1. Detector architecture analysis Compare YOLOv8, RT-DETR, and Mask R-CNN networks – look up the papers on arXiv/github implementation. Have a look at the following: - - - number of parameters inference time on a single image how are the architectures of these networks different? Write a short summary (1-2 paragraphs) or a simple table in your own words, focusing on the basics. Use online resources or lecture notes to guide you. Submit a report.

| **Model** | **# Parameters (Approx.)** | **Inference Time (Single Image)** | **Architecture Highlights** |
| --- | --- | --- | --- |
| **YOLOv8** | ~11M (YOLOv8n) to ~68M (YOLOv8x) | ~5-10 ms (YOLOv8n on GPU) | Single-stage, anchor-free, lightweight. Combines CSPDarknet backbone, PAN neck, and detection head. Designed for real-time performance. |
| **RT-DETR** | ~50M (RT-DETR-R50) | ~20-30 ms (GPU) | Two-stage-like transformer-based detector. Uses ResNet or Swin backbone, learnable object queries, and a lightweight decoder for faster DETR-style detection. |
| **Mask R-CNN** | ~44M (with ResNet-50 backbone) | ~80-150 ms (GPU) | Two-stage detector. First generates region proposals (RPN), then classifies and segments. Accurate, but slower. |

YOLOv8 is a fast, single-stage detector optimized for real-time inference. It uses a streamlined architecture with a compact backbone and head, making it highly efficient on both edge devices and GPUs. RT-DETR (Real-Time DEtection TRansformer) adapts the DETR architecture with reduced complexity and latency, using object queries and transformer decoders to directly predict object positions and labels, offering a trade-off between speed and accuracy. In contrast, Mask R-CNN is a powerful two-stage model that first proposes regions of interest and then refines them with classification and mask prediction heads. While it offers superior segmentation performance, it's significantly slower and heavier than the other two.

1. Upload your images to Colab, run the detection, and save the outputs (showing boxes/labels). Write a brief commentary (1-2 paragraphs): Did it detect objects correctly, or were there errors? How did lighting, angles, or clutter affect it?

Using the pretrained Mask R-CNN model from Detectron2, I performed object detection on 21 custom images taken with my phone under various real-world conditions. These included images of the same cat from different angles and distances, under varying lighting conditions (natural light, low light, shadows), and in both sparse and cluttered environments.

Overall, the model performed well in identifying my cat. It consistently detected my cat in most of the pictures, especially in well-lit, uncluttered scenes. It was a little harder in low-light scenarios or when objects were partially occluded or angled steeply, detection performance slightly dropped— the labels were sometimes more ambiguous, and bounding boxes/masks less precise. Cluttered scenes with many overlapping objects also presented a challenge, where the model misclassified parts of my cat, or wasn’t fully sure in some of her body parts. Also in some specific occasions (maybe depending on the posing) the model was considering my cat to be a dog) Despite these small limitations, the model demonstrated strong generalization without fine-tuning, showcasing the effectiveness of Detectron2's pretrained weights for real-world applications.