NC STATE UNIVERSITY

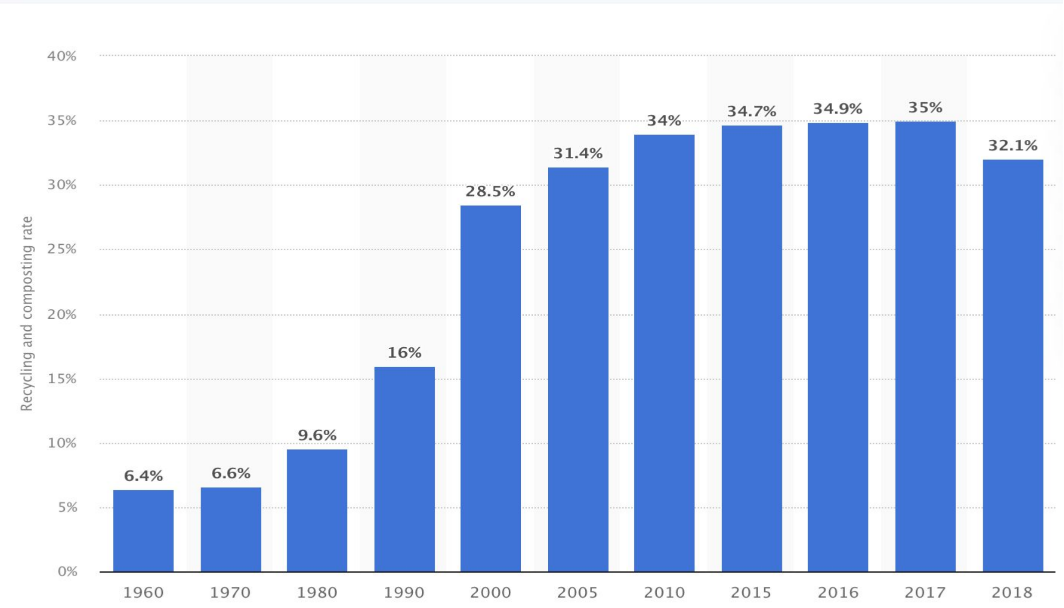
Team 66

Waste Detection using Faster RCNN and Mask R-CNN

Vignesh Muthukumar Sunidhi Hegde Divya Giridhar

Motivation

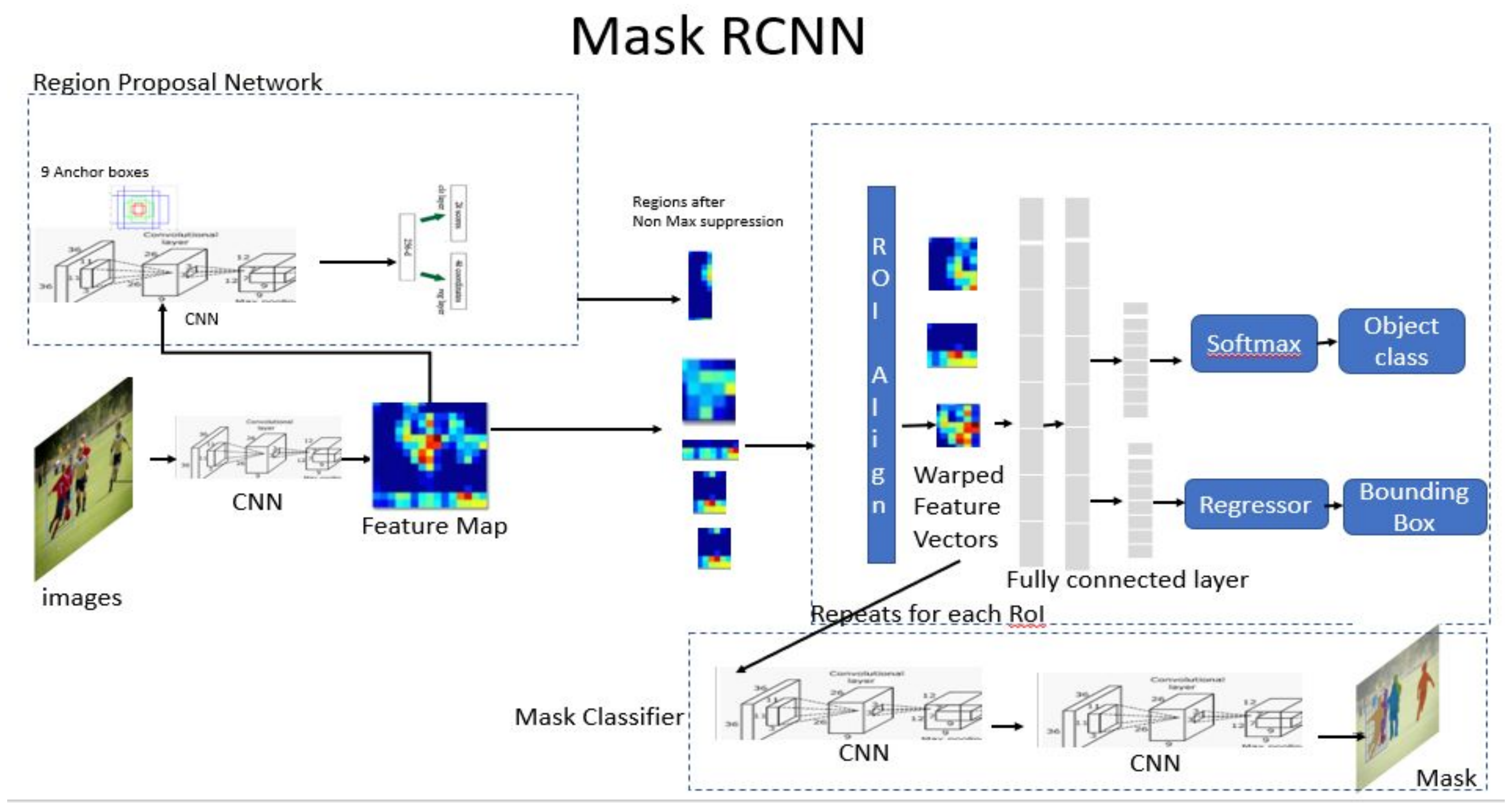
- Only 30%* of recyclable materials actually get recycled.
- Automated system needed to improve efficiency and create a sustainable process to manage waste.
- Instance segmentation is challenging as it requires the correct detection of all objects in an image while also precisely segmenting each instance.
- We aim to mitigate this issue by developing a model that automatically detects different types of waste products into predefined classes.



*** source** - Recycling rate of municipal solid waste in the United States 1960-2018. Published by Ian Tiseo, Mar 30, 2022

Task

- Implement **Faster R-CNN** model.
- Implement **Mask R-CNN** for image segmentation to detect the type of waste.
- Measured using **Region of Interest** (RoI).



* **source** - Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girshick, 'Mask R-CNN' https://arxiv.org/pdf/1703.06870.pdf

NC STATE UNIVERSITY

Dataset

- The primary dataset used to train the model is inspired from the resources curated by Jay et.al *[1].
- The data for other classes like Medical waste was scraped from Google Images using the Simple Image Downloader library *[2].
- The images are broadly classified into 6 classes such as **Metal**, **Glass**, **Paper**, **Organic**, **E-Waste** and **Medical**.
- We have: **856** examples, **685** are training and **171** testing. (80-20 split)

*** sources** - [1] https://drive.google.com/drive/folders/1CTvTgnTvwlcKwJ8yz4jUOs0JYTKrplA
- [2] https://libraries.io/pypi/simple-image-download

NC STATE UNIVERSITY

Sample Images

Glass

Medical

Organic

Metal

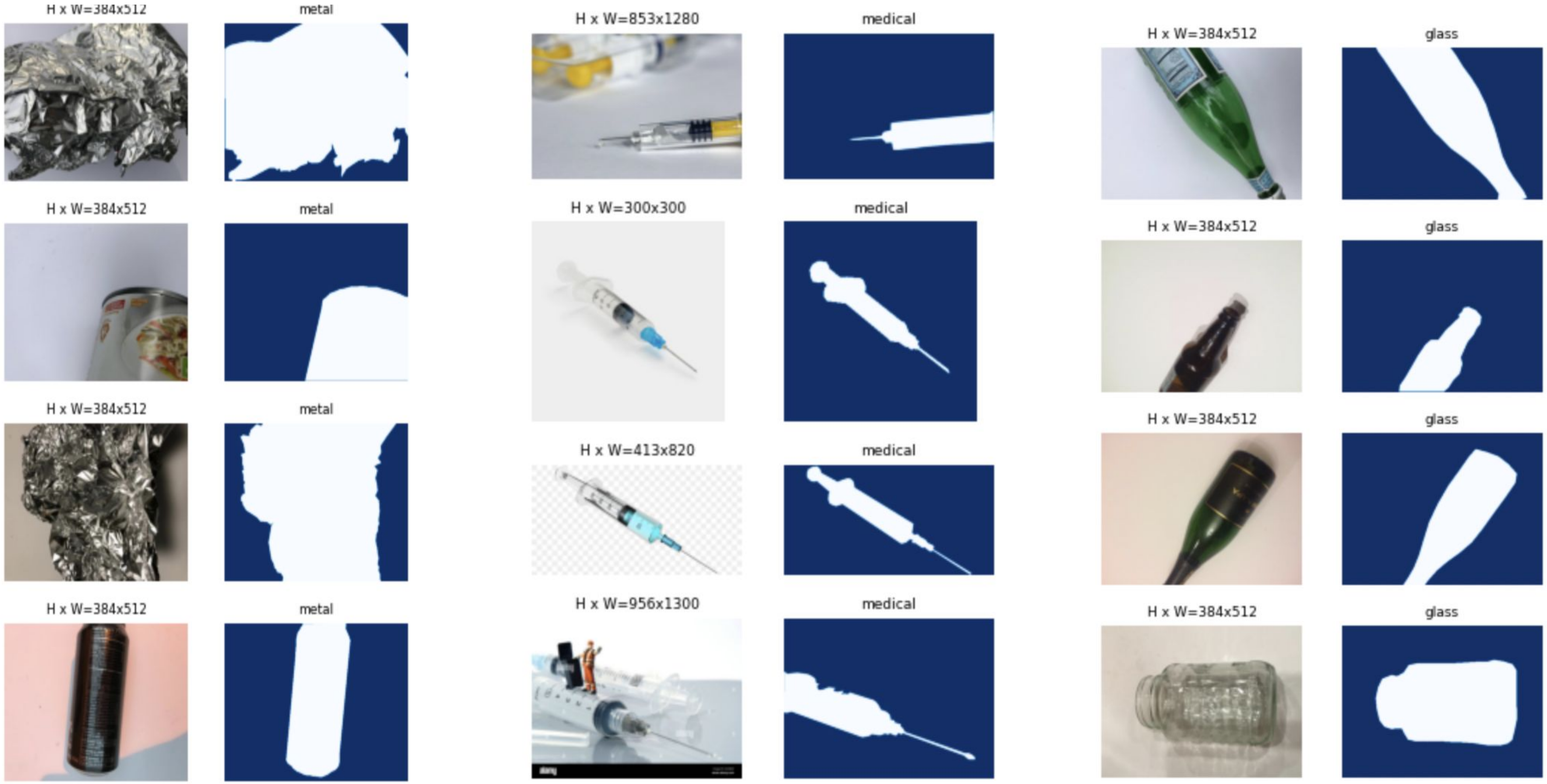
Paper

E-waste

NC STATE UNIVERSITY

Annotations

- The six classes were annotated for the actual masked segments to be found



NC STATE UNIVERSITY

Methodology - Faster RCNN

- **Faster R-CNN** consists of two stages:
  1. Region Proposal Network (RPN)
  2. Fast R-CNN

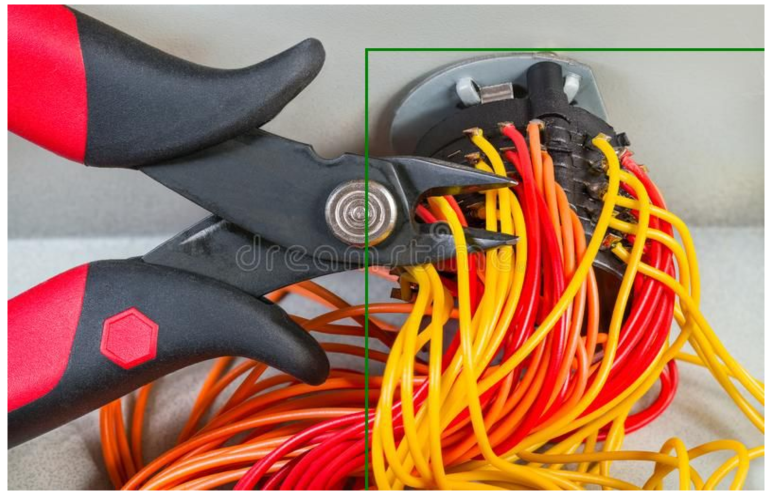
- During training*[1], the model expects input tensors and targets containing:
  - **boxes (FloatTensor[N, 4]):** the ground-truth boxes in [x1, y1, x2, y2] format, with 0 <= x1 < x2 <= W and 0 <= y1 < y2 <= H
  - **labels (Int64Tensor[N]):** the class label for each ground-truth box

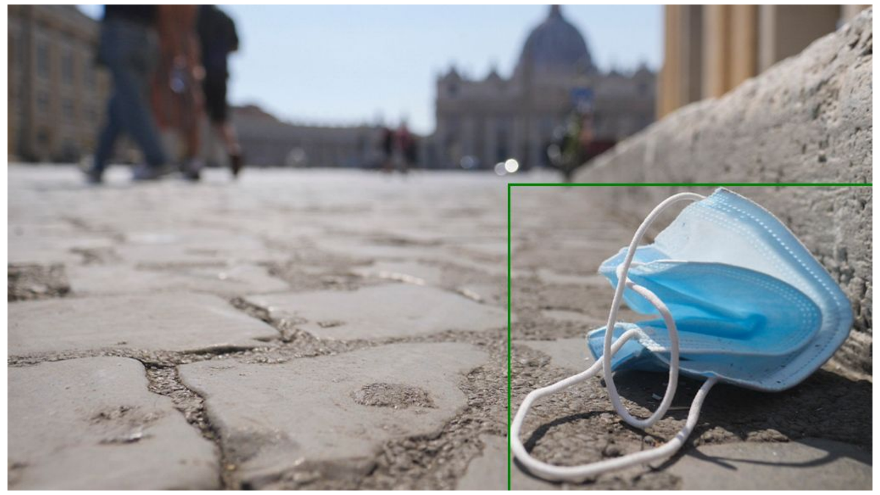
- The model returns a Dict[Tensor] during training, containing the classification and regression losses for both the RPN and the R-CNN.

* **sources** - [1] https://pytorch.org/vision/stable/generated/torchvision.models.detection.fasterrcnn_resnet50_fpn.html

Baseline Performance

```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.001
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.010
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.001
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.002
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.009
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.009
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.009
```





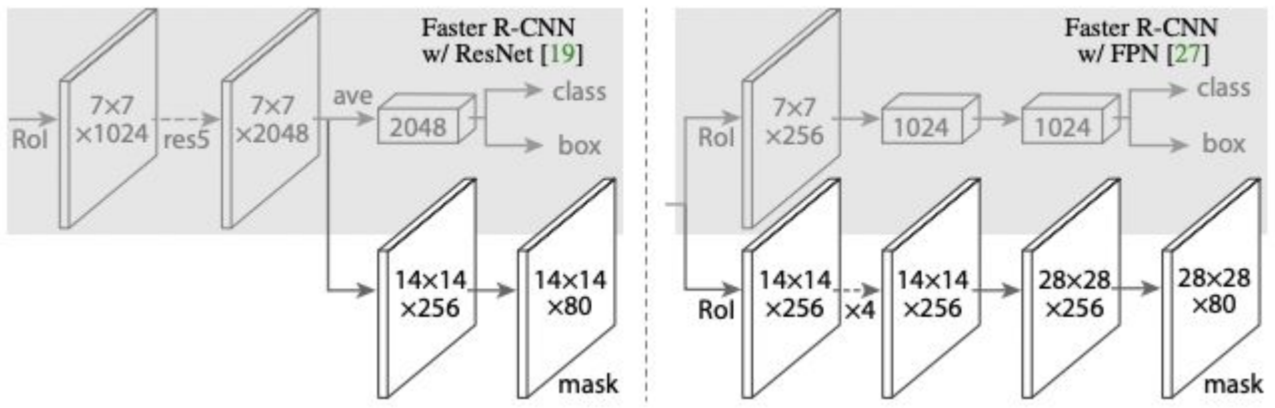
Methodology - Mask RCNN

- **Extends Faster R-CNN** by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition
- **Mask R-CNN** is **simple to train** on Faster R-CNN, running at **5 fps**.

- Two level architecture
  - Convolutional **backbone** architecture used for feature extraction over an entire image.
  - The network **head** for bounding-box recognition (classification and regression) and mask prediction that is applied separately to each RoI.

- A multi-task loss on each sampled RoI is defined as $\mathbf{L = L_{cls} + L_{box} + L_{mask}}$.

- $\mathbf{L_{mask}}$ is defined only on positive RoIs.



* **sources** - [1] arXiv:1703.06870 [cs.CV]

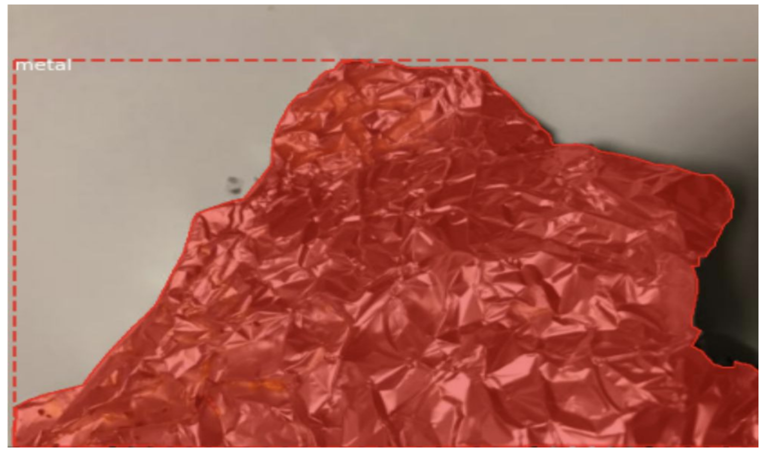
Proposed Model: Mask R-CNN

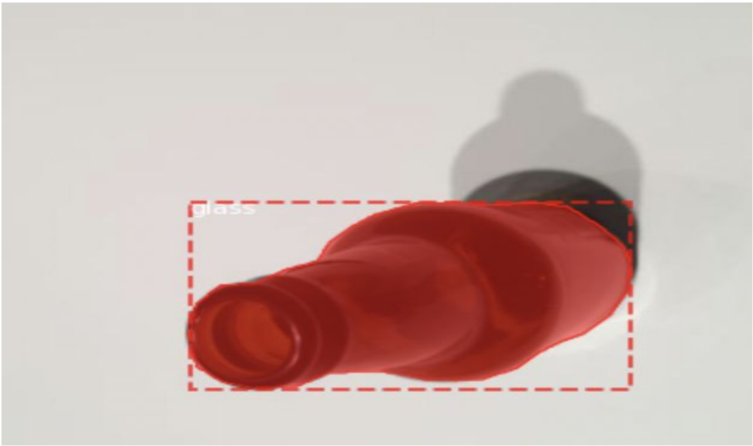
```python
def detect(self, images, verbose=0):

        assert self.mode == "inference", "Create model in inference mode."
        assert len(
            images) == self.config.BATCH_SIZE, "len(images) must be equal to BATCH_SIZE"

        if verbose:
            log("Processing {} images".format(len(images)))
            for image in images:
                log("image", image)

        # Mold inputs to format expected by the neural network
        molded_images, image_metas, windows = self.mold_inputs(images)

        # Validate image sizes
        # All images in a batch MUST be of the same size
        image_shape = molded_images[0].shape
        for g in molded_images[1:]:
            assert g.shape == image_shape,\
                "After resizing, all images must have the same size. Check IMAGE_RESIZE_MODE and image sizes."

        # Anchors
        anchors = self.get_anchors(image_shape)
        # Duplicate across the batch dimension because Keras requires it
        # TODO: can this be optimized to avoid duplicating the anchors?
        anchors = np.broadcast_to(anchors, (self.config.BATCH_SIZE,) + anchors.shape)

        if verbose:
            log("molded_images", molded_images)
            log("image_metas", image_metas)
            log("anchors", anchors)
        # Run object detection
        detections, _, _, mrcnn_mask, _, _, _ =\
            self.keras_model.predict([molded_images, image_metas, anchors], verbose=0)
        # Process detections
        results = []
        for i, image in enumerate(images):
            final_rois, final_class_ids, final_scores, final_masks =\
                self.unmold_detections(detections[i], mrcnn_mask[i],
                                       image.shape, molded_images[i].shape,
                                       windows[i])
            results.append({
                "rois": final_rois,
                "class_ids": final_class_ids,
                "scores": final_scores,
                "masks": final_masks,
            })
        return results
```

NC STATE UNIVERSITY

Predictions

Input Image





Class Detection

NC STATE
UNIVERSITY

THANK YOU