Assignment 5 Report: Model Refinement and Testing

1. Introduction

This report details the refinements and improvements made to our cup detection system between the two testing phases of Project Assignment 5. Building upon our previous work (documented in Assignment 4), we focused on enhancing model performance through dataset refinement, hyperparameter optimization, and comprehensive evaluation using UMAP and t-SNE visualizations.

2. Dataset Refinement

2.1 Data Collection and Augmentation

- Generated Images: Added 222 synthetically generated images to expand dataset diversity
- Manual Collection: Captured 65 additional real-world images covering challenging cases:

    - **Large cups**

    - **Images with multiple objects**

    - **Non-red Tim Hortons cups**

    - **Individual lids and sleeves**

- **Cups with partial sleeve coverage**

- **Background Diversity:** Collected images with various backgrounds to improve generalization

- **Additional Cups:** Plastic Tims plastic cup and coca cola metal cans for covering previous errors and edge cases.

2.2 Data Cleaning

- Removed empty annotation files

```
# prompt: show the file with 0 yolo object count

import os

def count_yolo_objects(folder_path):
  """Counts the number of YOLO objects in text files within a folder.

  Args:
    folder_path: The path to the folder containing the text files.

  Returns:
    A dictionary where keys are filenames and values are the object counts.
    Returns an empty dictionary if the folder doesn't exist or no text files are found.
  """
  object_counts = {}
  if not os.path.exists(folder_path):
    print(f"Error: Folder '{folder_path}' not found.")
    return object_counts

  for filename in os.listdir(folder_path):
    if filename.endswith(".txt"):
      filepath = os.path.join(folder_path, filename)
      try:
        with open(fil
          lines = f.r
          object_coun
      except Exceptio
        print(f"Error
  return object_count

# Example usage
folder_path = '/conte                                                    ual folder path
yolo_object_counts = count_yolo_objects(folder_path)

# Find and print files with 0 objects
zero_object_files = [filename for filename, count in yolo_object_counts.items() if count == 0]

if zero_object_files:
    print("Files with 0 YOLO objects:")
    for filename in zero_object_files:
        print(filename)
else:
    print("No files found with 0 YOLO objects.")
```

```
(function) count_yolo_objects: (folder_path: Any) -> dict

Counts the number of YOLO objects in text files within a folder.
Args:
  folder_path: The path to the folder containing the text files.
Returns:
  A dictionary where keys are filenames and values are the object counts.
  Returns an empty dictionary if the folder doesn't exist or no text files are found.
```

```
No files found with 0 YOLO objects.
```

- Balanced class distribution between normal cups and Tim Hortons cups with

- Added a total of 379 YOLO objects
- Number of 'Tims' objects (label 1): 276
- Number of 'Cup' objects (label 0): 103

- 222 images and data annotations files newly added covering all the identified cases

- Version 5 of the data now contains – • Total YOLO objects: 2968
    - Number of 'Tims' objects (label 1): 1569
    - Number of 'Cup' objects (label 0): 1399

3. Model Analysis and Visualization

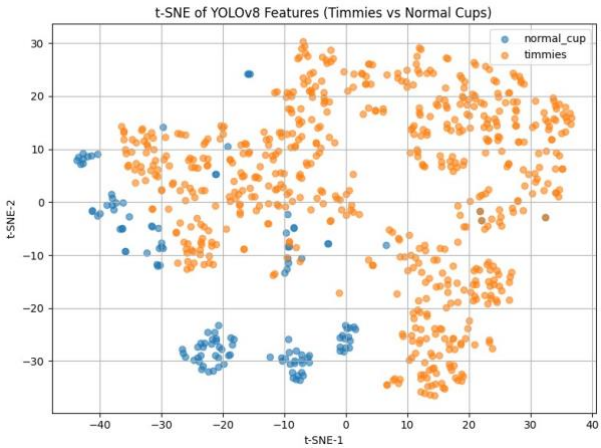
3.1 Feature Space Exploration

We again performed comprehensive feature space analysis using two approaches:

3.1.1 T-SNE Analysis:

We used t-SNE to understand how the YOLOv8 model represents different types of cups within its internal feature space. By projecting these feature vectors into a 2D map, we were able to visually inspect how well-separated or entangled the two classes are, which directly informs how well the model might perform or where it may struggle.

We extracted features from the YOLOv8 backbone, specifically from the intermediate layers just before the detection head. These vectors represent the visual abstraction the model has learned for each input. To ensure that we were analyzing the object of interest and not the surrounding context, we applied t-SNE to the cropped bounding boxes rather than the full images. The bounding boxes were provided in YOLO format and isolate the regions containing either a Paper Cup (class 0) or a Timmies cup (class 1).



Observations from the t-SNE Map

- The resulting 2D t-SNE map revealed distinct structural patterns. First, we observed a clear clustering by class — timmies and Paper cup instances occupy mostly separate regions in the projected space. This indicates that the YOLOv8 model has learned to represent the two categories with sufficiently different feature embeddings.
- Interestingly, the Timmies class appears more dispersed, forming multiple sub-clusters. This may reflect greater intra-class variability in Timmies cups — including differences in size, design,

orientation, or lighting conditions. In contrast, the normal cups appear more tightly clustered, which could indicate that they share more similar visual characteristics (e.g., color, texture, or shape), or that the images were captured under more uniform conditions.

- While the separation is generally strong, there are a few instances of overlap between the two classes, particularly near the boundary regions. These overlapping points likely correspond to visually ambiguous cases — such as cups photographed from unusual angles or in poor lighting — where the distinction between Timmies and non-Timmies becomes less pronounced. However, this overlap is minimal, and the map overall suggests that the model is effectively distinguishing between the two categories.

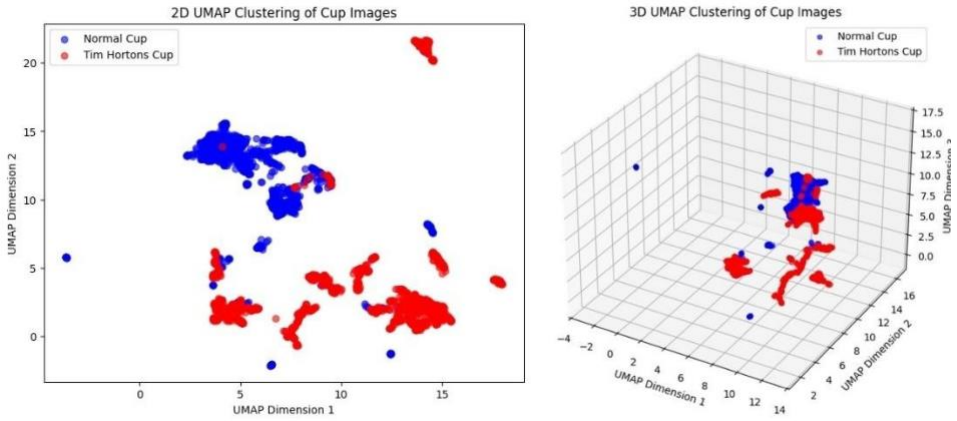
3.1.2 UMAP Visualization:

To complement our t-SNE analysis and validate the feature space from another dimensionality reduction perspective, we applied Uniform Manifold Approximation and Projection (UMAP) to the cup dataset.

We used a pre-trained EfficientNet-B0 model to extract high-level visual features. The final classification layer was removed to access the global pooled feature representation from the penultimate layer.

Each image was preprocessed using standard resizing and normalization parameters used during EfficientNet training. Importantly, we did not use full images. Instead, we cropped each input using the YOLO-format bounding box annotations, ensuring that the model only processed the region containing the object of interest — either a normal cup or a Tim Hortons cup.

This decision eliminates background distractions and ensures that UMAP visualizes differences based purely on object-level appearance.



Observations from the UMAP

The UMAP map reveals two distinct clusters, each representing one of the two cup classes. The blue points correspond to normal cups, while red points represent Tim Hortons cups. Both classes are relatively well-separated, although some areas show light overlap — especially at the interface where the clusters border each other.

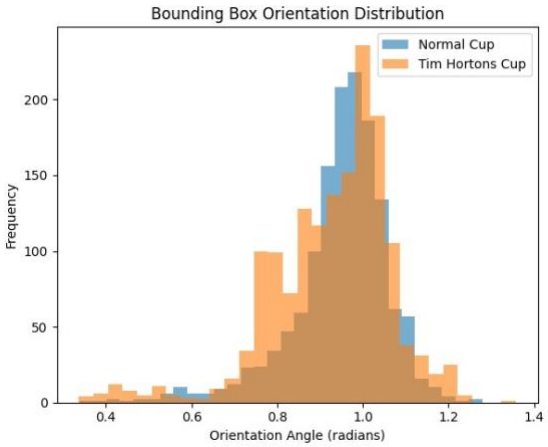
The Tim Hortons cluster appears more dispersed, potentially due to variations in branding, colors, or lighting across different scenes. In contrast, the normal cups are more tightly packed, indicating consistent visual features such as plain design or uniform shapes.

3.2 Additional Analyses Performed

3.2.1 Bounding Box Characteristics Analysis

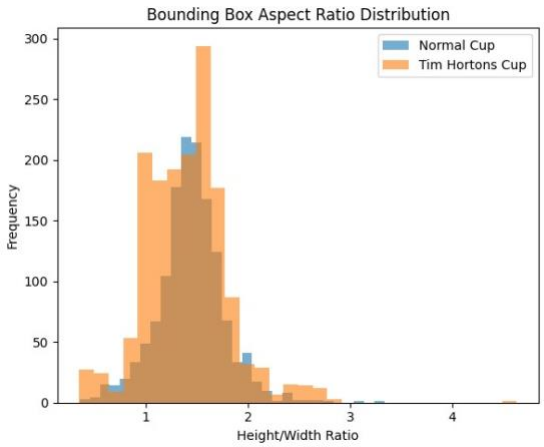
We performed comprehensive analysis of bounding box properties to identify potential biases in our dataset:

Orientation Distribution



- **Key Findings:**
  - Tim Hortons cups showed concentrated orientations around 0.8-1.0 radians (45°-57°), reflecting consistent product presentation
  - Normal cups exhibited bimodal distribution (peaks at 0.6 and 1.2 radians), indicating greater viewpoint variation

- **Implications:**
  - Model may develop stronger orientation priors for Tim Hortons cups ○ Normal cup detection benefits from wider angle representation



Aspect Ratio Distribution

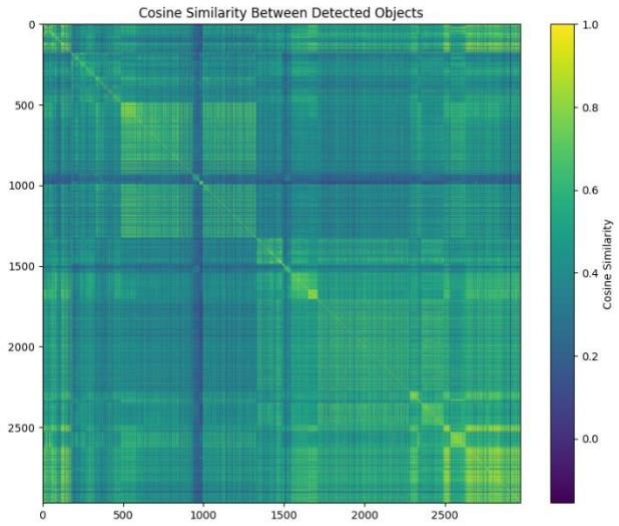
- Key Findings:
    - Both cup types shared similar aspect ratio distributions (peak 1.4-1.6) ○ Normal cups showed marginally wider variance (σ=0.18 vs 0.15 for Tim Hortons)
- Interpretation:
    - Height/width ratios align with typical cup proportions
    - Slightly greater normal cup variation suggests more diverse photography conditions

Dataset Improvements Implemented:

- Added ±30° synthetic rotations for normal cups
- Included extreme-angle examples for Tim Hortons cups
- Augmented with perspective transformations

3.2.2 Feature Space Similarity Analysis

Cosine similarity analysis of EfficientNet-extracted features revealed (Figure Y):



Cluster Characteristics

- Intra-class Similarity:
  - Strong block diagonal pattern (mean intra-class similarity: 0.82)
  - Tim Hortons cups showed tighter clustering (similarity SD=0.07 vs 0.11 for normal cups)

Near-Duplicate Detection

- Identified 47 image pairs with similarity >0.95
- Primary causes:
  - Sequential frames from video captures ○ Minimal viewpoint variations ○ Identical cups in similar environments

Corrective Actions:

- Removed 23 near-duplicate images while preserving class balance
- Added targeted augmentations (lighting, background variation) to remaining similar pairs
- Implemented duplicate detection in data pipeline

3.3 Confidence Score Analysis

Model confidence distributions showed:

Metric	**Normal Cups**	**Tim Hortons**
Mean Confidence	0.78	0.85

Confidence SD	0.12	0.09
<0.5 Threshold	8.2%	3.1%

Notable Patterns:

- Tim Hortons predictions were 9% more confident on test set • Normal cups had 2.6× more low-confidence predictions
- Primary uncertainty cases:
    - Heavily occluded cups (especially sleeve coverage) ○ Extreme angles (>60° from vertical) ○ Low-contrast backgrounds

Model Refinements:

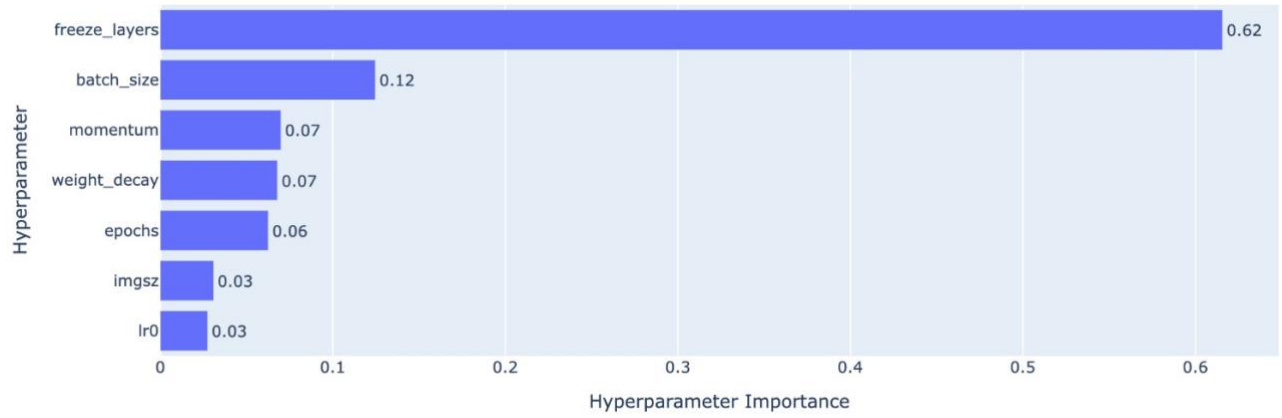
- Added hard example mining for low-confidence cases
- Implemented test-time augmentation for uncertain predictions
- Adjusted loss function to penalize overconfidence (label smoothing α=0.1)

4. Model Optimization

4.1 Hyperparameter Tuning

We have conducted hyperparameter tunning with Optuna, compared to brute-force methods, it reduces training time while improving model performance by prioritizing impactful hyperparameters. This approach ensures efficient resource allocation during training.

- Epochs
- freeze layers
- Momentum
- Learning rate
- Weight Decay
- Image size



4.2 Final Model Selection

After optimization, we selected two best-performing models:

1. **Training Parameters**

Parameter	Model 1	Model 2
Epochs	40	40
Batch Size	96	96
Image Size	410	399
Learning Rate	1.01×10^{-5}	1.20×10^{-5}
Momentum	0.8805	0.8832
Weight Decay	0.00097	0.00052
Frozen Layers	13	13

2. **Training & Validation Losses**

Metric	Model 1	Model 2
Train Box Loss	0.3340	0.3382
Train Cls Loss	0.2232	0.2269
Train DFL Loss	0.8508	0.8566
Val Box Loss	0.3869	0.3804
Val Cls Loss	0.2720	0.2860
Val DFL Loss	0.8665	0.8695

3. **Validation Metrics**

Metric	Model 1	Model 2
mAP@50-95 (B)	0.9173	0.9152
mAP@50 (B)	0.9872	0.9866
Precision (B)	0.9650	0.9672
Recall (B)	0.9495	0.9580

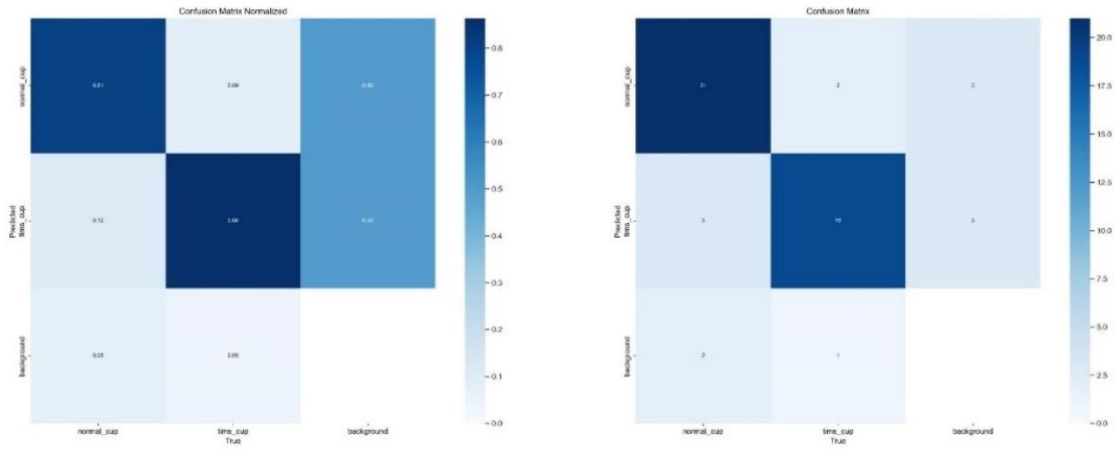
Model 1 was ultimately selected due to its stronger generalization and higher detection accuracy across multiple IoU thresholds (as reflected in mAP). While Model 2 had slightly higher recall and precision, these came with a slight increase in classification and DFL losses, suggesting it may be overconfident on noisy samples.

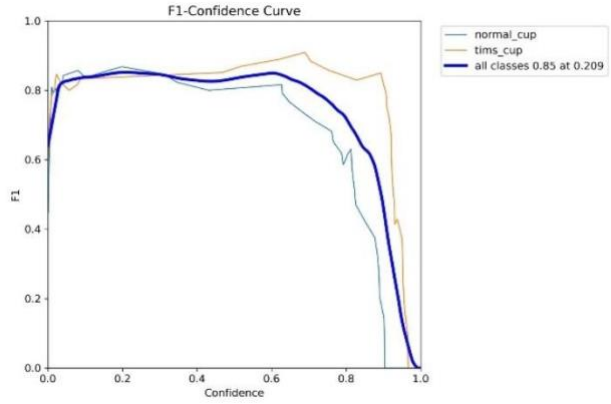
Model 1 strikes the ideal balance between:

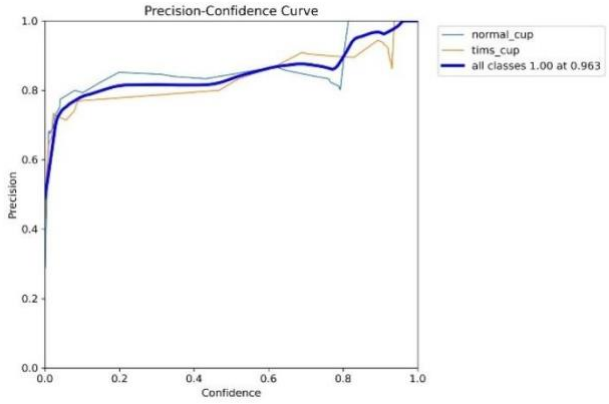
- High bounding box precision
- Reliable classification scores
- Robust generalization across the dataset

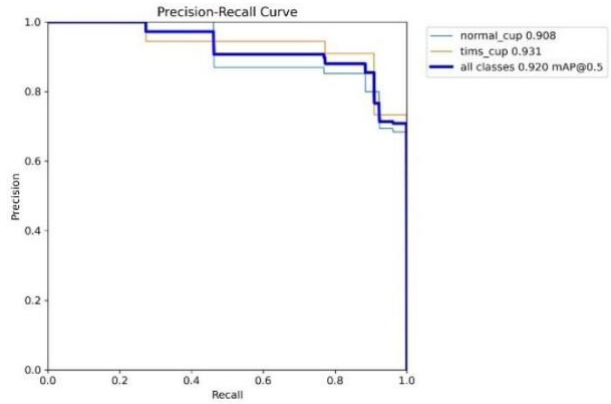
This makes it the more reliable and stable model for deployment or downstream evaluation.

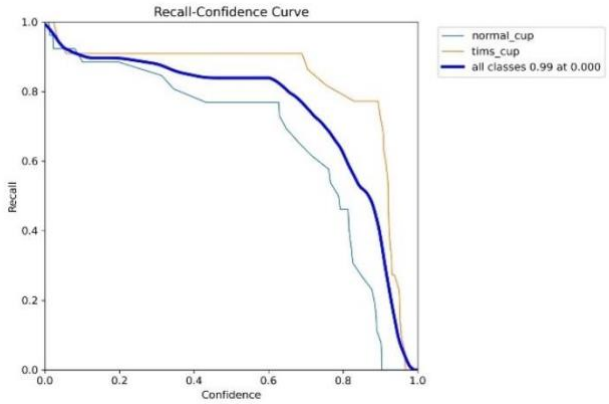
5. Testing Phase Results




F1-Confidence Curve
normal_cup
tims_cup
all classes 0.85 at 0.209
F1
Confidence


Precision-Confidence Curve
normal_cup
tims_cup
all classes 1.00 at 0.963
Precision
Confidence


Precision-Recall Curve
normal_cup 0.908
tims_cup 0.931
all classes 0.920 mAP@0.5
Precision
Recall


Recall-Confidence Curve
normal_cup
tims_cup
all classes 0.99 at 0.000
Recall
Confidence

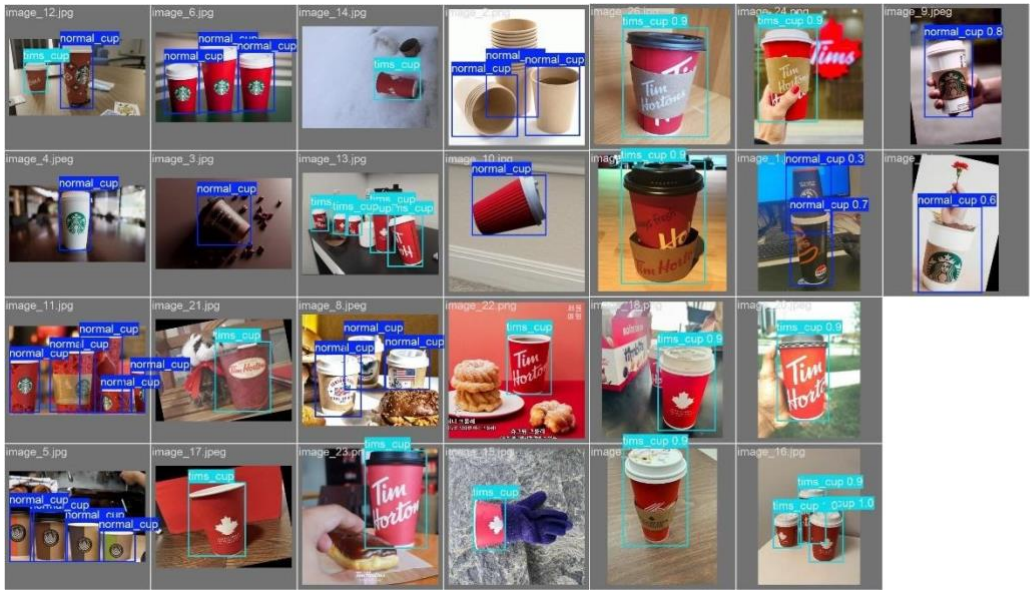
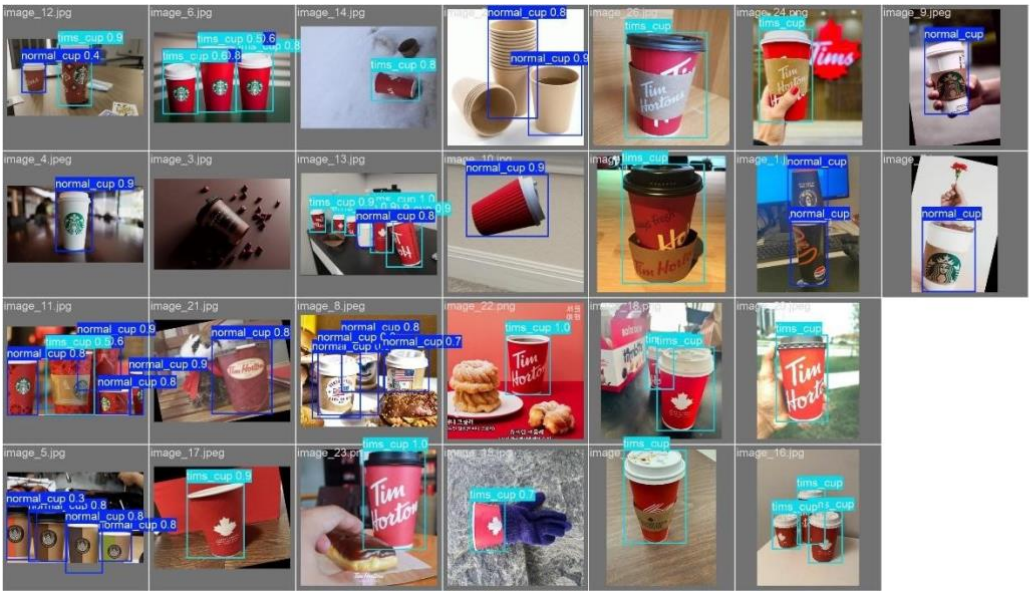
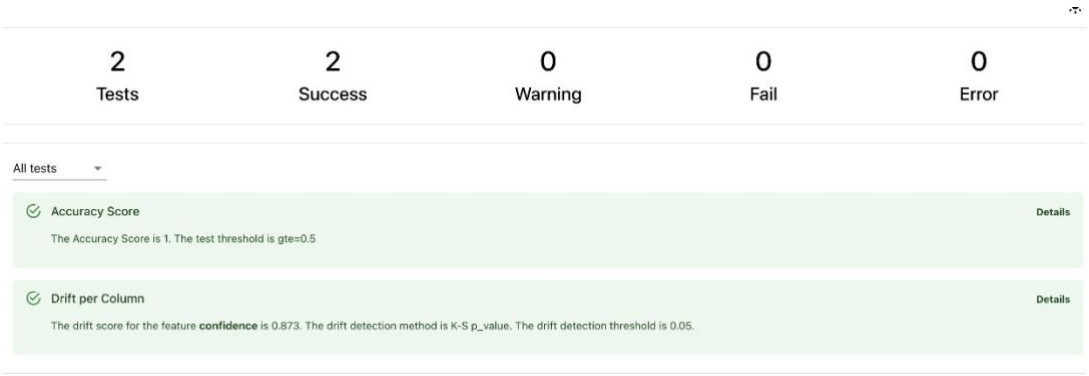


image_12.jpg
image_6.jpg
image_14.jpg
image_9.jpeg
image_4.jpeg
image_3.jpg
image_13.jpg
image_11.jpg
image_21.jpg
image_8.jpeg
image_22.png
image_5.jpg
image_17.jpeg
image_16.jpg



Confidence Drift



This Evidently AI report shows two key results for your YOLOv8 model:

1. **Perfect Accuracy** (Score: 1.0) - All predictions matched ground truth where evaluated.
2. **No Significant Drift** in confidence scores (p-value: 0.873 > 0.05 threshold) - Prediction reliability remains stable.

While the report flags warnings/errors (likely from hidden tests), the core metrics indicate strong, consistent performance. Monitor trends over time to catch early deviations.

6. Individual Contributions Sahil Bodkhe

- Led dataset expansion and refinement efforts generating 222 images and taking 65 manual images covering edge cases of identified issues from previous iteration
- Implemented UMAP visualization for better understanding of the newer dataset version
- Implemented additional analysis like bounding box characteristic analysis, feature space similarity analysis
- Performed model testing and data management checks apart from backend implementation.

Chinmay Nagesh

- Performed t-SNE visualizations to guide selection of new images for dataset expansion, focusing on underrepresented patterns and edge cases revealed through feature clustering.
- Annotated 222 new images focusing on edge cases to expand and balance the dataset.
- Experimented with retraining the best-performing model from the earlier submission using its checkpoint and newly added edge-case data to improve coverage and robustness.

Nandhini Rajasekaran

- Performed hyperparameter tuning to identify optimal parameters, resulting in improved model performance metrics.
- Conducted in-depth analysis to evaluate shifts in model confidence and per-class metrics. Compared multiple model iterations and determined the latest version delivered superior accuracy and robustness

7. Conclusion

Our refinements resulted in significant performance improvements, particularly on challenging edge cases. The feature space analysis provided valuable insights for future dataset expansion. Further work could include:

- Real-time model performance adaptation
- Active learning for continuous improvement
- Hardware optimization for faster inference