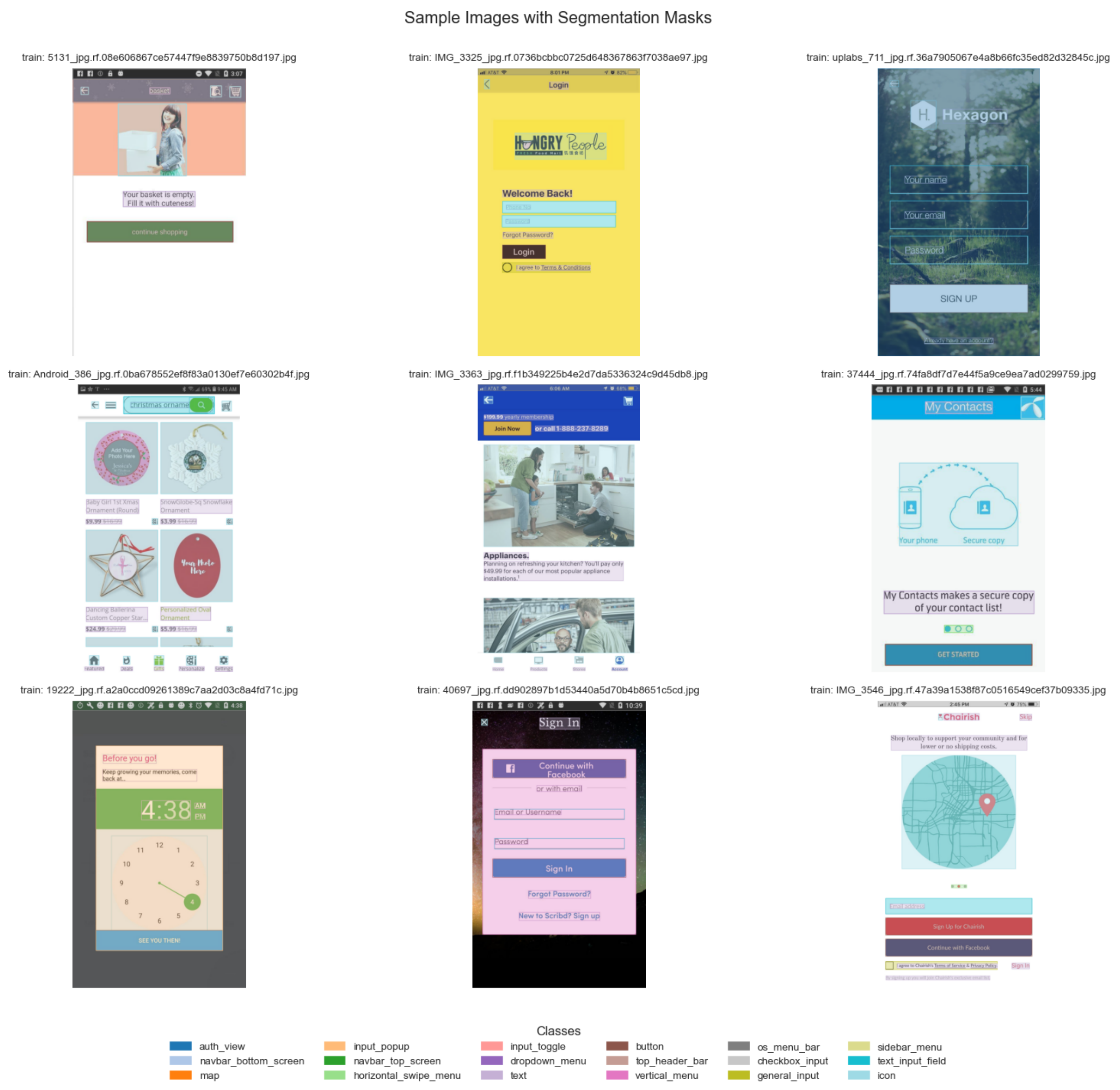
UI Element Segmentation and Classification Dataset Exploration

Matthew Favela

General Information of This Dataset

This dataset called `segmentation-UI Computer Vision Dataset` by Jonatan Fragoso focuses on UI element segmentation and classification that has 4794 images with 67,461 labeled instances. This dataset has a lot of mobile and web interface screenshots annotated with bounding boxes. Throughout the dataset, there are 18 unique labels/types of UI elements, ranging from common components like `text`, `icon`, and `button` to more specialized ones like `map` which is actual google map map views and `auth_view` which is a view that has options to sign in or sign up. This helps me generalize to a lot of different types of UI elements to look for specific types of screenshots and determine if input is required from the user or not. Next, this dataset was setup entirely allocated to a training split, which means that I would need to make custom validation and test splits to evaluate model bias and variance.


Sample Images with Segmentation Masks
train: 5131_jpg.rf.08e606867ce57447f9e8839750b8d197.jpg
train: IMG_3325_jpg.rf.0736bcbbc0725d648367863f7038ae97.jpg
train: uplabs_711_jpg.rf.36a7905067e4a8b66fc35ed82d32845c.jpg
train: Android_386_jpg.rf.0ba678552ef8f83a0130ef7e60302b4f.jpg
train: IMG_3363_jpg.rf.f1b349225b4e2d7da5336324c9d45db8.jpg
train: 37444_jpg.rf.74fa8df7d7e44f5a9ce9ea7ad0299759.jpg
train: 19222_jpg.rf.a2a0ccd09261389c7aa2d03c8a4fd71c.jpg
train: 40697_jpg.rf.dd902897b1d53440a5d70b4b8651c5cd.jpg
train: IMG_3546_jpg.rf.47a39a1538f87c0516549cef37b09335.jpg
Classes
auth_view
navbar_bottom_screen
map
input_popup
navbar_top_screen
horizontal_swipe_menu
input_toggle
dropdown_menu
text
button
top_header_bar
vertical_menu
os_menu_bar
checkbox_input
general_input
sidebar_menu
text_input_field
icon

Ethical Considerations

For ethical considerations, there are a few things that might have been unaccounted for by the dataset authors. First off, since these are UI screenshots, they probably capture a lot of personal information like usernames or private messages, which could leak sensitive data if they didn't scrub it properly. Next, this dataset has a lot of screenshots from proprietary apps, which means there are copyrighted designs and logos that could cause intellectual property issues if I used this commercially. Also, the screenshots collected might heavily skew towards Western, English-based apps or specific operating systems like iOS or Android, which means my model might perform poorly on UIs from other cultures. Lastly, a really accurate UI segmentation model could be weaponized by bad actors to possibly scrape proprietary app data at scale.

Exploratory Data Analysis

For the exploratory data analysis, I noticed a really big class imbalance where the most frequent class, `text`, shows up 31,121 times, while the least frequent one, `map`, only shows up 3 times. This means my model will heavily bias towards text and icons if I don't fix it. Looking at the image sizes, most images are standard mobile resolutions, but there are some crazy outliers up to 6,056 pixels tall, which means I'll probably need to crop or resize these really long screenshots when cleaning. I also looked at the instances per image and found a wide spread, but luckily all the bounding boxes are just simple 4-point polygons, which simplifies the task to object detection instead of complex pixel-perfect masking. Next, the EDA showed that there are no images currently in the validation or test sets, which means I'll need to make a data pipeline to randomly sample the data so all 18 classes are represented. Finally, when looking at some sample images with masks on them, I saw that UI elements are often nested, like text inside a button inside a navbar, which is definitely going to be kinda tricky to navigate.

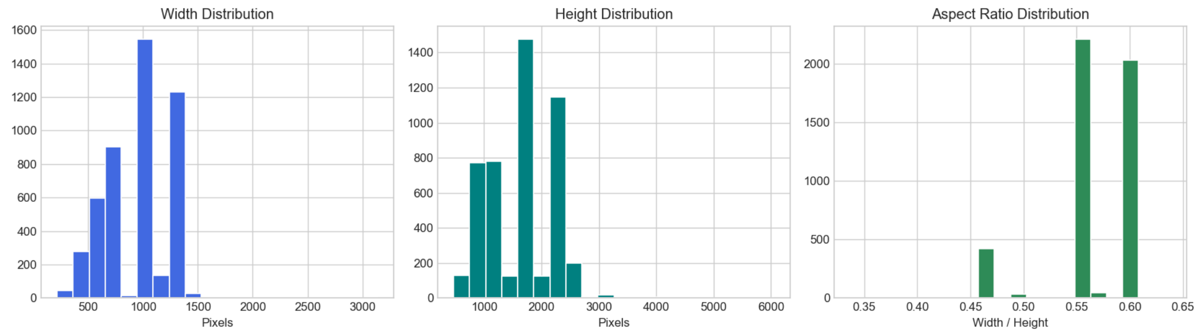


Figure 1: Geometry Statistics

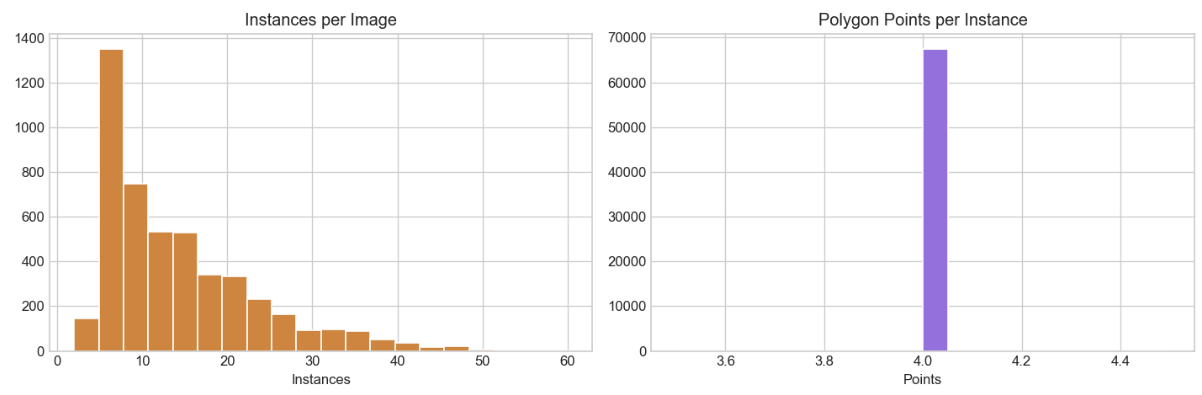


Figure 2: Complexity Statistics

Why Deep Learning is Needed for This Task

Deep learning is needed for this task because UI elements like buttons or icons don't have a single fixed look, they vary a lot in color, shape, size, and styling across different apps, which traditional computer vision techniques just can't handle. There's also a lot of contextual ambiguity, for example, a rectangle with text could be a button, a text input field, or just a general input depending on where it's placed, and deep learning models are great at capturing this spatial context to make accurate classifications. Also, UIs are highly structured with elements frequently overlapping or nested inside one another, and architectures like YOLO (with Roboflow) or Mask R-CNN are specifically designed to untangle and detect these overlapping bounding boxes independently. Next, with nearly 67,000 instances across thousands of images, hand-crafting rules for every single UI component is basically impossible, so deep learning lets the system automatically learn the underlying features of UI design directly from the data. The task will primarily be about assigning actual semantic meaning like identifying an `auth_view` or a `navbar_bottom_screen` to regions of pixels, which deep neural networks are perfect for, so that we can navigate through the UI and determine if input is required from the user or not.