

Trademark Identification & Analysis Tool

Team 11 Sprint 3 Lane Keck, Logan Taggart, Caleb Stewart

Our model is fully trained!

Recap: We are training our YOLO model on the LogoDet-3k dataset found on kaggle

Using Kaggle's servers we were able to finish training the YOLO model using 50 epochs, meaning the model passed through the entire training set 50 times.

Each epoch took about 40 minutes to complete, so $40 \text{ minutes} * 50 \text{ epochs} = \sim 33 \text{ hours}$ of training time.



Results of our YOLO model

Precision (81.03%):


Measures how many of the predicted logos are actually correct.

Interpretation: 81.03% of the logos the model detected are correct. The model does a good job avoiding false positives.

Recall (76.73%):

Measures how many of the actual logos in the dataset the model successfully detected.

Interpretation: The model successfully detected 76.73% of the logos. However, it missed about 23.27% of the logos.



Examples



Precision Example

Picture of a Starbucks cup with the lights also detected



Recall Example

Picture of 4 Starbucks cups having 3 of their logos detected



Python Flask Backend

Now that we have the model trained and working, we have started the process of setting up the backend to not just output the image back to a file in the current directory, but instead return it back to the front end via Flask.

We still need to create an outline of how our final API will be designed, but we do have it working with basic HTTP requests.

```
from flask import Flask, request, send_file, jsonify
from flask_cors import CORS
import cv2
import numpy as np
import io
from ultralytics import YOLO

app = Flask(__name__)
CORS(app)

model = YOLO("best.pt")

@app.route("/upload", methods=["POST"])
def upload_file():
    if "file" not in request.files:
        return jsonify({"error": "No file provided"}), 400

    file = request.files["file"]
    file_bytes = np.frombuffer(file.read(), np.uint8)
    img = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)

    results = model(img)
    confidence_threshold = 0.25

    for box in results[0].boxes:
        conf = box.conf[0].item()
        if conf > confidence_threshold:
            xxyy = box.xyxy[0].tolist()
            x1, y1, x2, y2 = map(int, xxyy)
            color = (255, 255, 255)
            thickness = 2
            cv2.rectangle(img, (x1, y1), (x2, y2), color, thickness)

    _, img_encoded = cv2.imencode(".jpg", img)
    return send_file(io.BytesIO(img_encoded.tobytes()), mimetype="image/jpeg")

if __name__ == "__main__":
    app.run()
```

Hosting

The easiest option for us to host our React front end is Vercel. It provides automatic deployment from GitHub and is free.

To host the backend Python Flask application we will most likely use Render which is also free and provides automatic deployment from GitHub.

This means we will have two GitHub repositories.



Challenges

- Uploading videos
- Tracking specific logos and brands via image upload
- False positives
- False negatives



Timeline Plan

By the end of winter quarter, our goal is to have a functional solution capable of analyzing still images for brand visibility. Specifically, we aim to develop a system that can accurately detect and track all logos present in a given image, while also implementing a method to identify a specific logo within an image based on an uploaded reference image.

This will lay the foundation for extending our solution to video analysis, allowing us to refine the system's ability to measure brand exposure.

