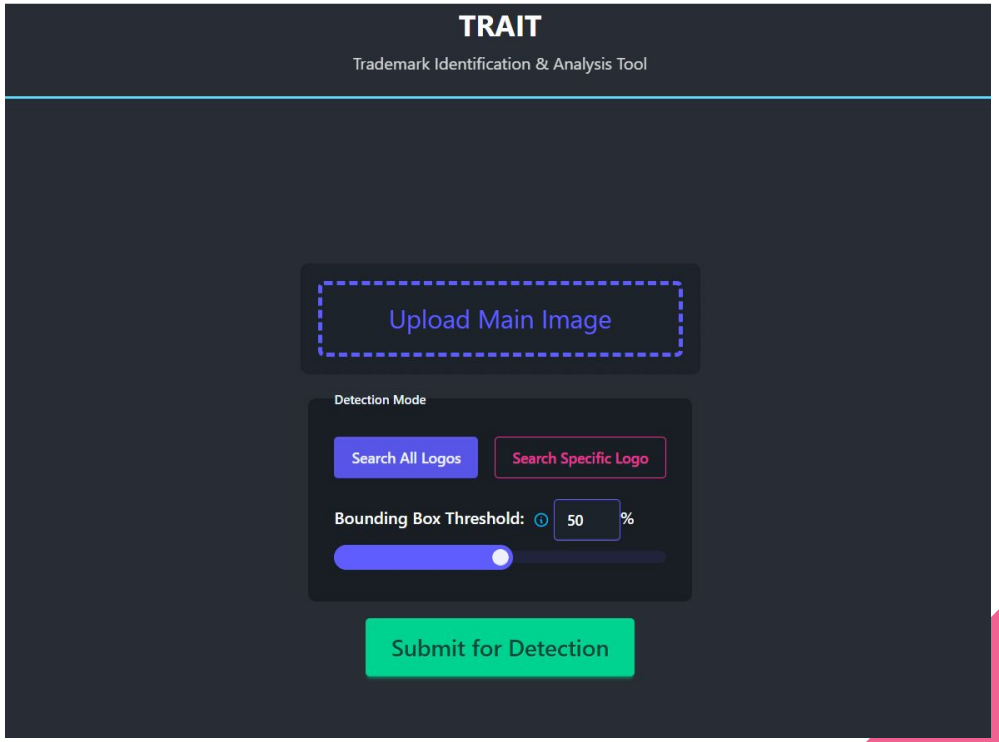
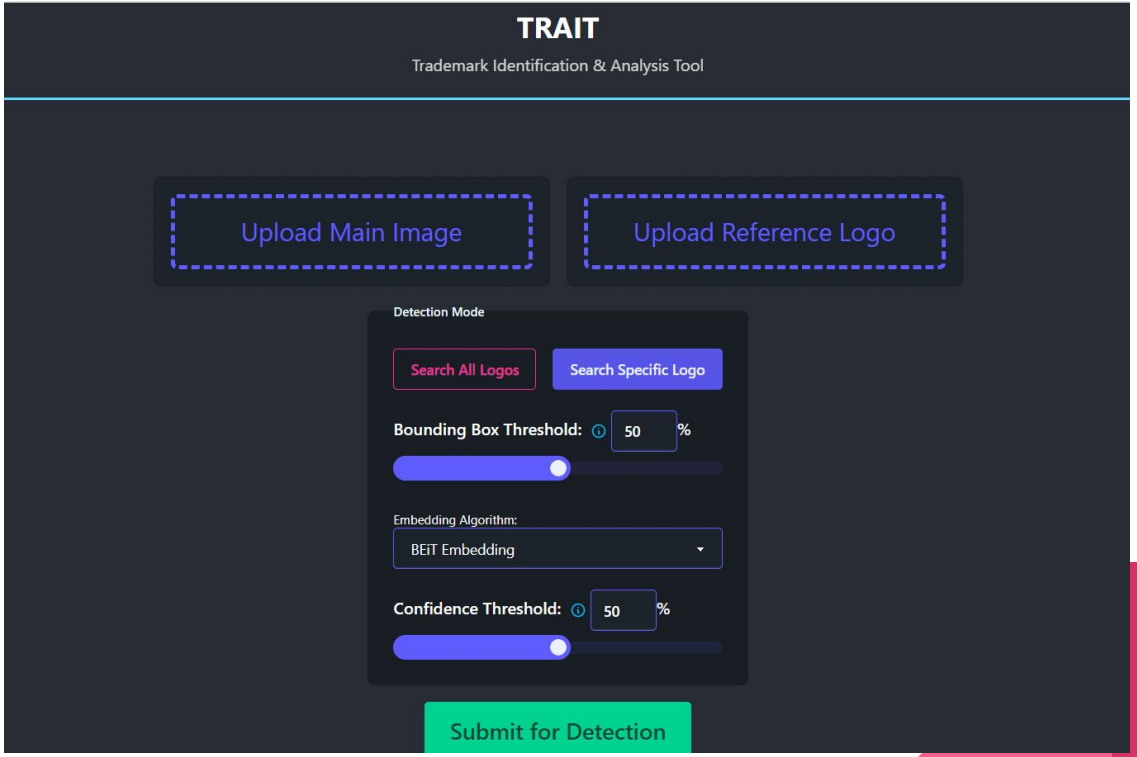
Trademark Analysis & Identification Tool (TRAIT)

Team 11 Sprint 5 Lane Keck, Logan Taggart, Caleb Stewart

Updated Front End!



Updated Front End!



Embedding Algorithms

Current Embedding Algorithms: BEiT, CLIP, ResNet

Our current embedding algorithms with cosine similarity is not working as well as we expected. It gives us low similarity scored when we compare images and makes it hard for us to distinguish between logos. → False Positives

We are looking for a clear cut difference between similarity scores which is proving hard to find.

Proposed Solution: Improving Similarity Detection

To enhance the accuracy of logo matching, we propose a multi-step approach that refines similarity evaluation across multiple embedding algorithms:

1. Compute Cosine Similarity:
    - Compare all detected logos in the main image with those in the reference image. (We already have this)
    - If the Cosine Similarity score passes a certain threshold, then the detected logo gets +1 point added to it

Proposed Solution: Improving Similarity Detection

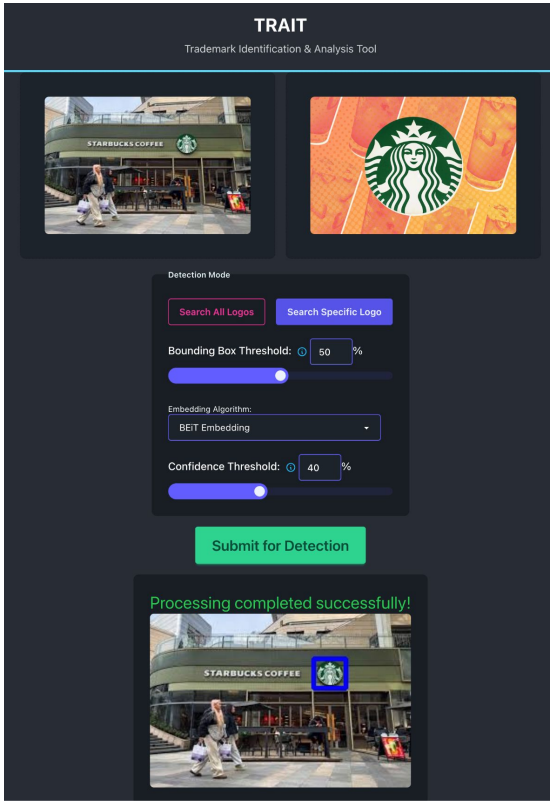
2. Compute Euclidean Distance:
    - Compare all detected logos in the main image with those in the reference image. (Same thing as the Cosine Similarity, but with Euclidean Distance)
    - If the Euclidean Distance score passes a certain threshold, then the detected logo gets +1 point added to it

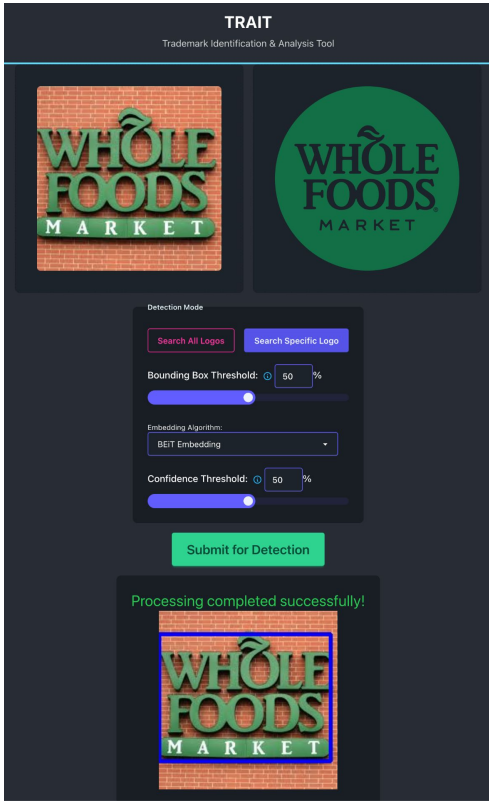
Proposed Solution: Improving Similarity Detection

3. Multi-Embedding Voting:
    - Repeat the process for all three embedding algorithms (BEiT, CLIP, ResNet).
    - A logo is classified as a match if it gets at least 4 points from all off the embedding algorithms (4/6 cosine and euclidean thresholds match)

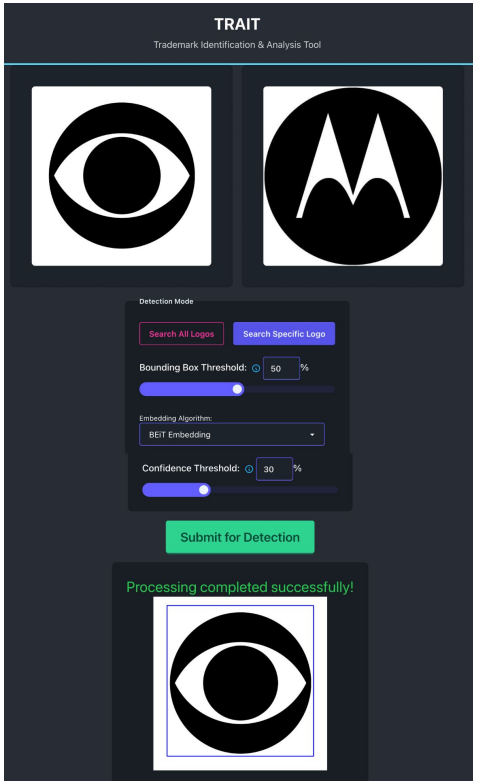
This takes a majority voting approach using all 3 embedding algorithms. This ensures that we don’t rely on one embedding algorithm for a similarity result. This should result in less false positives.

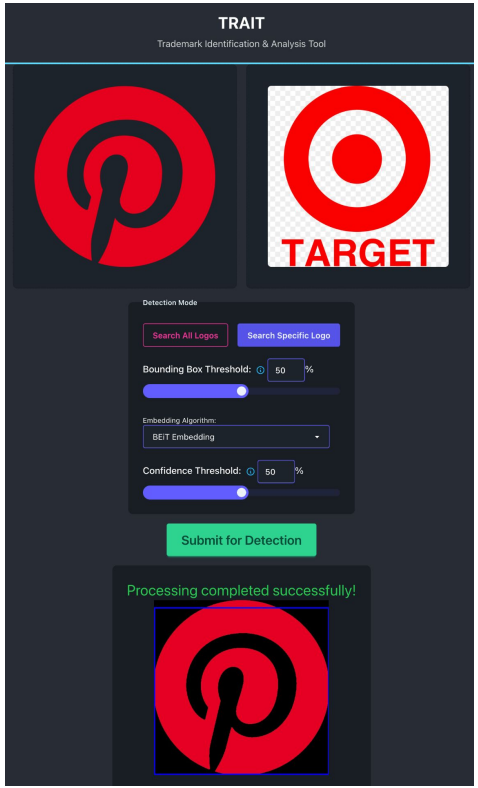
True Positive Examples





False Positive Examples





Changing Hosting Services

We chose render to host our backend service. As we discussed last week, it does not function properly and we will need to find a different way to host the backend.

A solution that we have found is pivoting from our application being a web app to instead having it be a desktop app.

To accomplish this we are going to wrap our current React app with Electron.js to easily be able to make it run locally on the user's machine.

We are picturing this as a single executable that will be able to be launched by the user to be able to start the front and back end of the app.

Challenges

- Dealing with false positives from high similarity scores
- Figuring out approach to minimize these
- Working with Electron.js
- Front end Development