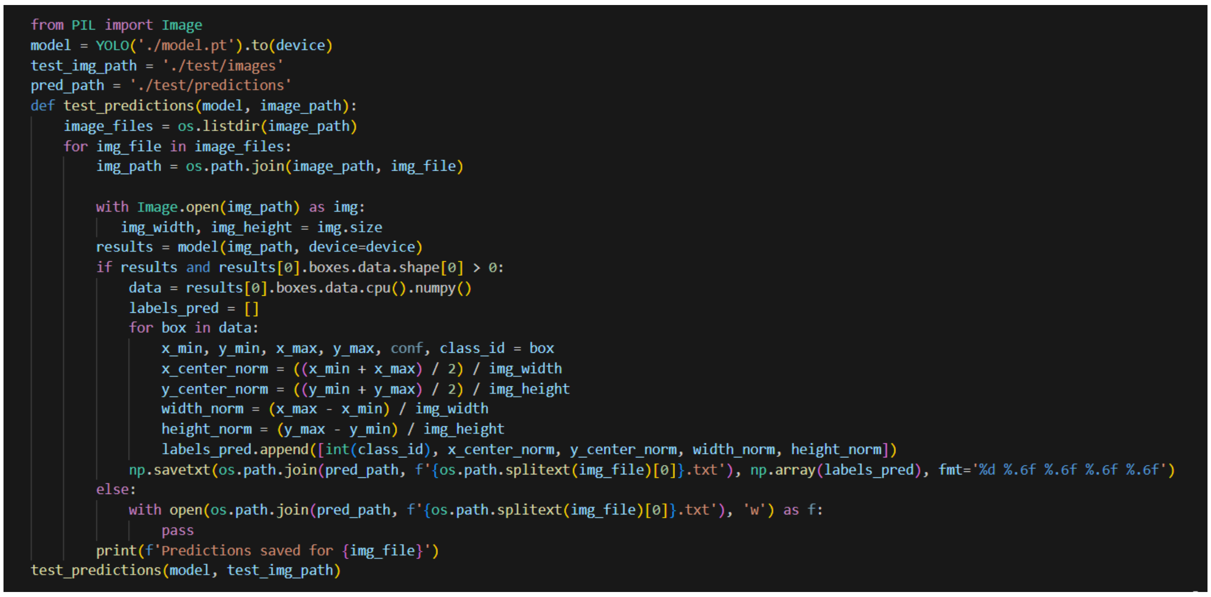
**Approach**: First, we understood the given dataset and labels for the images. For model building, we chose the pretrained YOLO model from ultralytics as it was difficult to train a CNN model from scratch and also the format of dataset matches with the format in which YOLO accepts the input. We used the YOLOv8 pre trained model and trained it on our dataset. We increased the batch size while training instead of the default YOLO batch size. Also, we used image size as 320x320 in the parameter while training the model as images in training were 416x416 and 640x640 and this image size was sweet spot for decreasing training time and have a standard size for all images. We trained first with 50 epochs with batch size 64. For more fine tuning, we took the saved model from first training and trained it for more 20 epochs and 320x320 image size with default batch size and other parameters. We used Gemini AI for getting the YOLO library code as we weren't familiar with the library before.

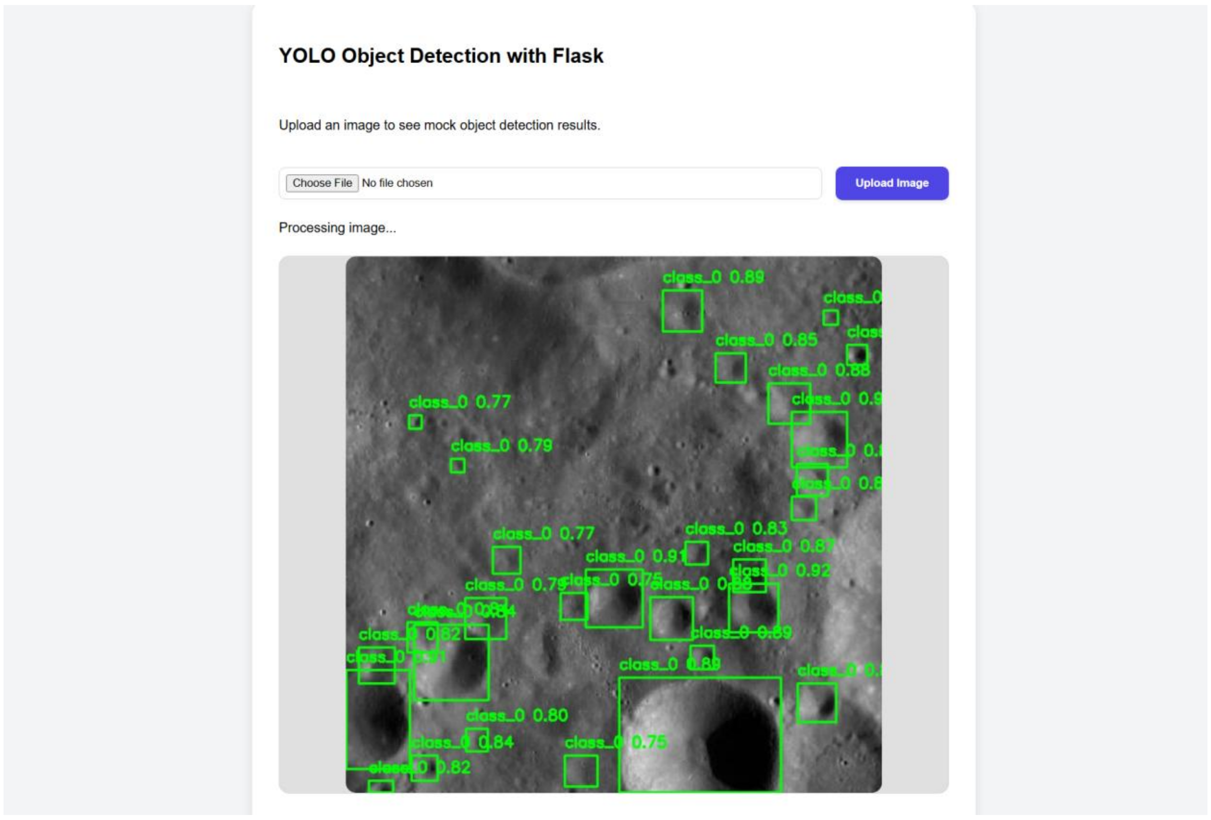
**Implementation**: We also implemented a UI where we could upload a lunar image and get the annotated image as an output. We used Flask python library to implement the python server and designed the HTML webpage. We used Gemini AI to generate a template for the same and tweaked it as per the requirement of our project. For the prediction of labels of test images, we defined a function which could save the predictions for class, and normalized coordinates for x_centre, y_centre , width and height.



```python
from PIL import Image
model = YOLO('./model.pt').to(device)
test_img_path = './test/images'
pred_path = './test/predictions'
def test_predictions(model, image_path):
    image_files = os.listdir(image_path)
    for img_file in image_files:
        img_path = os.path.join(image_path, img_file)

        with Image.open(img_path) as img:
            img_width, img_height = img.size
        results = model(img_path, device=device)
        if results and results[0].boxes.data.shape[0] > 0:
            data = results[0].boxes.data.cpu().numpy()
            labels_pred = []
            for box in data:
                x_min, y_min, x_max, y_max, conf, class_id = box
                x_center_norm = ((x_min + x_max) / 2) / img_width
                y_center_norm = ((y_min + y_max) / 2) / img_height
                width_norm = (x_max - x_min) / img_width
                height_norm = (y_max - y_min) / img_height
                labels_pred.append([int(class_id), x_center_norm, y_center_norm, width_norm, height_norm])
            np.savetxt(os.path.join(pred_path, f'{os.path.splitext(img_file)[0]}.txt'), np.array(labels_pred), fmt='%d %.6f %.6f %.6f %.6f')
        else:
            with open(os.path.join(pred_path, f'{os.path.splitext(img_file)[0]}.txt'), 'w') as f:
                pass
        print(f'Predictions saved for {img_file}')
test_predictions(model, test_img_path)
```

UI:


## YOLO Object Detection with Flask

Upload an image to see mock object detection results.

Choose File No file chosen | Upload Image

Processing image...

The real world use cases that we could think about is using it on a moon lander which could detect craters in real time before landing which will help it in finding a safe spot for landing.