Verizon #1: Identifying Accessories in Retail with Real-Time Vision Algorithm

AI Studio Final Presentation

University of California, Los Angeles
December 4th, 2024

Introductions

Meet Our Team!

Anastasiia Kiseleva
CSU Northridge

Lulu Shao
UC San Diego

Chris Taguba
UC San Diego

Mariia Nikitash
CSU Fullerton

Emily Zheng
UC Santa Barbara

Our AI Studio TA and Challenge Advisors

Aryaman Gokarn
AI Studio TA

Aleksandar Saric
Challenge Advisor

Dustin Pulver
Challenge Advisor

Presentation Agenda

1. Introduction
2. Project Overview
3. Specific Implementations
    a. Data Preparation
    b. Case Design Classification Model
    c. Segmentation Models
    d. Post Processing
    e. Phone Classification
4. Conclusion
5. Future Improvements

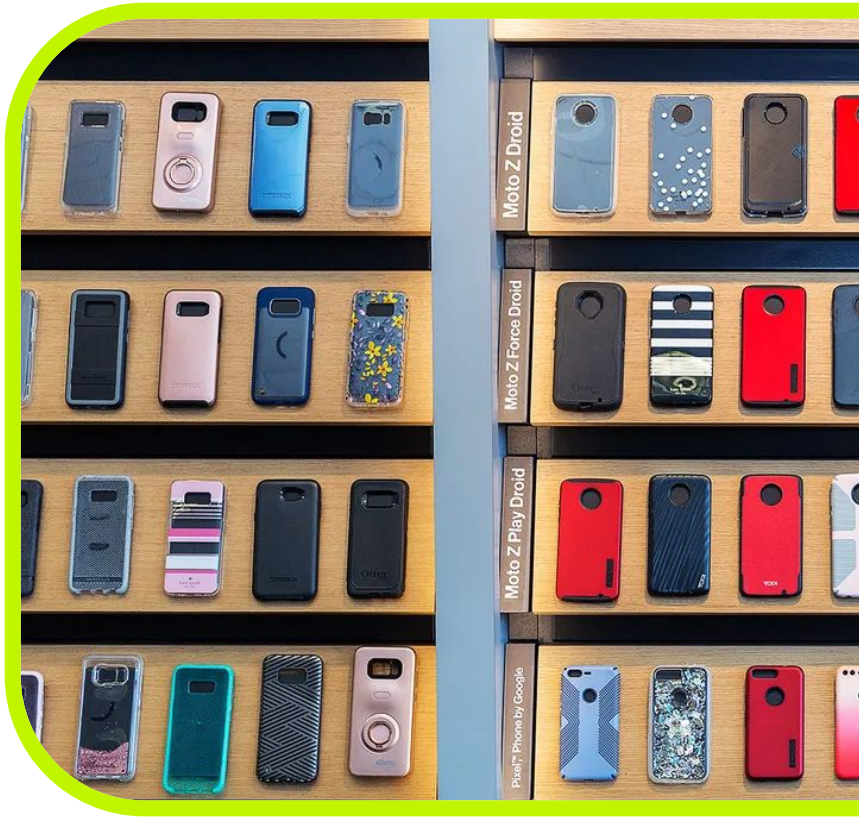
Project Overview

Develop a vision-based system to identify and recommend compatible accessories for different phone device models in real time.

Business Impact

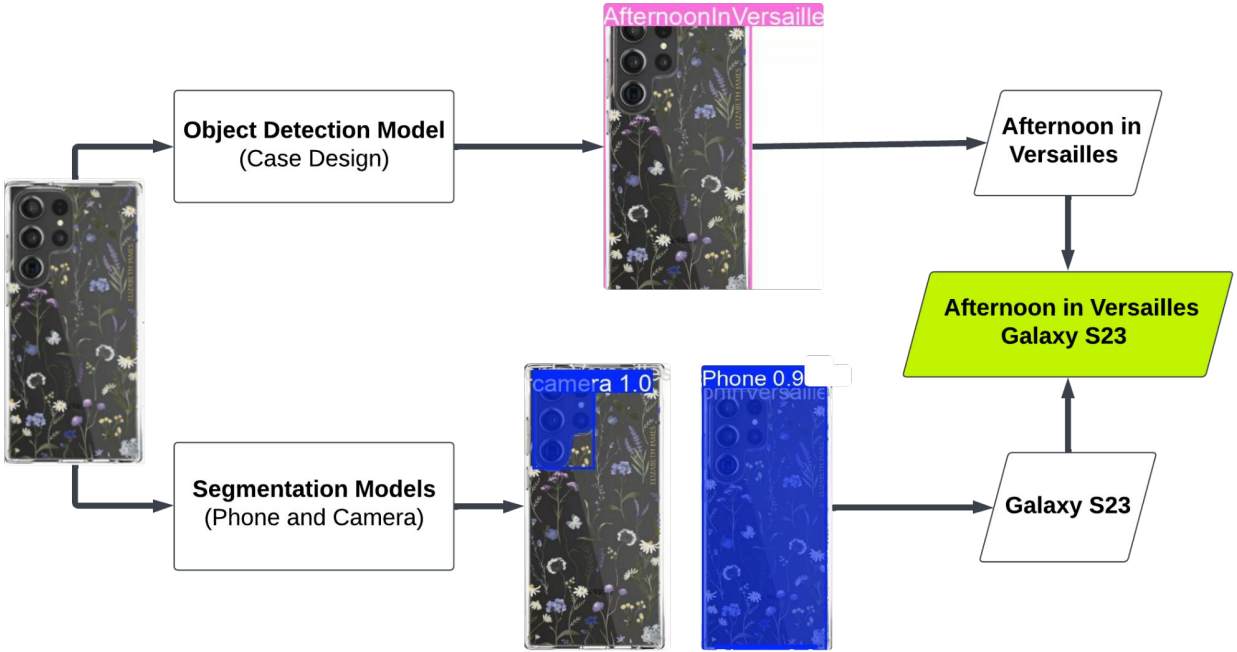
By gathering and training on diverse data, the model could:

- Distinguish between visually similar products (e.g. phone cases)
- Reduce dependent on staff for product compatibility checks
- Minimize returns due to mismatches
- Scale to other retail product categories and platforms
- Enhance efficiency in the in-store shopping experience.

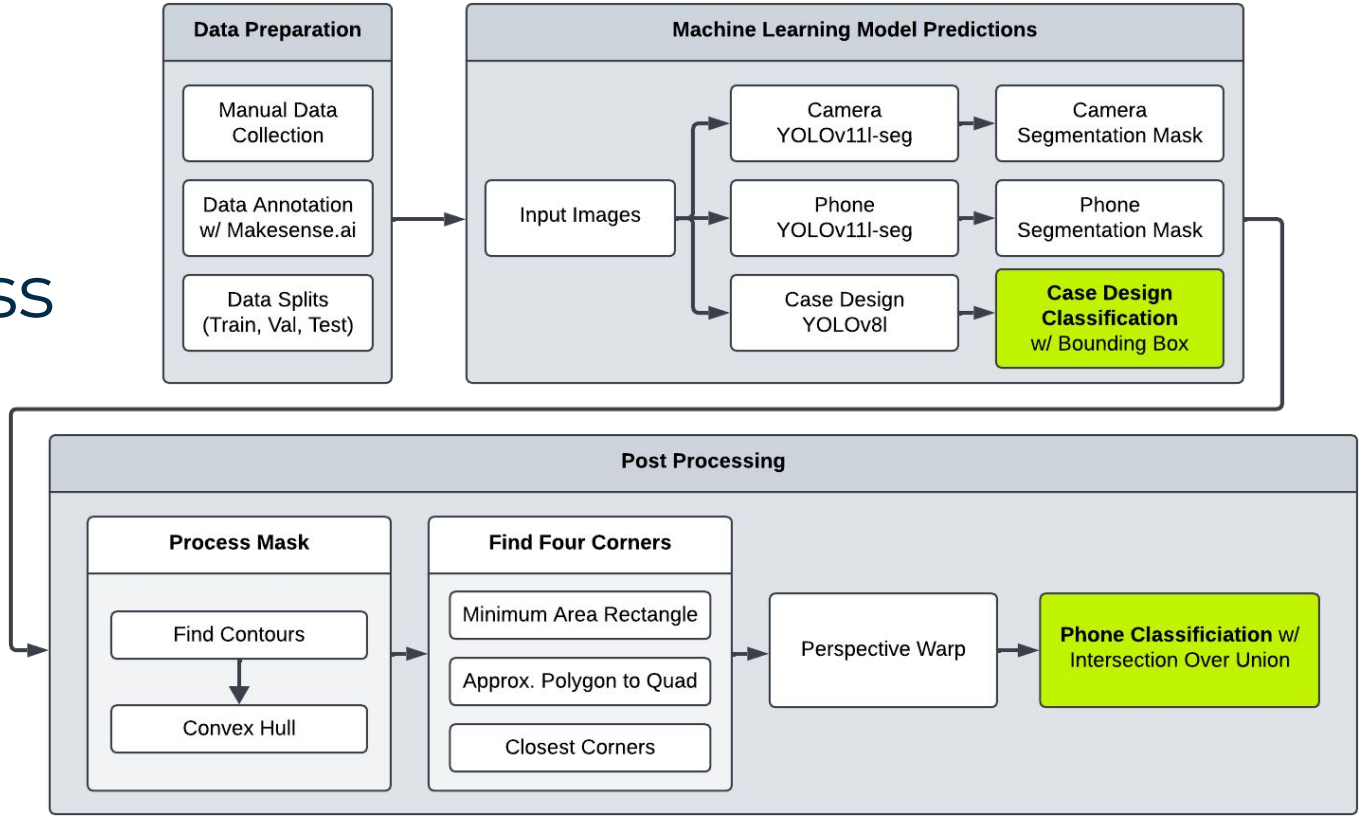


Our Process

1. Train an object detection model for identifying phone case design
2. Train a segmentation model for identifying phone models
3. Combine the pipeline into a real-time identification model for phone classification



Our Process


Data Preparation
Manual Data Collection
Data Annotation w/ Makesense.ai
Data Splits (Train, Val, Test)
Machine Learning Model Predictions
Input Images
Camera YOLOv11l-seg
Camera Segmentation Mask
Phone YOLOv11l-seg
Phone Segmentation Mask
Case Design YOLOv8l
Case Design Classification w/ Bounding Box
Post Processing
Process Mask
Find Contours
Convex Hull
Find Four Corners
Minimum Area Rectangle
Approx. Polygon to Quad
Closest Corners
Perspective Warp
Phone Classificiation w/ Intersection Over Union

Our Process

Data Preparation

- Collect diverse image set
- Label bounding boxes and segmentation masks
- Split dataset directory into YOLO format

Classify Phone Models

- Train segmentation models
- Transform images with OpenCV
- Use transformed segmentation masks to classify phone model

Model Training and Tuning

- Explore YOLOv8 models
- Perform hyperparameter tuning
- Find list of best parameter values

Finalizing Models

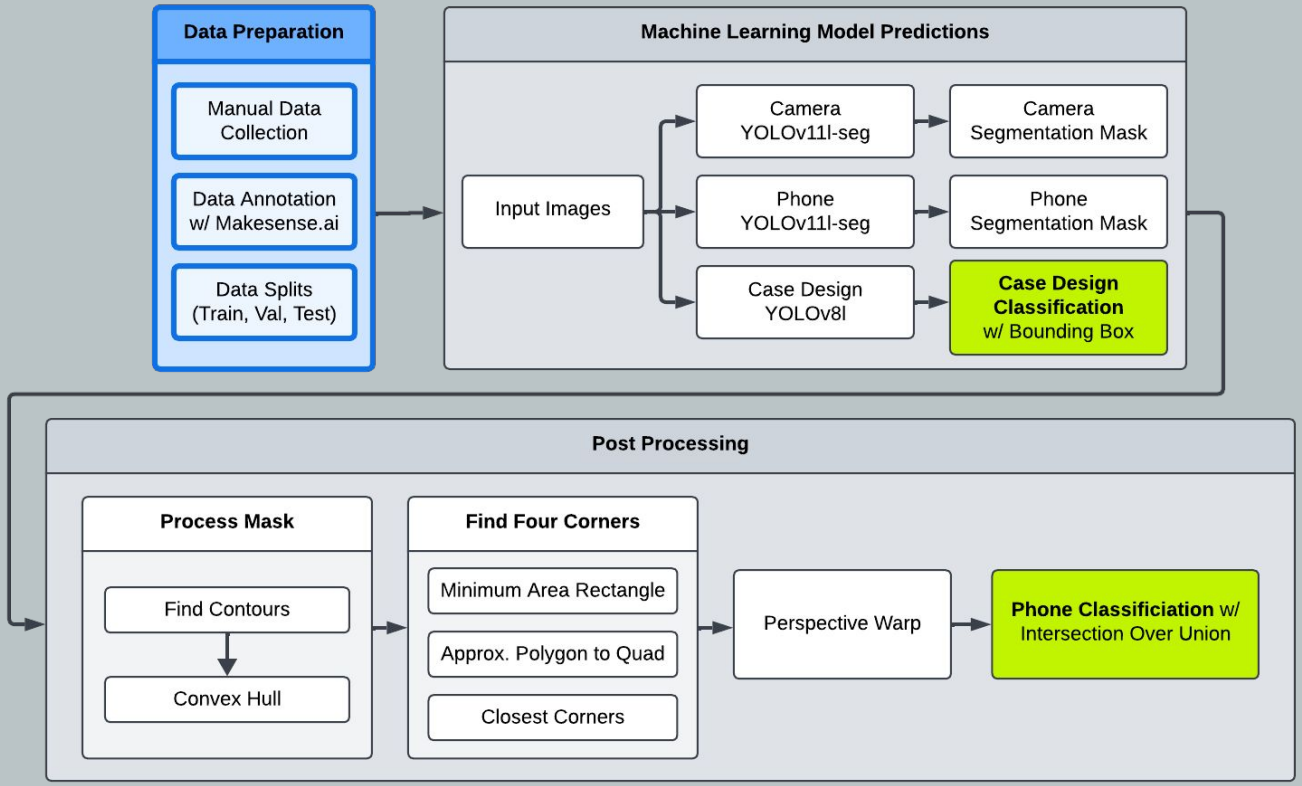
- Prepare for final presentation
- Improve model accuracy
- Connect classification pipeline

Resources Leveraged

- **Google Colab:** Collaborative cloud-based IDE
- **Ultralytics YOLO:** Object detection models
- **PyTorch:** Deep learning framework
- **OpenCV:** Image processing
- **Makesense.ai:** Data annotation







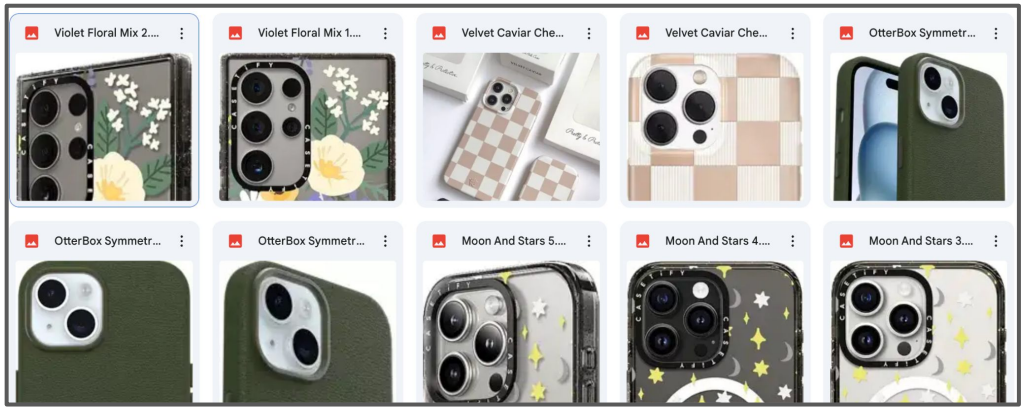
Data Preparation

Data Understanding

Collected a **diverse** dataset of phone case images that vary in:

- Case Color
- Design Details
- Phone Model
- Phone Brand

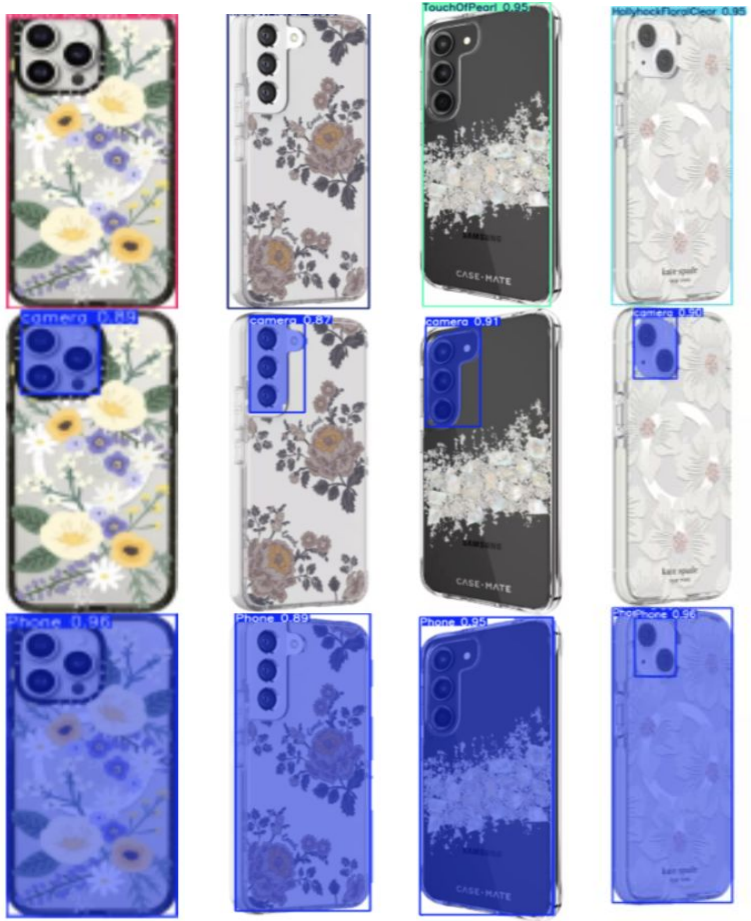
- Orientation
- Rotation
- Scale
- ...other transformations



Data Preparation

Approach:

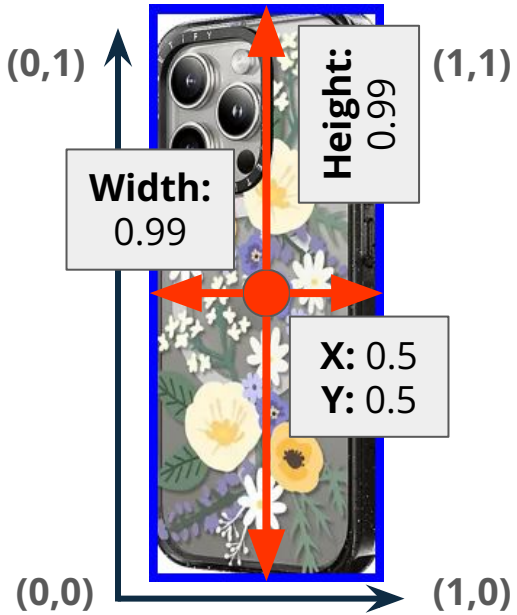
- **Dataset 1:** Case design
- **Dataset 2:** Camera cutout
- **Dataset 3:** Phone area



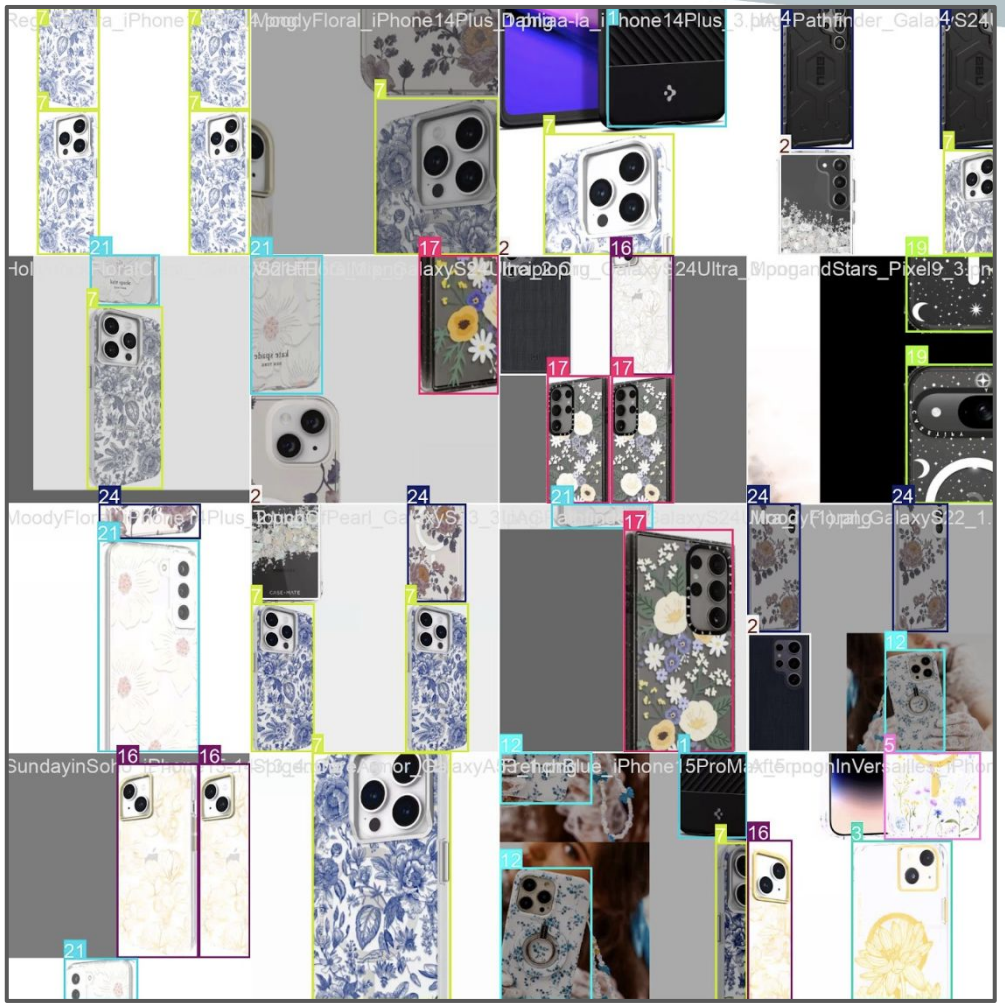
Data Preparation

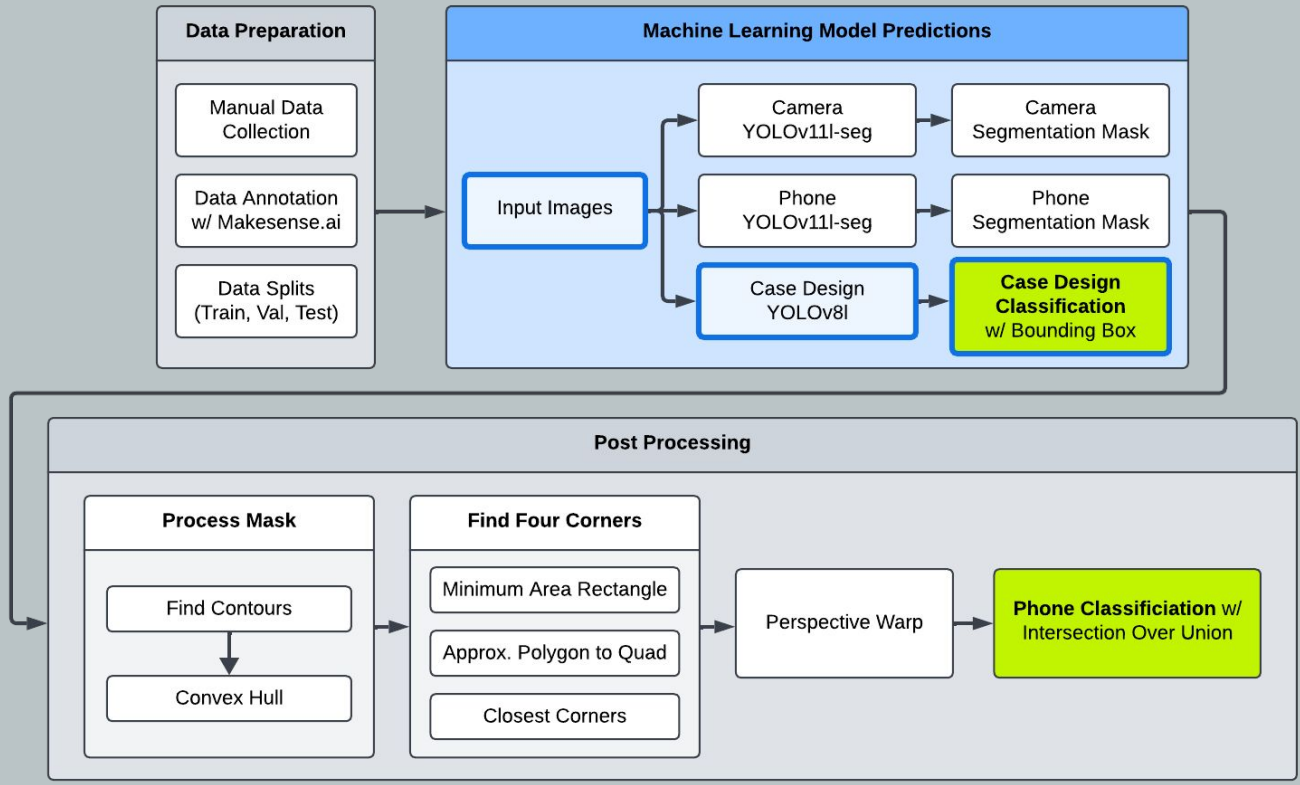
- Created corresponding .txt label file for each .png image file
- Split datasets into training (70%), validation (20%), testing (10%) partitions

VioletFloralMix_iPhone15ProMax_5.txt:
17 0.500939 0.501739 0.998121 0.996522



Training Input





Case Design Classification

Model Training

Fine-tuned a large, pre-trained **YOLOv8** model

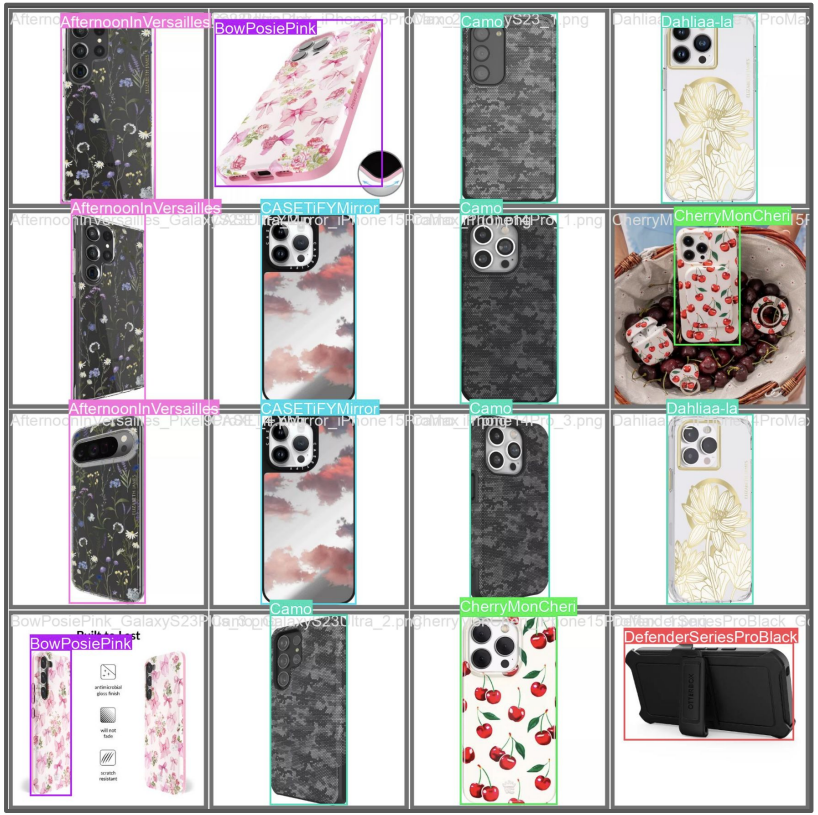
- Achieved a **95% precision** to minimize false positives (mismatched purchases)
- Stayed within computing resources (15 GB of GPU credits in Colab)

Model Training and Tuning

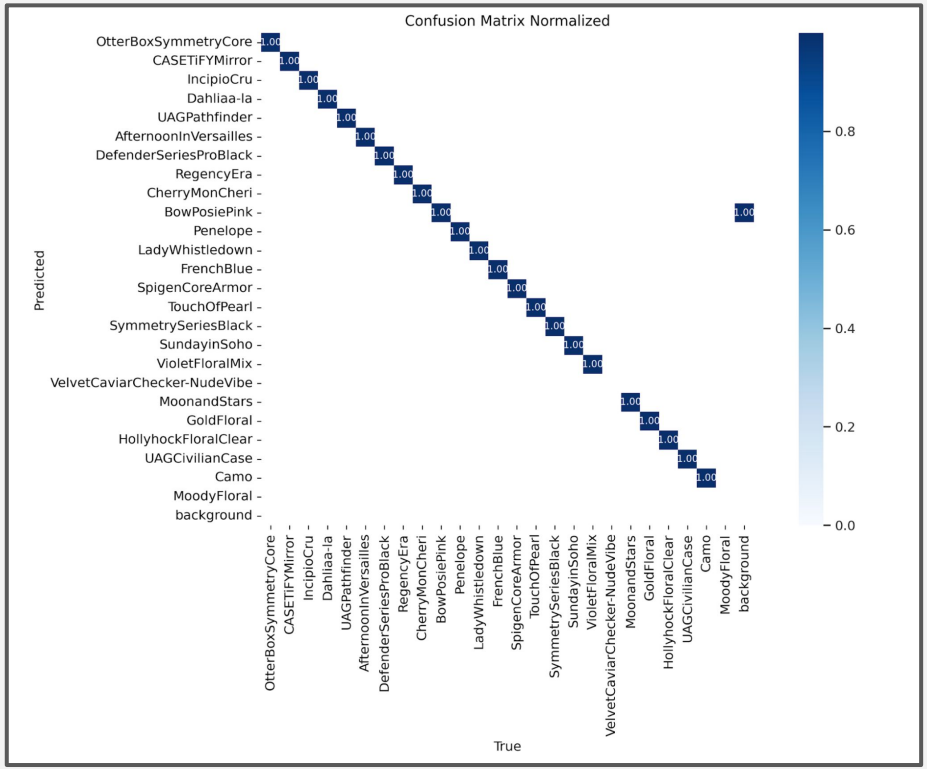
Tuned hyperparameters using custom training loop and Ultralytics built-in `tune()` method

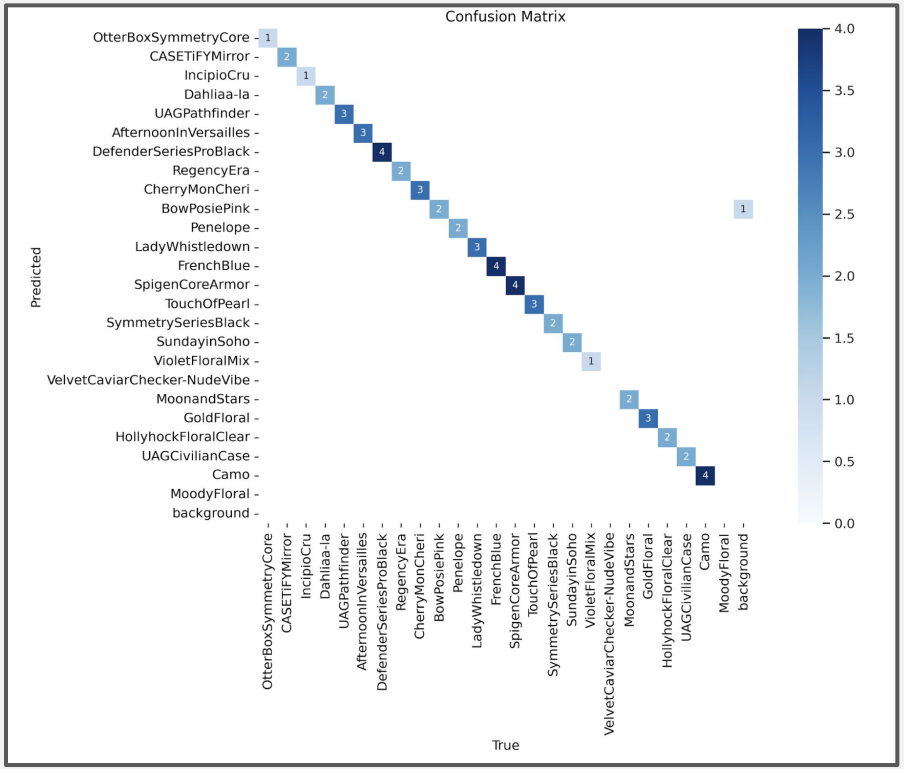
- Defined list of values to test in model training
- Analyzed which values scored the best precision
- Data augmentations include scale, translation, HSV adjustment, brightness, etc. (see image)

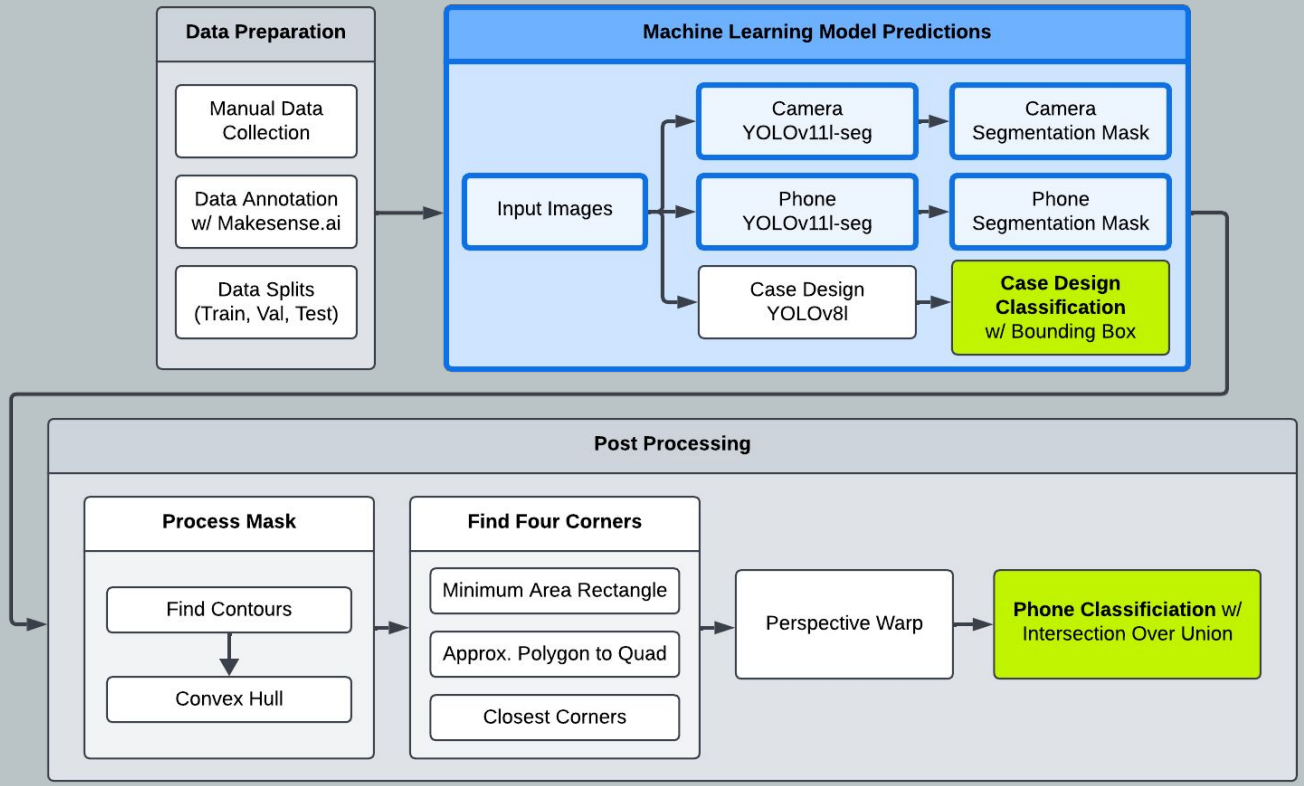
Design Classifications



Graphs and Results



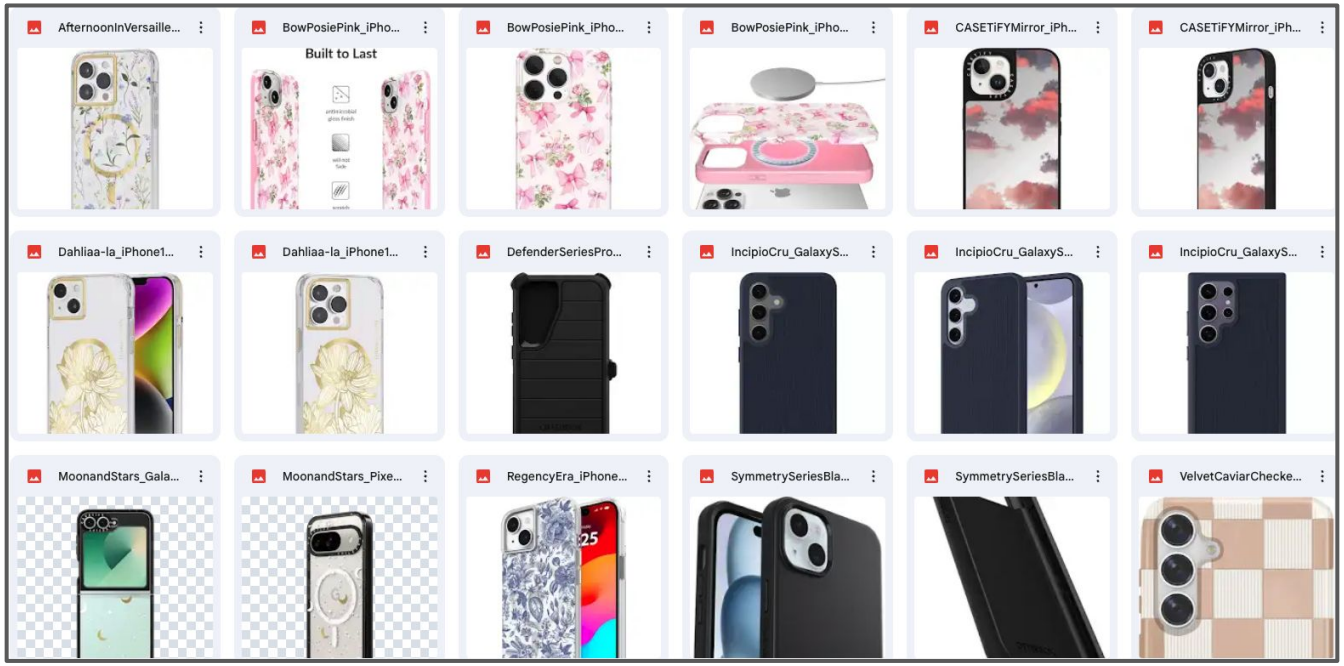




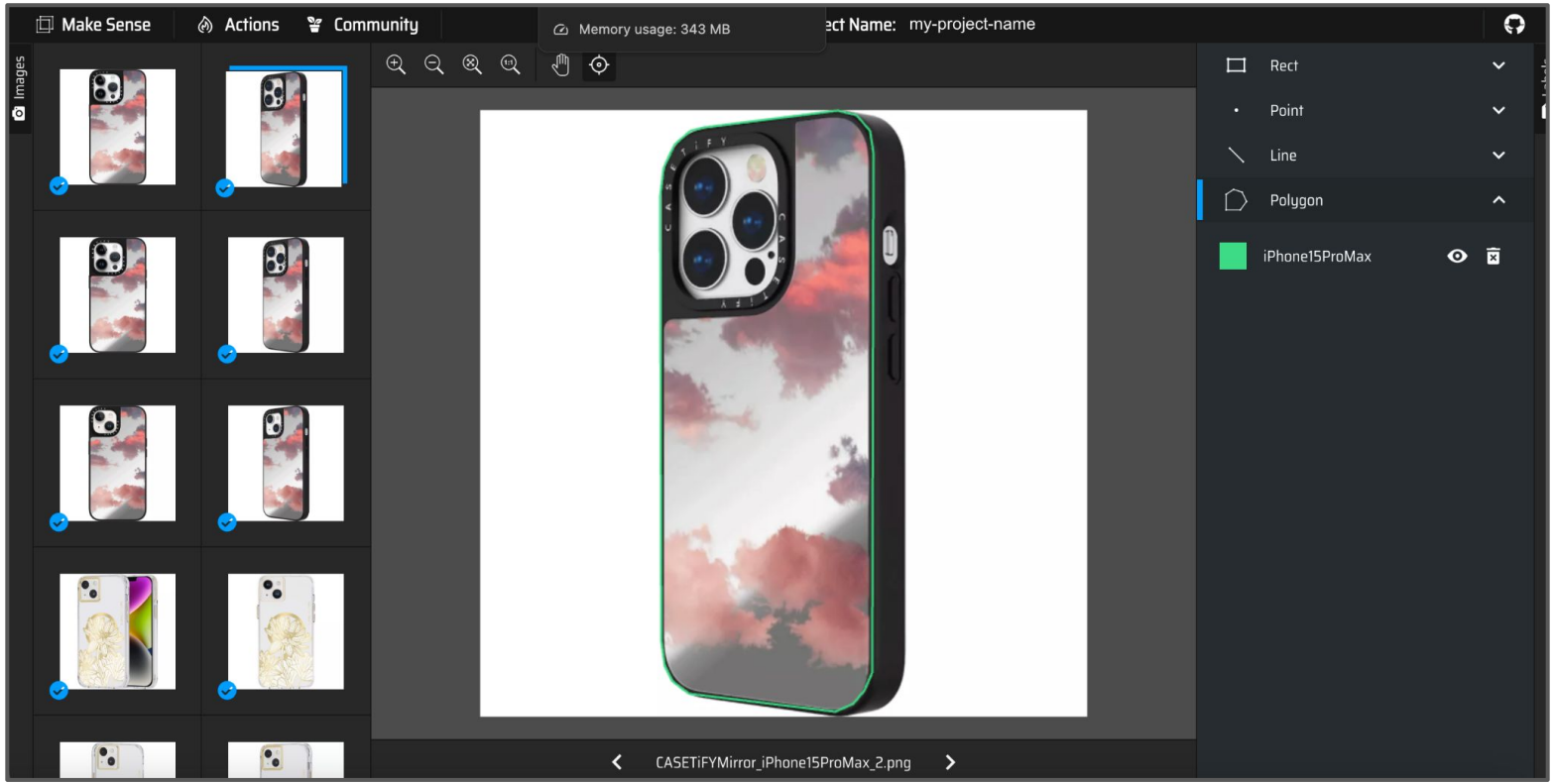
Phone and Camera Segmentation

Segmentation Model - Data

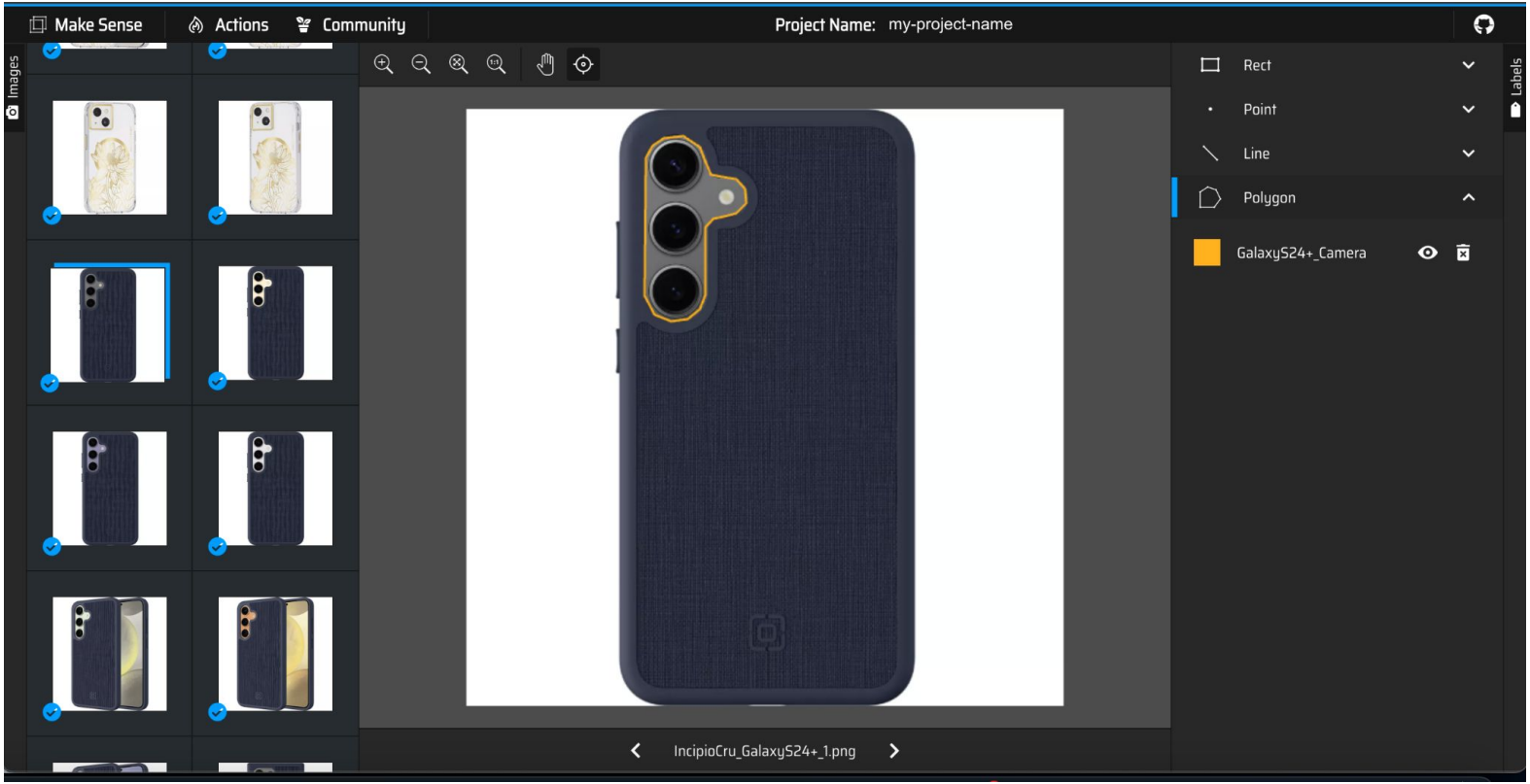
- 1st dataset is used to detect the area of a phone
- 2nd dataset is used to detect its camera cutout



Segmentation Model - Data - Phone Cutouts

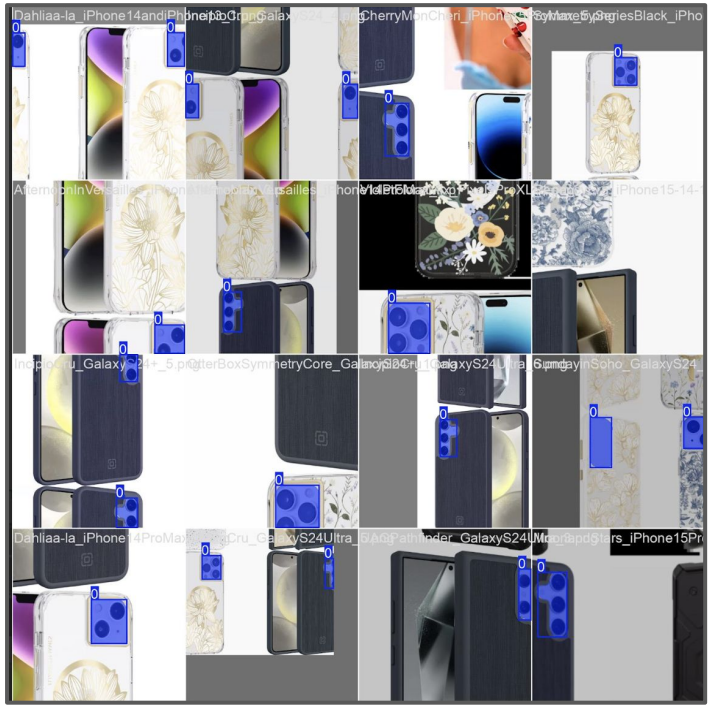


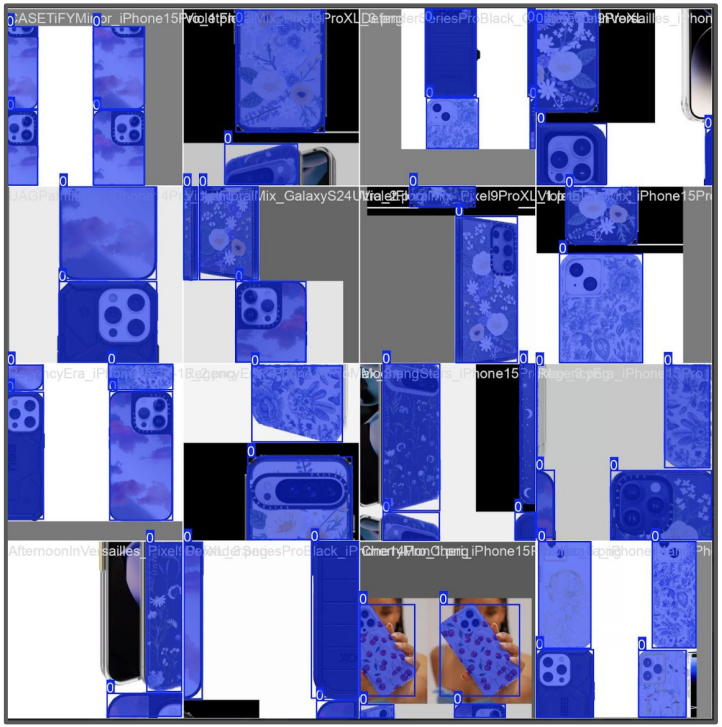
Segmentation Model - Data - Camera Cutouts



Segmentation Model - Data

Below are some example training batches for the camera cutouts and phone areas:



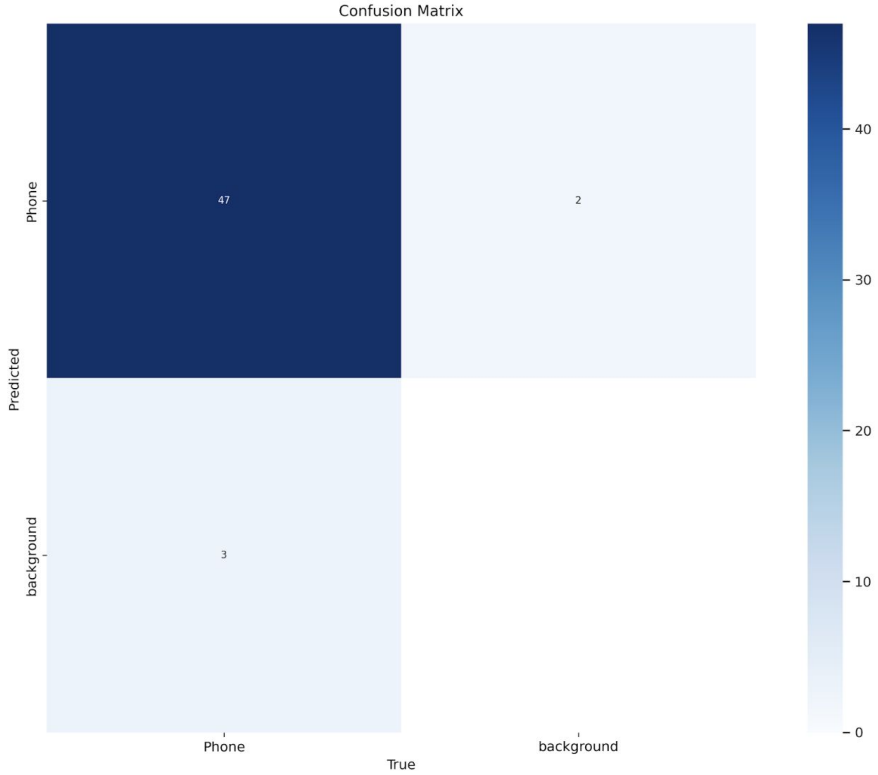


YOLO Segmentation - Results (Phone)

Input

Output

YOLO Segmentation - Results (Phone)

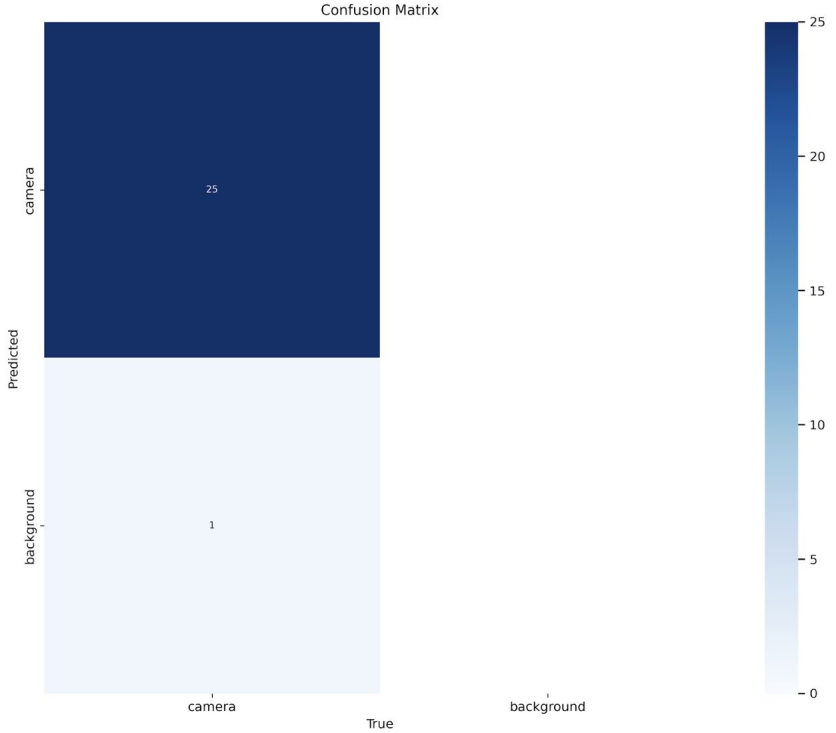


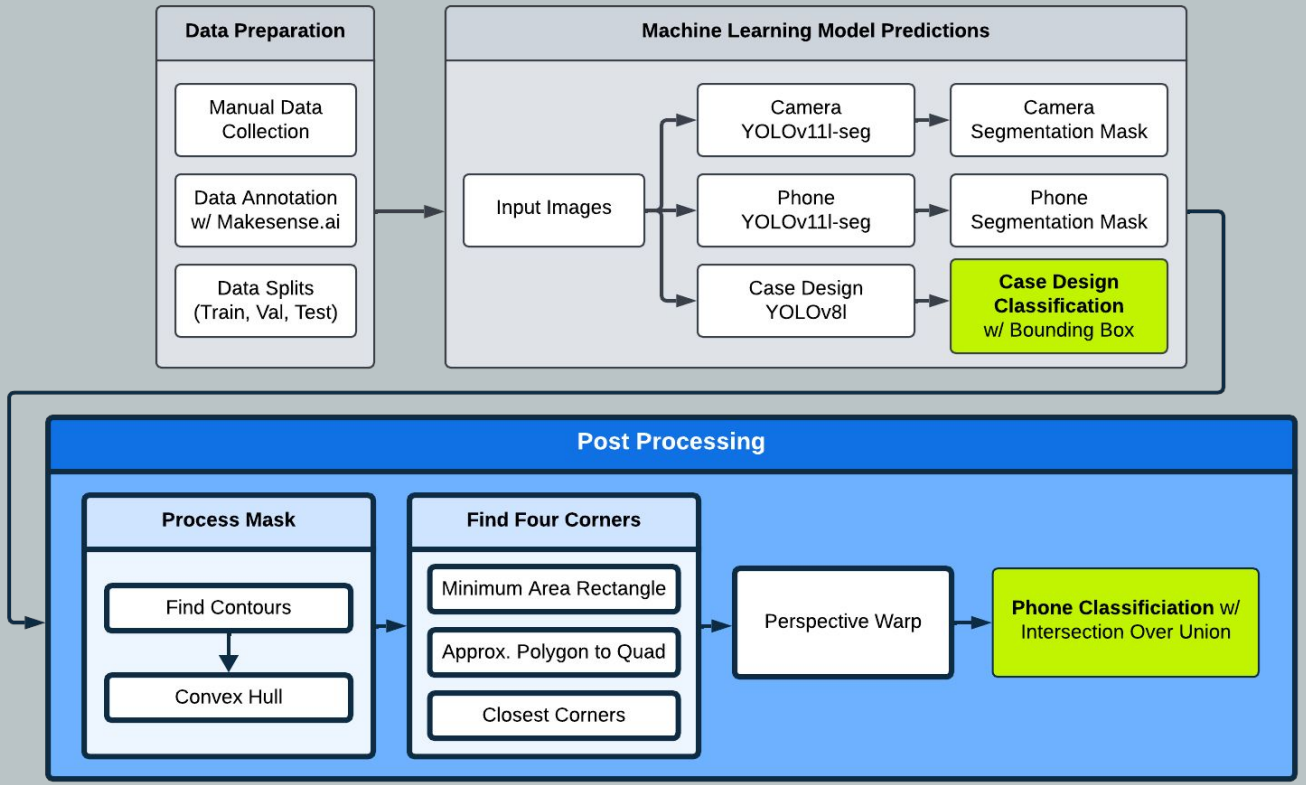
YOLO Segmentation - Results (Camera)

Input

Output

YOLO Segmentation - Results (Camera)



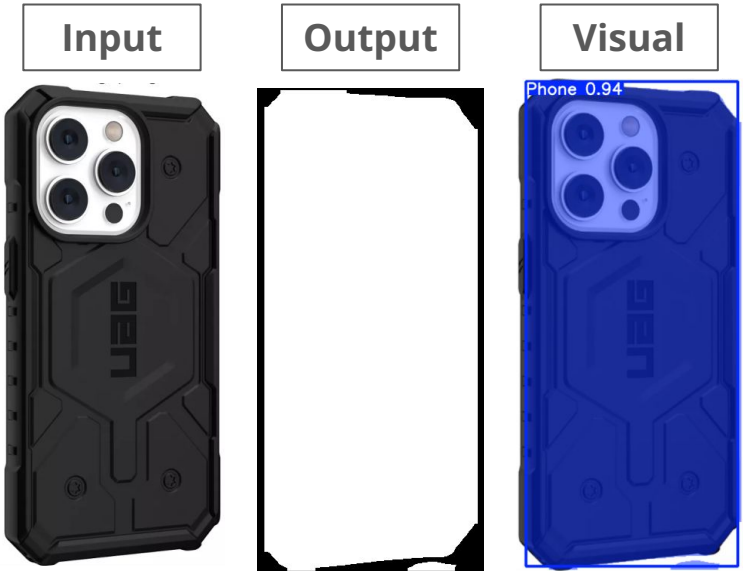


Post Processing

Classify Phone - Process Masks

Process masks to make them compatible with NumPy and OpenCV operations

- Convert PyTorch Tensor to NumPy Array
- Standardize image dimensions (640 x 640) and format (BGR)
- Ensures no errors occur due to image format issues



Classify Phone - Find Contours

1. Use `findContours()` on the phone mask
2. Save the phone contour (assumed to be the largest area contour)
3. Smooth out the contour with `convexHull()`

Classify Phone - Convex Hull

Classify Phone - Four Corners (Rectangle)

1. Use `minAreaRect()` to get a `RotatedRect` (x-center, y-center, size, angle)
2. Calculate the rectangle corners w/ `boxPoints()`.
3. Use the four corners to warp perspective.

Classify Phone - Four Corners (Polygon)

1. Approximate a polygon w/ `approxPolyDP()`.
2. Verify if the polygon is a quadrilateral.
3. Use the four corners to warp perspective.

Classify Phone - Four Corners (Manual)

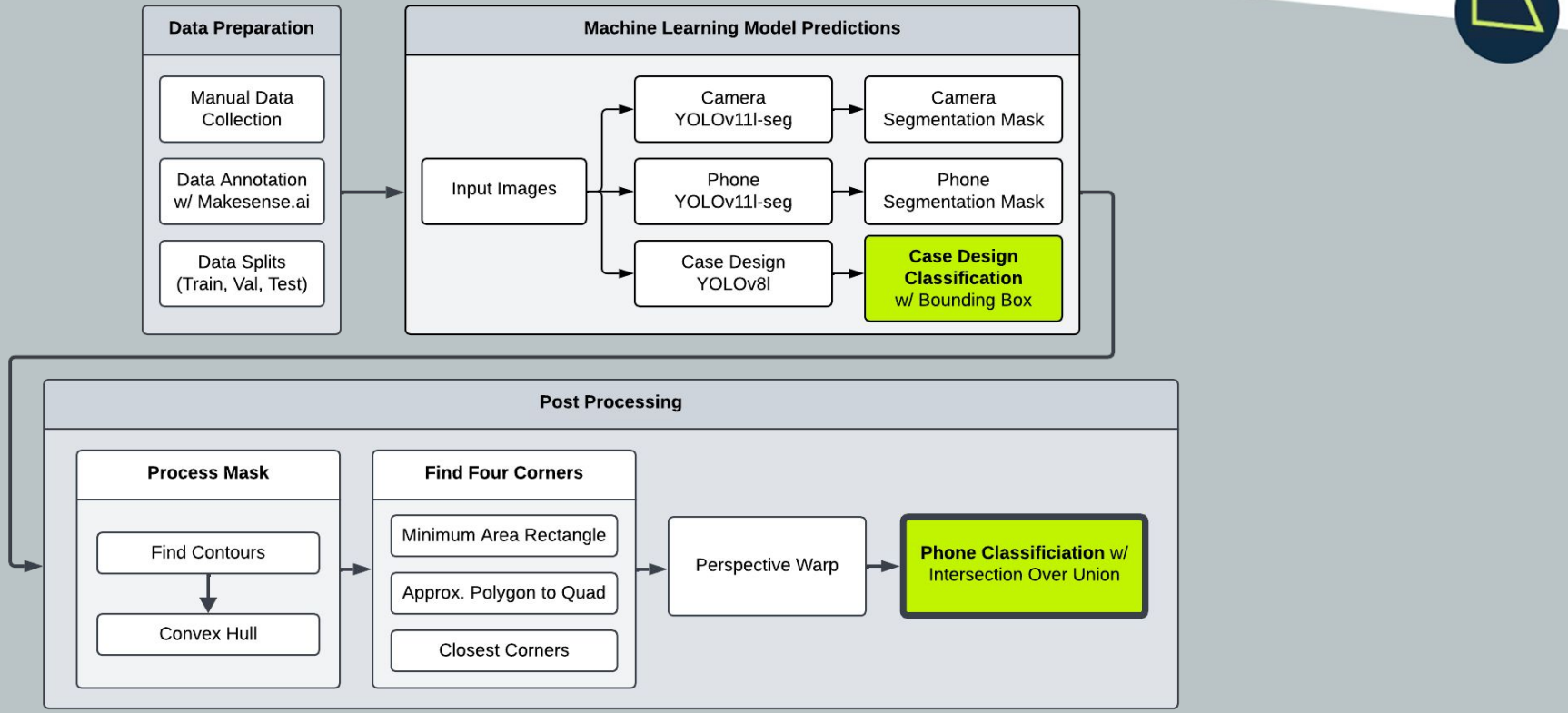
1. For each point in the phone contour, find the point **closest** to each four image corners.
2. Use the four corners to warp perspective.

Classify Phone - Perspective Warp

1. Create a perspective matrix that inputs four corners and outputs to the corners of an **upright** phone.
2. Apply this matrix to the phone and camera masks.

Classify Phone - Perspective Warp

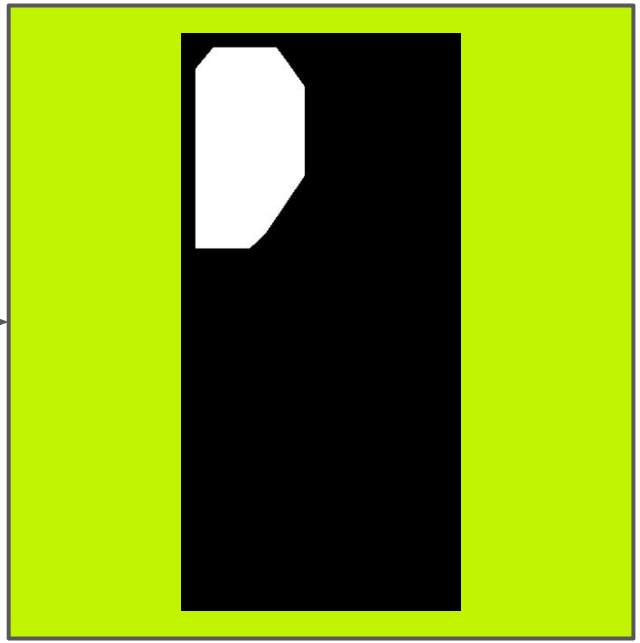
1. Create a perspective matrix that inputs four corners and outputs to the corners of an **upright** phone.
2. Use the matrix to warp the phone and camera into an **upright** position.



Phone Classification

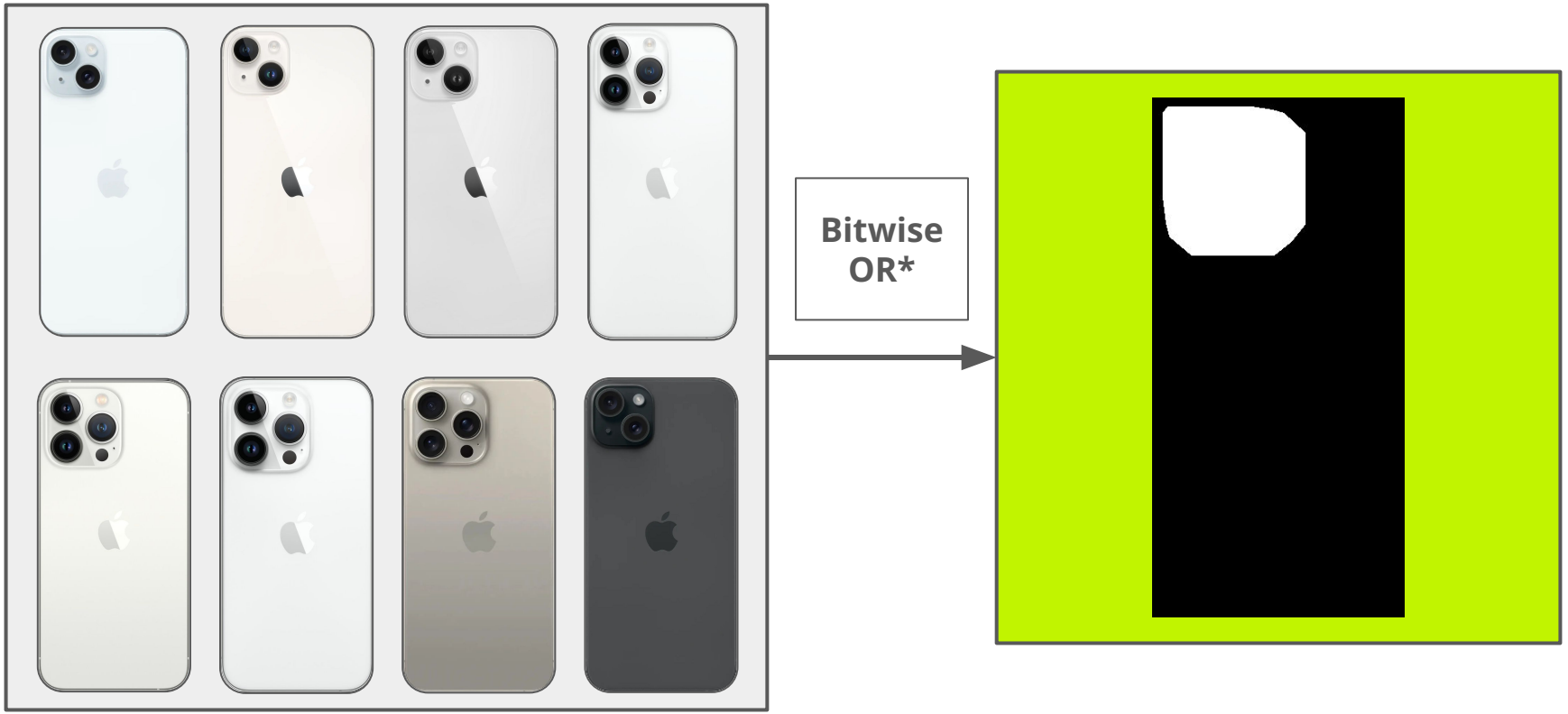
Classify Phone - IoU of Pred. and Truth

Bitwise OR*



* Camera masks of ALL Samsung smartphones

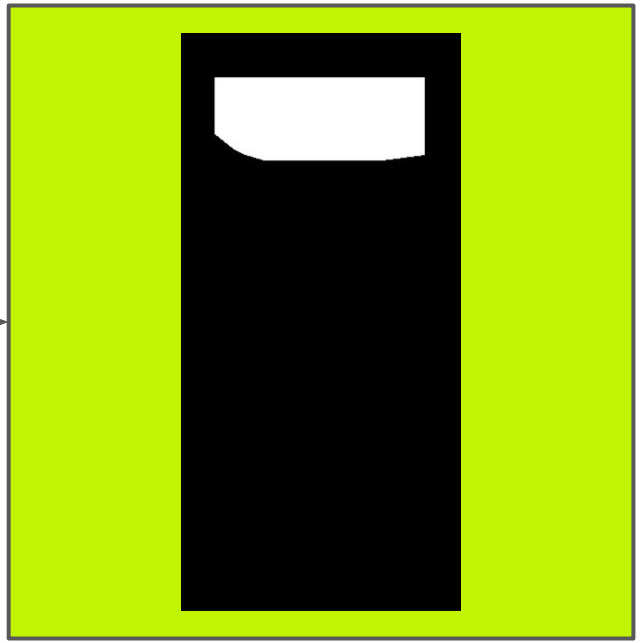
Classify Phone - IoU of Pred. and Truth



* Camera masks of ALL iPhone smartphones (after iPhone 12)

Classify Phone - IoU of Pred. and Truth

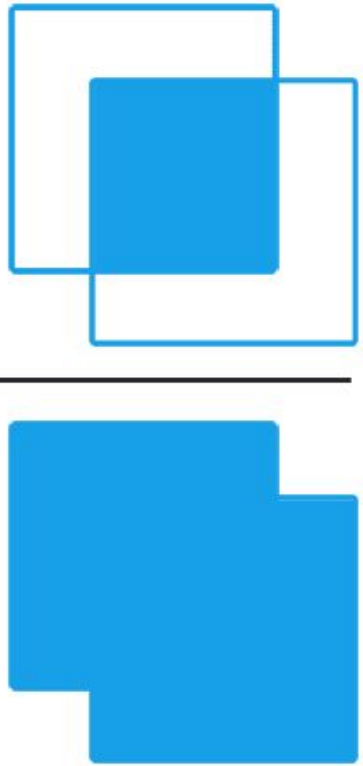
Bitwise OR*



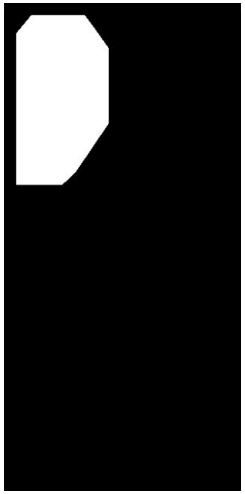
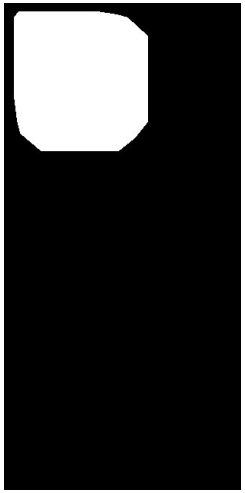
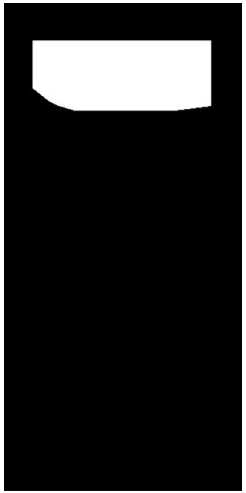
* Camera masks of ALL Pixel smartphones (after Pixel 6)

Classify Phone - IoU of Pred. and Truth

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



Classify Phone - IoU of Pred. and Truth

Corners	Warp Upright	Samsung?	iPhone?	Pixel?
				

Classify Phone - IoU of Pred. and Truth

* Bitwise OR of camera masks of ALL Samsung smartphones

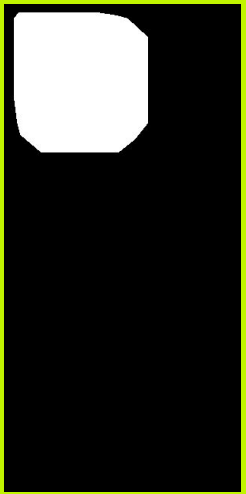
Classify Phone - IoU of Pred. and Truth

Rect. (0.744)

Poly. (0.747)

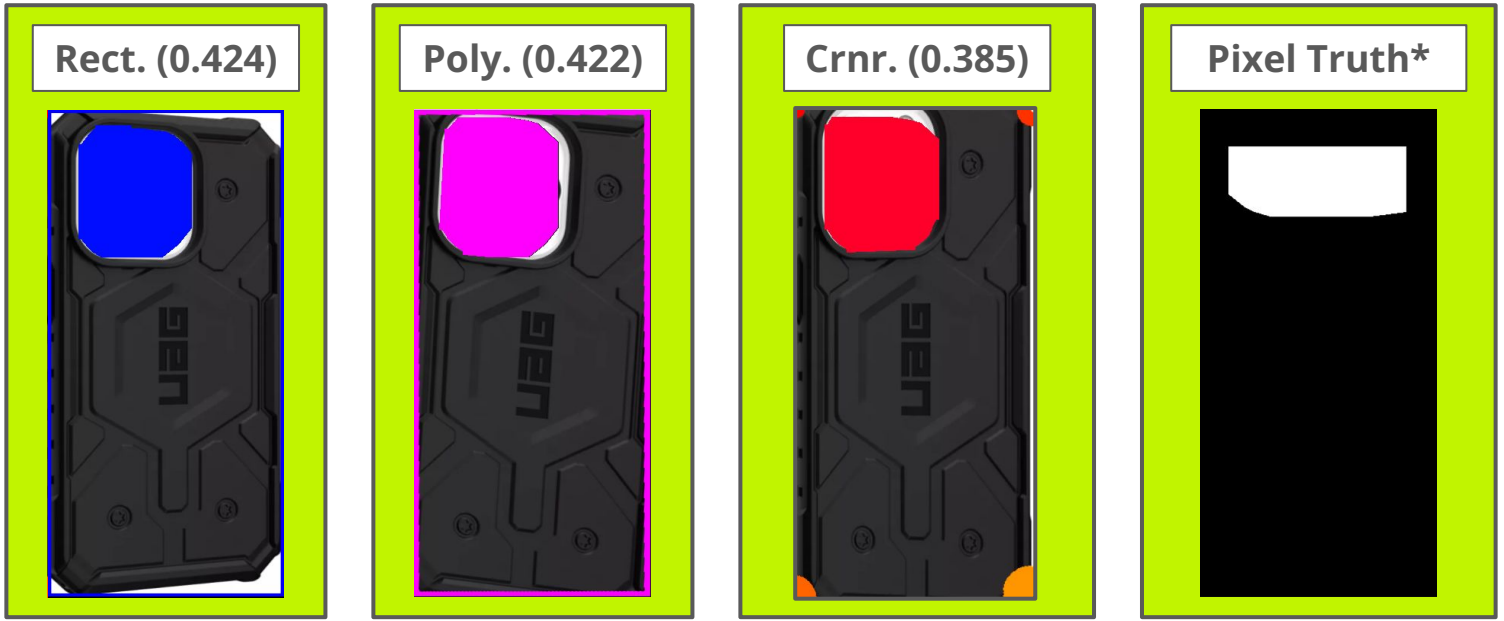
Crnr. (0.771)

iPhone Truth*



* Bitwise OR of camera masks of ALL iPhone smartphones (after iPhone 12)

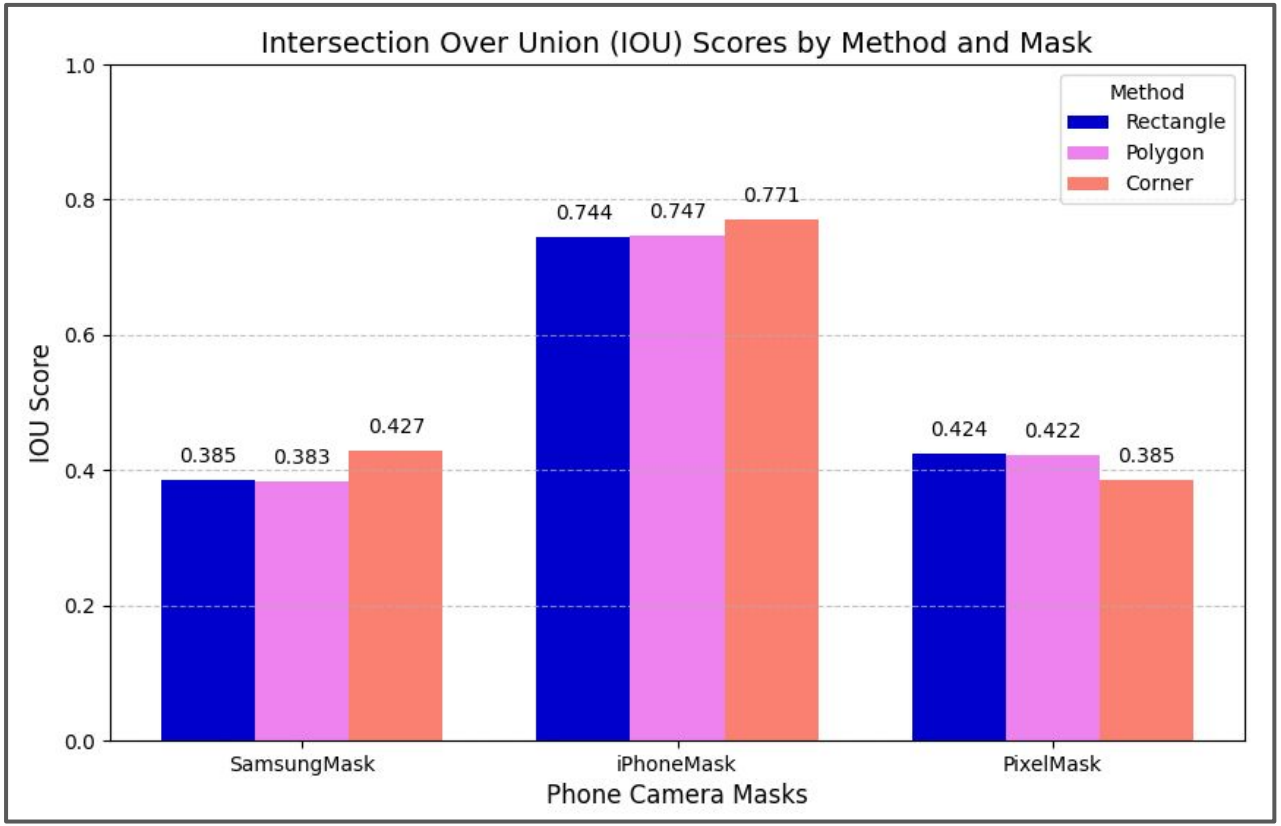
Classify Phone - IoU of Pred. and Truth



* Bitwise OR of camera masks of ALL Pixel smartphones (after Pixel 6)

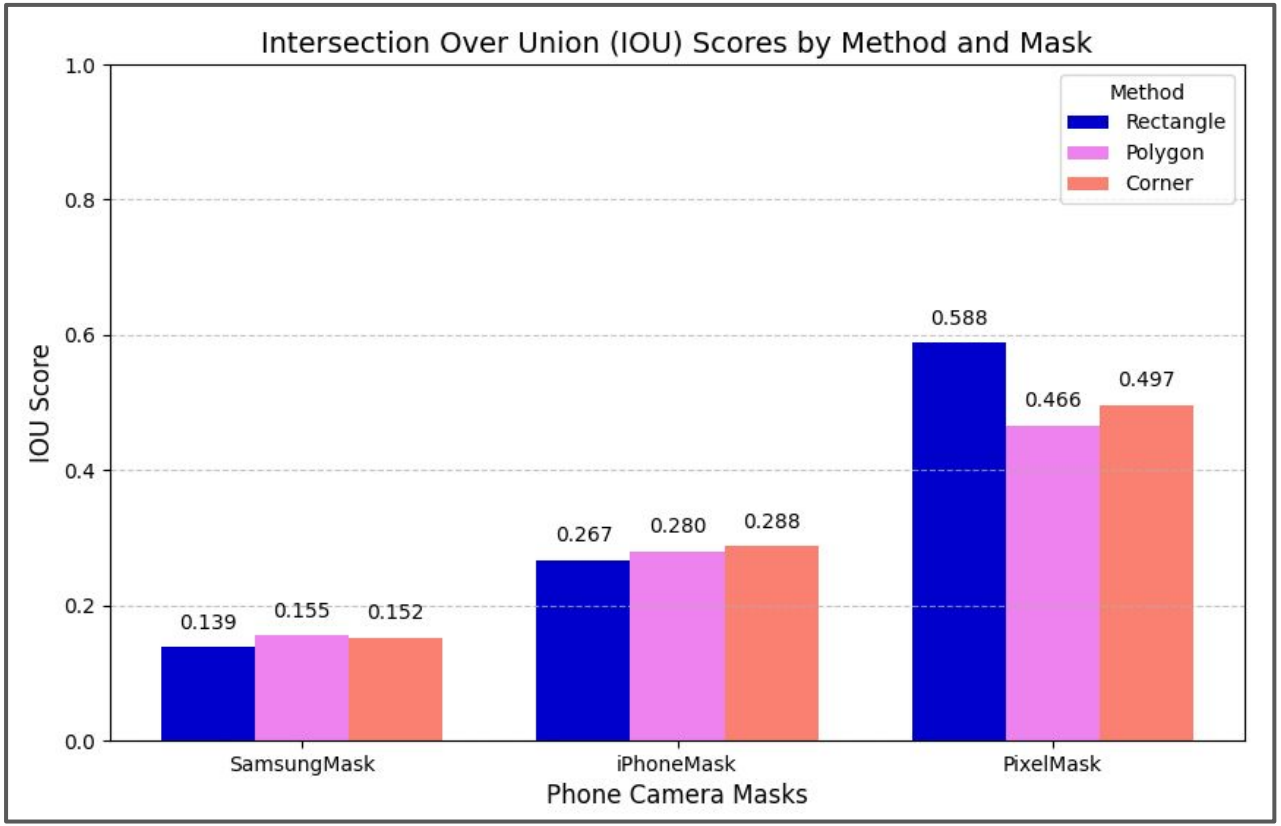
Classify Phone - IoU of Pred. and Truth

UAG Pathfinder
iPhone 14 Pro



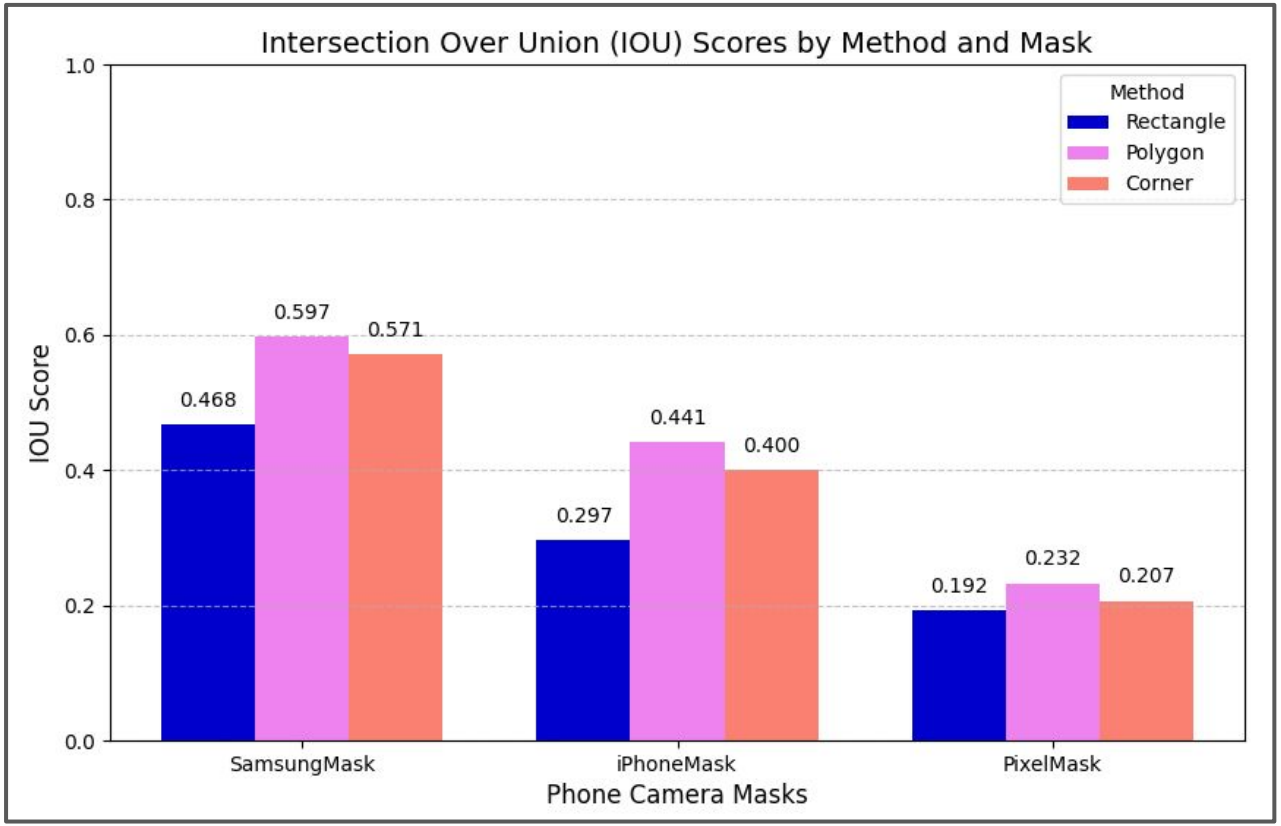
Classify Phone - IoU of Pred. and Truth

Otterbox Symmetry
Google Pixel 8



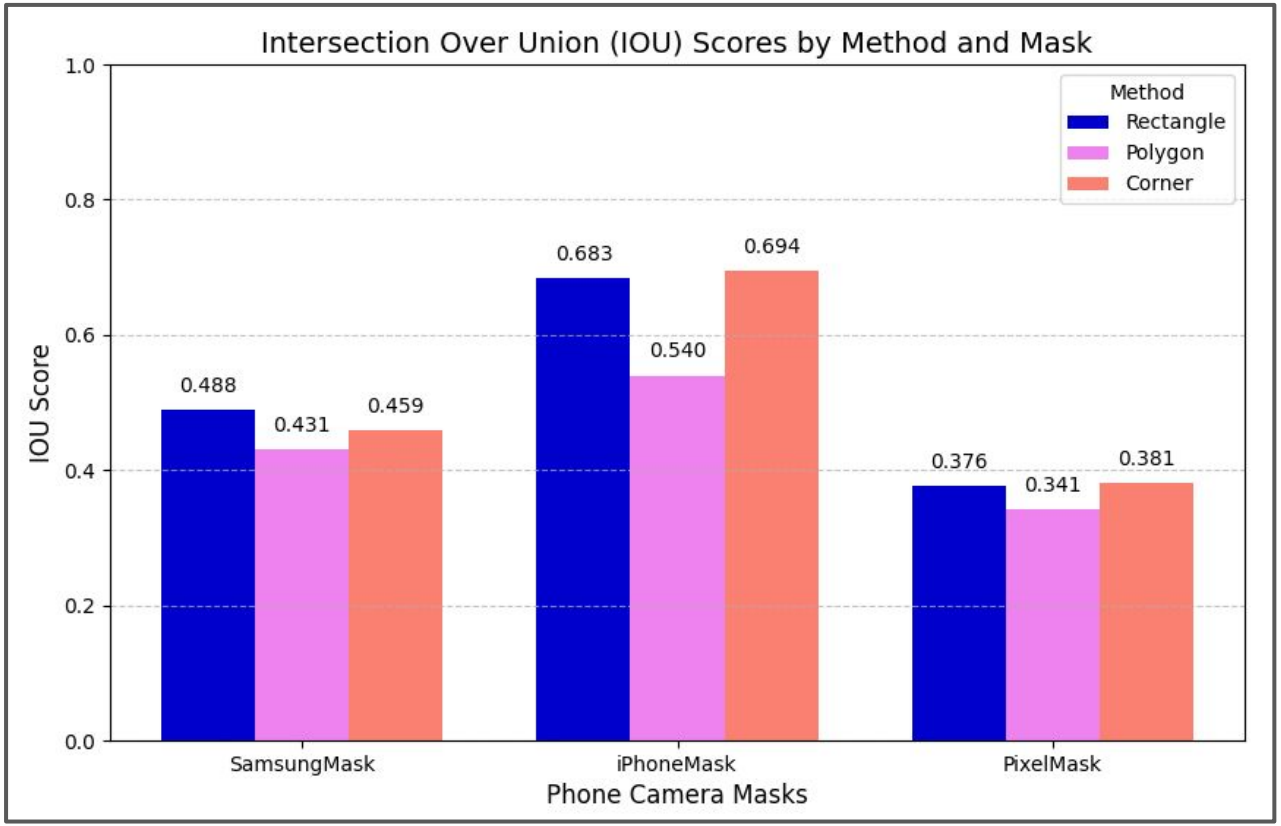
Classify Phone - IoU of Pred. and Truth

Incipio Cru
Galaxy S24



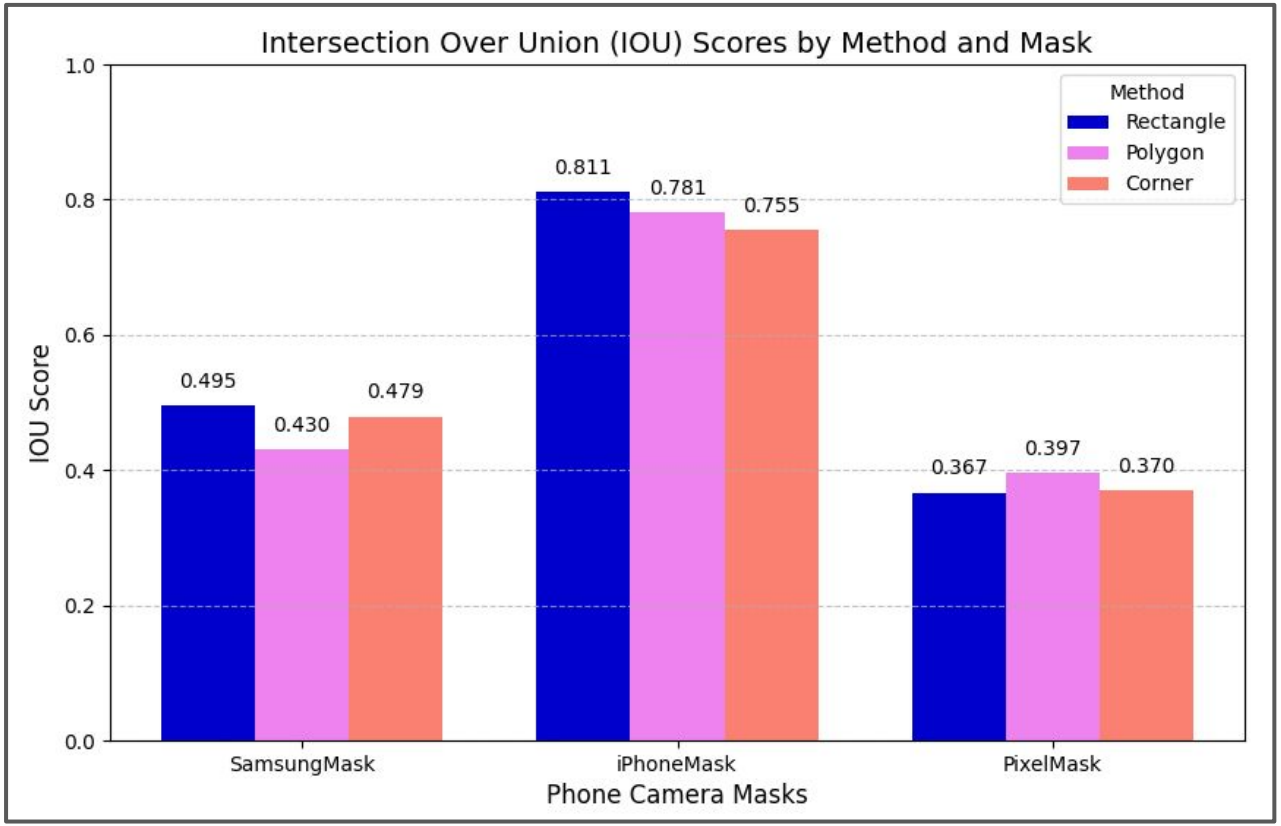
Classify Phone - IoU of Pred. and Truth

Afternoon in Versailles
iPhone 14 Pro Max



Classify Phone - IoU of Pred. and Truth

Gold Floral
iPhone 14 Pro



Classify Phone - Limitations

Rounded corners make it difficult to capture the **full area** of the phone

Input	Expected	Actual

Classify Phone - Limitations

Rounded corners make it difficult to capture the **full area** of the phone

Expected

Corners (green) go outside the phone contour (blue), which is **impossible** to obtain.

Actual

Corners (pink) stay attached to the phone contour, **cropping** the phone area.

Classify Phone - Limitations

Rounded corners make it difficult to capture the **full area** of the phone

Input	Expected	Actual
		

Classify Phone - Limitations

- Rounded corners make it difficult to capture full phone
- Truth masks can’t differentiate between specific models
  - Galaxy S23 vs Galaxy S24 OR iPhone 14 vs iPhone 15
  - Errors in camera segmentation may also prevent this
- Code is prone to errors due to incompatible image formats or different image dimensions (e.g. `findContours()` on a BGR image)
- Phone model classification isn’t up to standard with case design classification, in terms of accuracy

Conclusion & Future Improvements

Summary of Insights & Key Findings

- **Triple-Model Approach**: Three YOLO models for case design, phone area, and camera cutout has improved the precision of accessory recommendations.
- **High Accuracy in Similar Item Classification**: The model achieved high accuracy in distinguishing visually similar items, crucial for recommending compatible accessories.
- **Importance of Data Diversity**: A diverse dataset significantly improved model performance, highlighting the need for comprehensive data in visually similar product categories.
- **Real-Time Feasibility**: Preliminary results indicate that each model can meet low-latency demands in milliseconds, an important factor for real-time retail applications.
- **Enhanced Customer Experience**: The vision algorithm provides quick, relevant accessory suggestions, streamlining the shopping process and improving customer satisfaction.

Further Improvements and Goals

- **Experiment with Different Models:** Due to its robustness with overlapping masks, Mask R-CNN can turn both segmentation models into one.
- **Integrate a Real-Time Pipeline:** Optimize the model by using techniques such as **model quantization**, and leveraging hardware acceleration. Use **batch processing** where applicable to process multiple frames at once for speed improvement.
- **Set Up the Backend:** Connect the model to a web-based backend (REST-API or WebSocket) and build a database of the store inventory to match.
- **Closed Beta Launch**: Build the front-end deploy the app for beta-testing on a small user base.
- **User Feedback Analysis**: Analyze user feedback to implement improvements and finalize the application for open deployment.
- **Application Deployment**: Deploy the final application for public use.


Questions?

Verizon #1: Identifying Accessories in Retail with Real-Time Vision Algorithm

AI Studio Final Presentation

University of California, Los Angeles
December 4th, 2024