Project Title:: Identifying and Organizing Household Objects Android App

Group Member Names: Jimmy Davis, Donna Quach

Summary of Data Sources: We plan on using a dataset that contains a list of household objects for testing. This will be represented as a set of images that will then be used by our model for object detection.

Description of Methods: There will be a few models we will need for this project. First, we will use a pre-trained model for object recognition by using Ultralytics YOLO. (This is to save us time and reduce the scope.) To determine which version of YOLO to use, we have performed a few tests as outlined in the "Results Generated So Far" section below comparing results and model latency. We also plan on using a model (classifier or clustering) to categorize our items.

Results Generated So Far: Fist, we must look at a few generations of the YOLO model. Based on the latency outputs as well as detection outputs for versions 8n, 9t, and 11n. The image used for this simple test is a stack of 6 colored books on a white background, with one open faced on top. Our results show both 11n and 9t detected books, though they were both incorrect in quantity, 8n detected nothing at all. This should be noted when considering latency, as 8n and 11n were effectively the same. As such, 11 will be the generation of choice for the project moving forward.

Second, YOLO 11 has been trained at a few different levels of complexity, 11n, s, m, l, and x, listed from small to large (x will be disreguarded as it shows diminishing returns in terms of accuracy, according to ultralytics site). The important considerations for this project are balancing accuracy and complexity, as the finished project will need to be light weight enough to run on mid-low end mobile devices or devices from previous generations of hardware. It's important to note that latency will be much higher in this application than as demonstrated by our tests running on modern desktop/server hardware. Compared to the first test, all three versions tested were able to accurately guess the objects in the image, and recognized 5 or 6 out of 6 books, compared to 0-2/6. the latency however is also much higher, and increases with model complexity.

There is no obvious answer, but 11s, m, and l show promising results.

```
!pip install ultralytics &>dev/null # In order for code below to work properly
```

```
/bin/bash: line 1: dev/null: No such file or directory
```

```
from ultralytics import YOLO
import time
import matplotlib.pyplot as plt

if __name__ == "__main__":

    path = "/home/book.jpg"
    models = ['yolov8n.pt', 'yolov9t.pt', 'yolo11n.pt']
    latencies = []

    ## Testing different model generations
    for model in models:
```

```
        model = YOLO(model)
        start = time.time()
        results = model.predict(path)
        latency = time.time() - start
        latencies.append(latency)

    # Create bar plot
    plt.figure(figsize=(10, 6))
    bars = plt.bar(models, latencies)

    # Add labels and title
    plt.xlabel('YOLO Versions', fontsize=12)
    plt.ylabel('Latency (seconds)', fontsize=12)
    plt.title('YOLO Inference Latency Comparison', fontsize=14,
pad=20)

    # Add value labels on top of bars
    for bar in bars:
        height = bar.get_height()
        plt.text(bar.get_x() + bar.get_width()/2., height,
                 f'{height:.2f}s',
                 ha='center', va='bottom')

    plt.ylim(0, max(latencies) * 1.2)  # Add some headroom
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.tight_layout()
    plt.show()

    ## Testing different levels of model complexity
    models = ['yolo11s.pt', 'yolo11m.pt', 'yolo11l.pt']
    latencies = []

    for model in models:
        model = YOLO(model)
        start = time.time()
        results = model.predict(path)
        latency = time.time() - start
        latencies.append(latency)

    # Create bar plot
    plt.figure(figsize=(10, 6))
    bars = plt.bar(models, latencies)

    # Add labels and title
    plt.xlabel('YOLO Versions', fontsize=12)
    plt.ylabel('Latency (seconds)', fontsize=12)
    plt.title('YOLO Inference Latency Comparison', fontsize=14,
pad=20)

    # Add value labels on top of bars
```

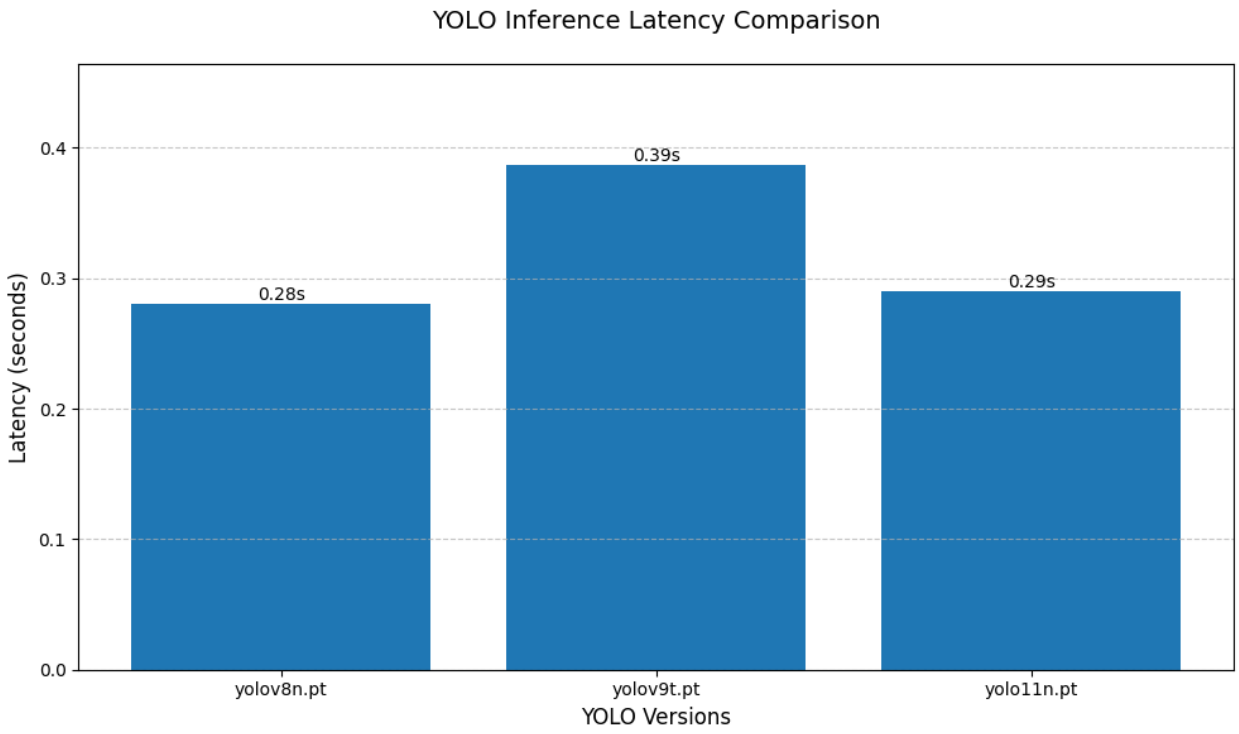
```
    for bar in bars:
        height = bar.get_height()
        plt.text(bar.get_x() + bar.get_width()/2., height,
                 f'{height:.2f}s',
                 ha='center', va='bottom')

    plt.ylim(0, max(latencies) * 1.2)  # Add some headroom
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.tight_layout()
    plt.show()
```

```
image 1/1 /home/book.jpg: 448x640 (no detections), 176.6ms
Speed: 3.5ms preprocess, 176.6ms inference, 1.0ms postprocess per
image at shape (1, 3, 448, 640)

image 1/1 /home/book.jpg: 448x640 3 books, 217.8ms
Speed: 3.6ms preprocess, 217.8ms inference, 1.4ms postprocess per
image at shape (1, 3, 448, 640)

image 1/1 /home/book.jpg: 448x640 1 book, 166.3ms
Speed: 3.3ms preprocess, 166.3ms inference, 1.7ms postprocess per
image at shape (1, 3, 448, 640)
```

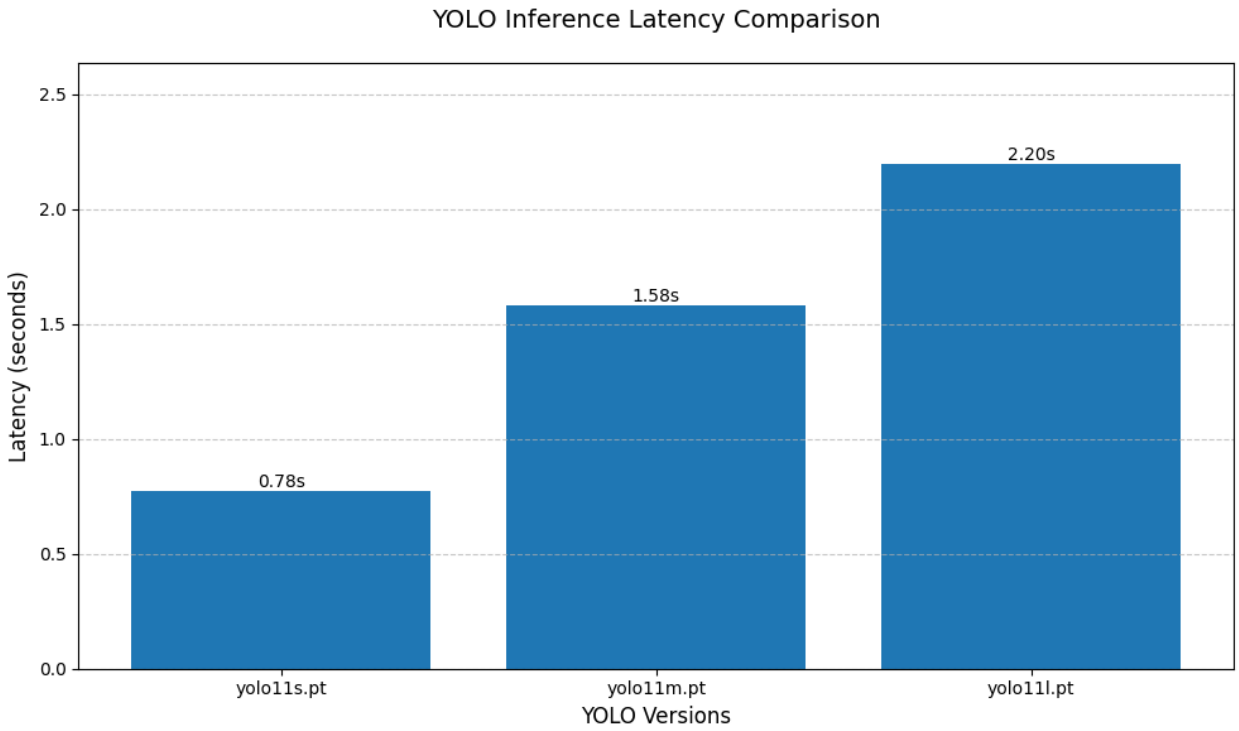



```
image 1/1 /home/book.jpg: 448x640 6 books, 465.8ms
Speed: 3.8ms preprocess, 465.8ms inference, 1.8ms postprocess per
```

```
image at shape (1, 3, 448, 640)

image 1/1 /home/book.jpg: 448x640 5 books, 1021.3ms
Speed: 3.6ms preprocess, 1021.3ms inference, 1.5ms postprocess per
image at shape (1, 3, 448, 640)

image 1/1 /home/book.jpg: 448x640 6 books, 1459.5ms
Speed: 3.4ms preprocess, 1459.5ms inference, 2.0ms postprocess per
image at shape (1, 3, 448, 640)
```



Anticipated Difficulties or Road Blocks: The only road blocks we anticipate for this project are making sure that all pieces of this project run cohesively together (i.e. it all functions) and running against time.