OGMEN ROBOTICS COMPUTER VISION INTERNSHIP TASK 1

NAME: Khush Jitendra Attarde

This particular problem or task consists of approach for detecting dog face and performing pose estimation on it using SSD.

Dataset Collection and Annotation

Dataset Collection:

Purpose: Gathered images that will be used to train the models.

Sources: Images are be collected from various sources such as online search engine.

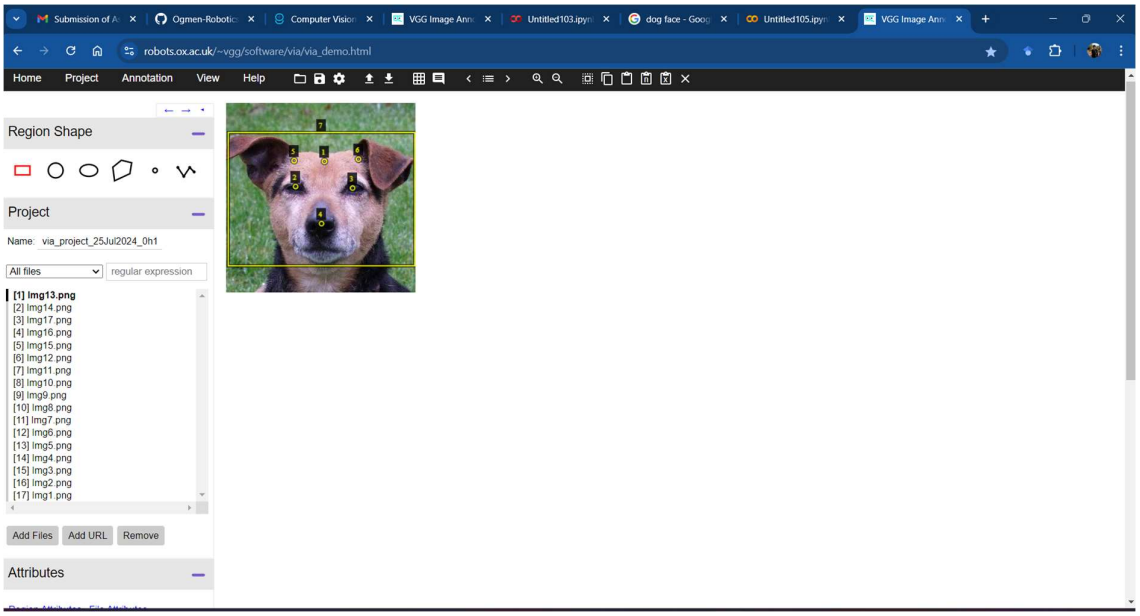
Format: Images are in PNG format.

Annotation Using VIA:

Tool: Visual Image Annotation (VIA) is used for annotating images with regions of interest (ROI) like bounding boxes, polygons, or keypoints.

Process:

- **Upload Images:** Loaded the collected images into the VIA tool.
- **Annotate:** Draw bounding boxes around objects (in this case, dog faces) as well as keypoints and specify attributes or labels.
- **Save Annotations:** Save the annotations in JSON and CSV format.



Conversion to XML:

Objective: Convert annotations from JSON to XML format, which is commonly used in many object detection frameworks.

Conversion Process:

- **Read CSV:** Extract filenames and annotation data from a CSV file, assuming a structured format where annotations are in JSON fields.
- **Parse JSON:** Convert JSON fields to Python dictionaries.
- **Generate XML:** Create an XML file for each image containing object details, such as bounding boxes and labels.

Output Directory: The converted XML files are saved in a specified directory.

Custom Dataset Creation for PyTorch

Creating a Custom Dataset:

Class Definition: 'YoloDataset' is defined to work with the PyTorch framework. It handles image loading, annotation parsing, and dataset creation.

Initialization:

- **Directories:** Specify paths for images and XML annotations.
- **Label Map:** Define a mapping from class names to integer labels.

Data Loading:

- **Read Images:** Load images from the specified directory.

- **Parse Annotations:** Load corresponding XML files and extract bounding box coordinates and labels.
- **Convert to Tensors:** Convert the annotations to PyTorch tensors.

Transformations: Apply transformations if needed (e.g., resizing, normalization).

DataLoader:

Purpose: Handles batching and shuffling of the dataset during training.

Batch Processing: Define a custom collate function to handle batches of images and annotations.

Object Detection and Pose Estimation

Object Detection:

Model: Use a pre-trained SSD (Single Shot MultiBox Detector) model for detecting objects in images.

Process:

- **Load Model:** Load the pre-trained SSD model.
- **Perform Detection:** Run object detection on input images to get bounding boxes, labels, and scores.

The SSD model, specifically ssdlite320_mobilenet_v3_large, is utilized for detecting dogs within images:

- **Model Loading and Setup**: The pre-trained model is loaded and set to evaluation mode to perform inference on input images.
- **Detection Execution**: The model processes the image tensor and outputs bounding boxes, labels, and confidence scores. Post-processing includes filtering out low-confidence detections and non-dog objects.

Pose Estimation:

Model: Use a pre-trained Pose Estimation model (Keypoint R-CNN) to detect keypoints on the detected objects.

Process:

- **Load Model:** Load the pre-trained Pose Estimation model.
- **Perform Estimation:** Run pose estimation on cropped regions of interest (e.g., dog faces) to get keypoints.

The Pose Estimation model (keypointrcnn_resnet50_fpn) is employed to identify keypoints on detected dogs:

- **Model Loading and Setup**: Similar to the object detection phase, the pose estimation model is loaded and set to evaluation mode.
- **Pose Estimation Execution**: The model analyzes cropped regions (i.e., detected dogs) and predicts keypoints for features like the nose, eyes, and ears.
- **Keypoint Processing**: Post-processing steps involve clustering keypoints using algorithms like DBSCAN to group similar keypoints and reduce noise, thereby improving the accuracy of pose estimation.

Visualization:

Objective: Visualize detection results with bounding boxes and pose keypoints.

Process:

- **Load Image:** Load the image and apply necessary transformations.
- **Detection Results:** Draw bounding boxes for detected objects and visualize keypoints.
- **Cluster Keypoints:** Optionally cluster keypoints if needed to reduce noise and improve visualization.

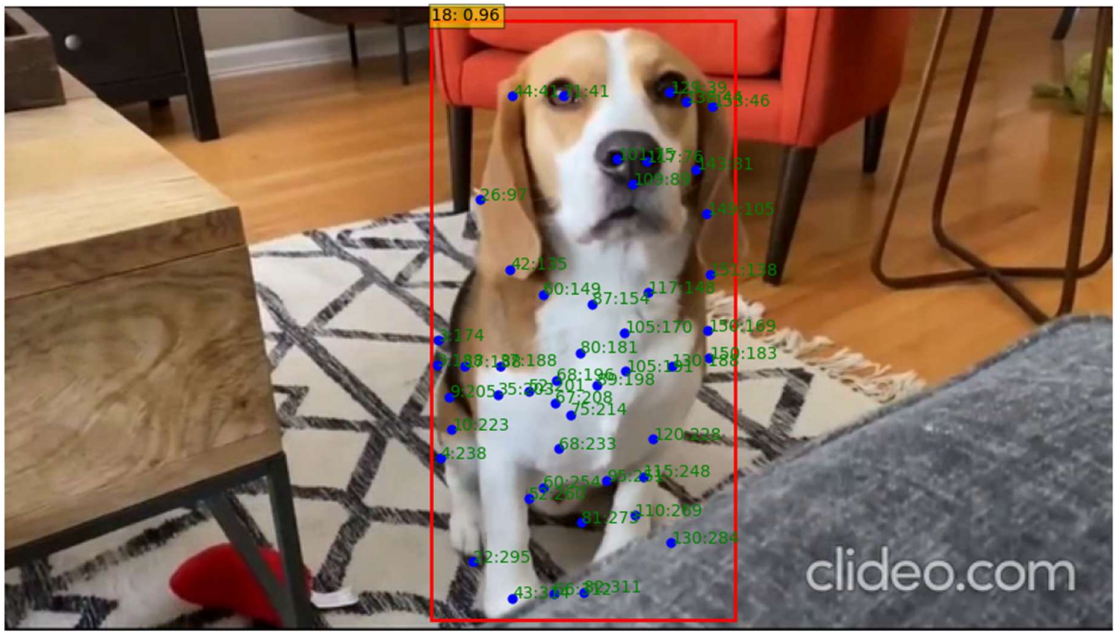
The visualize_inference function integrates and displays results from both object detection and pose estimation:

- **Visualization Techniques**: Bounding boxes are drawn around detected dogs, and keypoints are plotted on these regions. Use distinctive colors and markers for different keypoints to enhance clarity.

Results and Analysis

The pipeline successfully detects dogs in images and performs accurate pose estimation:

- **Detection Performance**: The SSD model provides robust detection with high accuracy. Bounding boxes clearly outline the detected dog regions but not face only but nearby parts of dog too.
- **Pose Estimation Accuracy**: The pose estimation model accurately identifies keypoints on the dog's face. Keypoints are correctly clustered, and visualizations are clear and informative. But also detect other areas too which is a problem.



Limitation was it was trained on just 16 images and it should be increased further and it was detecting dog areas of body too rather than just dog face.