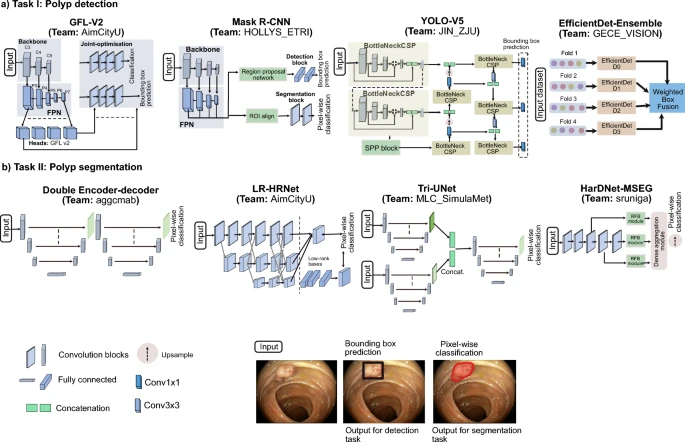
**Baseline Model Architecture Overview**

To the best of our knowledge, there is no non-deep-learning baseline model for object detection. In contrast, the typical way of getting a baseline model is to train a widely used, pre-trained model on our own dataset. Since most of our data come from [a past competition](https://www.nature.com/articles/s41598-024-52063-x) (in 2021) , we plan to choose one or two models from the top four teams to be our baseline model. The following plot lists their architectures:



Among these four models, **YOLO** (You Only Look Once) is the most popular model in object detection, with key features being **fast** and **easy to train**. And hence YOLO is good at real-time inference. Mask R-CNN is an improved version of the **Faster R-CNN**. The latter is reported to be very **precise** and **stable.** However it has the problem of being hard to train and slow when used in real-time. **GFL** (Generalized Focal Loss)and **EfficientDet** are less popular right now. GFL is an improved version of the RetinaNet, which solves the **class imbalance problem** for anchor-based models. However, modern models are often anchor-free for computational efficiency. **EfficientDet** is built to optimize efficiency while maintaining the same level of accuracy. Consequently, it has a backbone that is hard to modify compared with ResNet. Hence, we do not consider EfficientDet as our baseline model.

**One-stage vs Two-stage**

All the models we plan to explore have a CNN backbone. The main differences are in the paradigm. Faster R-CNN is a two stage detector while YOLO is a one-stage detector. A transformer detector would work from end-to-end via attention. As for the pipeline, Faster R-CNN follows a three step approach while YOLO directly predicts boxes and classes from feature maps. One of the drawbacks of the transformer approach is that it is harder to train and it requires much more data than the Faster R-CNN model. **Since we want the baseline model to be easy to train and have the ability for real time computations**, we will choose one of the YOLO models instead of R-CNN as our baseline.

**Anchor vs Anchor-free**

**Anchors** are a set of predefined bounding boxes, which turns an undefined regression problem (finding the coordinates of an unknown number of bounding boxes) into a regression problem (finding pre-defined boxes that are close to ground truth). Anchors are foundational in many frameworks (like Faster R-CNN, RetinaNet, YOLOv2–v4). But in recent years, **anchor-free methods** (like YOLOv8+, DETR) have gained traction because they simplify the design and often perform just as well or better. We will try both anchor (YOLOv5) and anchor-free (YOLOv8) models. If they turn out to have similar performance, then we will choose the anchor-based model as our baseline since it is more classic.

**A detailed description on Yolo and Fast R-CNN:**

* **YOLO:** This model has a three-part design consisting of backbone, neck and head.The backbone is responsible for feature extraction, the neck does a multi-scale fusion to preserve object signals, and the head makes predictions using decoupled branches – one for classification and one for box regression.Lastly, this model is anchor free meaning it predicts boxes directly rather than using predefined anchor boxes. The loss function is the weighted sum of box loss, distribution focal loss and class loss. Because we are only doing polyp detection, the class weight is small.
* **Faster R-CNN:** This model, created in 2015, consists of four main components: backbone, region proposal network (RPN), ROI Pooling/ROI Align, and detection head. The backbone extracts features and outputs a high level feature map so that the RPN can slide small convolutional windows across it to predict anchor boxes. Next, we have ROI pooling which crops the region from the feature map and pools it into a fixed size. Lastly, the detection head has two parallel outputs which are softmax classification and bounding box regression. The loss is the combination of both the softmax and the regression. For the baseline model of this architecture, we plan to freeze the backbone and neck layers and train the head. Because of the heaviness of the model, it is only possible to do 10 epochs.