

# Mask R-CNN

Task 2: Detect, localize and semantic segment the home objects using Mask-RCNN on Keras, and TensorFlow

- Tool used for annotation: VGG Image Annotator (VIA)
- Tool used for augmentation: imgaug library

Step 0: Data Exploration

- Key Observations
  - The Ground Truth values provided with the dataset only specify the bounding box coordinates of each sample – need to manually label the images into various classes
  - Most of the classes have just one sample in the training set – need to augment/oversample images in these cases to balance out the dataset
  - Dataset has unbalanced sample size in each class – need to make it more uniform to avoid bias towards high frequency classes and to improve overall accuracy
  - There are 101 images in training and 123 images in test and some objects in test dataset have significant variations in the angle of the camera, scale and lighting
  - Training set lacks enough variety for the model to be able to predict the test data well
  - Training data consists of classes that have more similarity features and less discriminating features making the model unable to distinguish between objects of different classes

Step 1: Annotation

- Used the VIA tool to define polygon regions in each image and labeled them into classes
- Exported the XY coordinates of region definitions for each image into a JSON format
- I ended up with 50 classes for the training data and min count 1 and max count 32 images for a class which is quite unbalanced

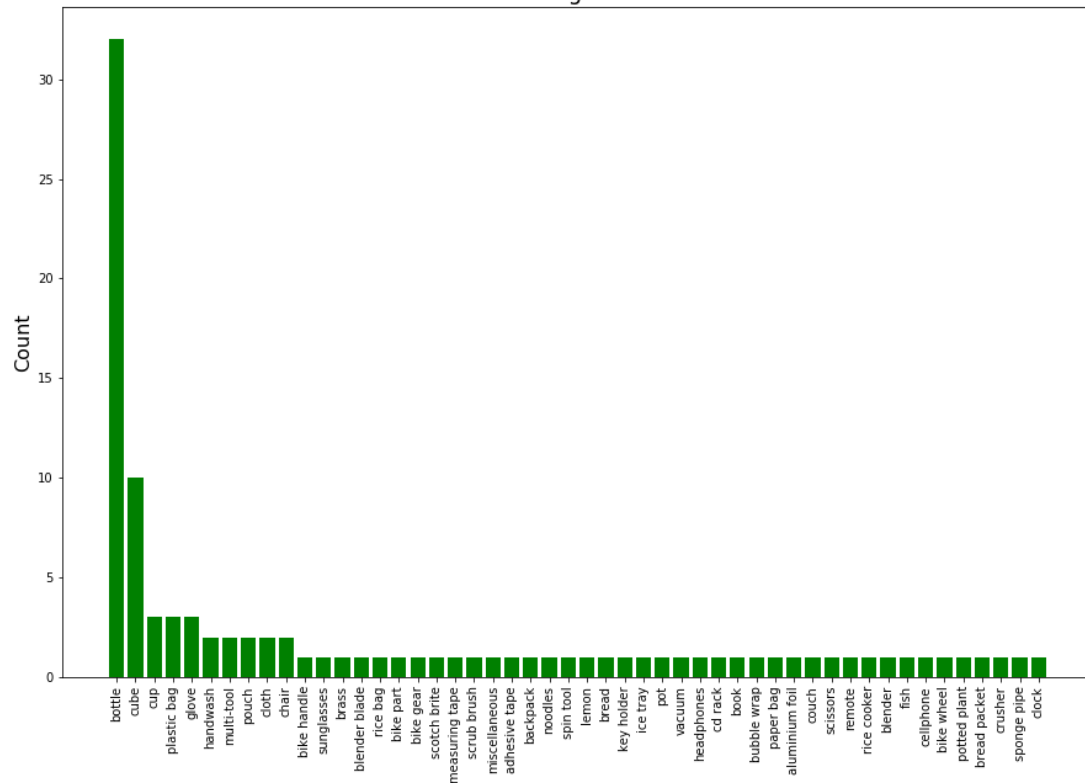
Step 2: Augmentation (added before and after data distribution images)

- Parsed the VIA JSON output file and created binary masks for each image and saved them to disk
- Performed various augmentations on the training images (along with the corresponding masks) like flipping, rotation, shearing, brightness and contrast variations and saved all these training images/masks to disk
- Generated augmentations in a manner that makes the dataset more balanced and less skewed – classes with less samples have more augmentations

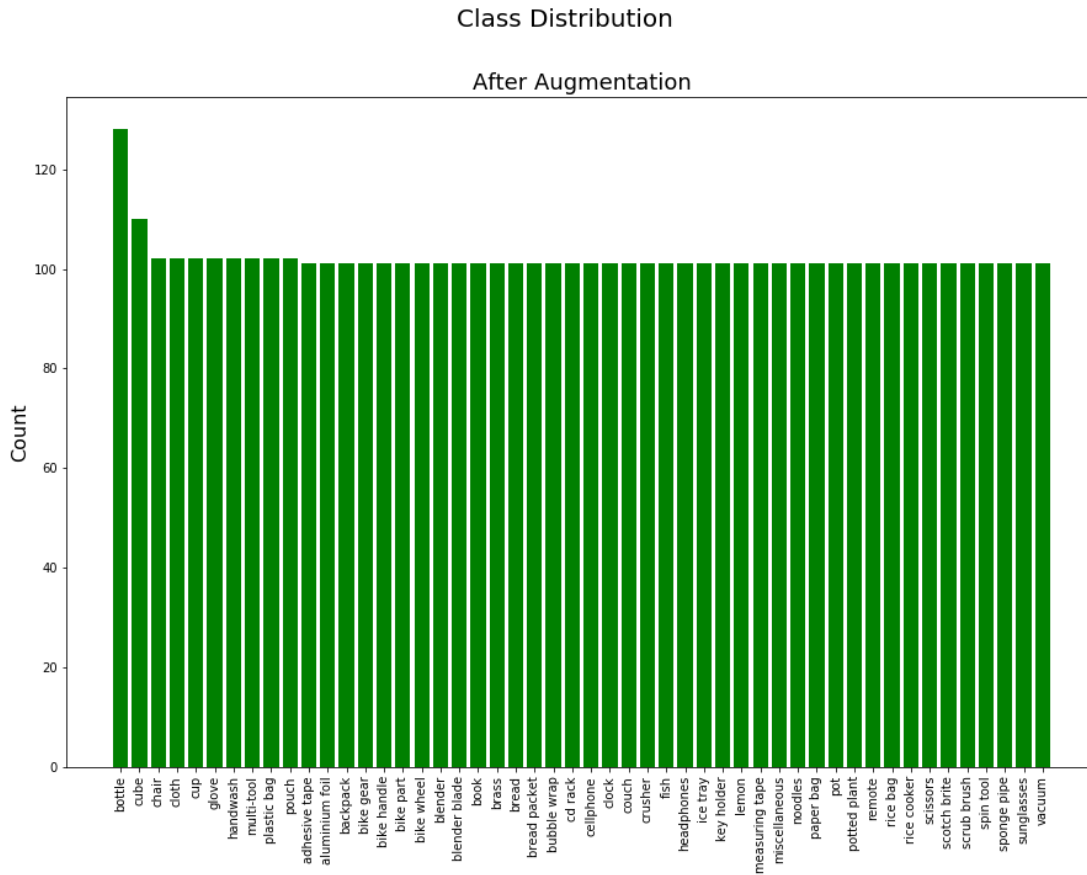
# Mask R-CNN

Class Distribution

Before Augmentation



# Mask R-CNN



## Step 3: Training Steps

- Split the offline augmented data into train/val sets with 0.75/0.25 ratio
- Initialize the network weights with pre-trained weights of model trained on COCO dataset
- We typically fine-tune newly initialized layer heads before training the body of the network
- Training Configuration:

- DETECTION\_MIN\_CONFIDENCE = 0.9
- BACKBONE = “resnet50”
- STEPS\_PER\_EPOCH = 300
- VALIDATION\_STEPS = 60
- LEARNING\_RATE 0.001

Phase 1: Freeze the weights of the body and train the layer heads

- We supply the train and validation set and allow the network to train for 120 epochs with a learning rate of LEARNING\_RATE/10
- Training Time Phase 1: 13189.79978609085 seconds (3.66 hours)

Phase 2: Freeze the weights of the body and train the layer heads

- We supply the train and validation set and allow the network to train for 180 epochs (60 more from phase 1) with a learning rate of LEARNING\_RATE/100
- Training Time Phase 2: 4426.72097492218 seconds (1.23 hours)

Phase 3: Unfreeze the weights of the body and train all the layers

# Mask R-CNN

- We supply the train and validation set and allow the network to train for 210 epochs (30 more from phase 2) with a learning rate of `LEARNING_RATE/100`
- Training Time Phase 3: 6660.193583250046 seconds (1.85 hours)

Total Training Time: 24276.71878027916 seconds (6.74 hours)

## Step 4: Inference Steps

- Find the last trained weights from the disk and load them into a model
- Use this model to predict the objects in an image from the test/unseen dataset
- Average Prediction Time: 0.25 seconds per frame for a video stream and 3.8 seconds per image for a single image
- As expected, the prediction output has poor results for inputs from test dataset since the training data has less samples and low variation per class – this would lead the model to overfit on the training set

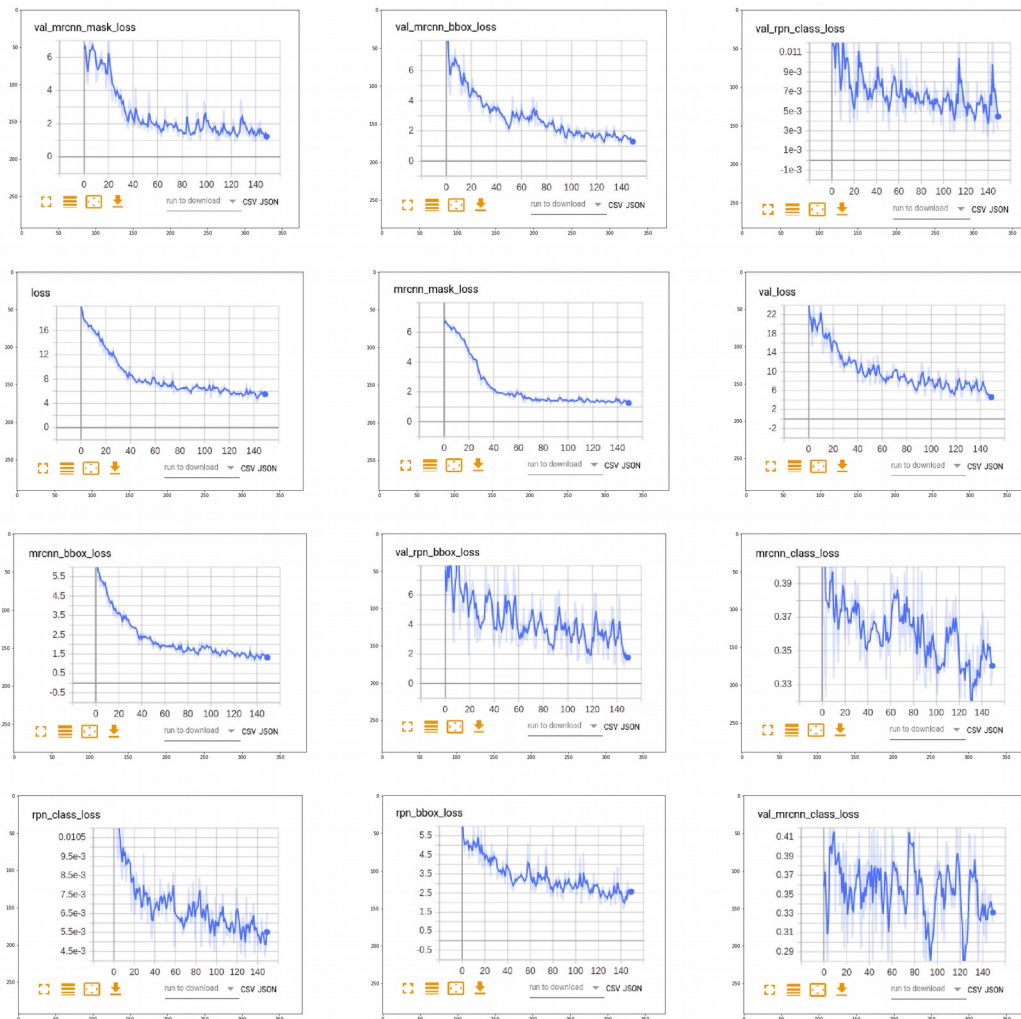
## Room for improvement:

- Learning rate is often the most important hyperparameter to tune
  - Too small learning rate might lead to network not learning at all
  - Too large learning rate may lead to network overshooting areas of low loss or even overfit from the start of the training
    - Accuracy may be improved by selecting the correct learning rate and appropriate learning rate schedule
- Since the model is clearly overfitting the training data, we can increase the weight decay thereby adding some regularization
- Accuracy would improve significantly with more samples and variation per class are provided in the training data – we can prove this by adding more samples in the training set

## Training Loss:

- Observed a steady decline in the overall loss as epochs progressed – this indicates that the model is learning the input data quite well
  - **rpn\_class\_loss** – this metric has a steady decline indicating the model is able to distinguish between the foreground and the background quite well
  - **rpn\_bbox\_loss** - this metric has a steady decline indicating the model is able to reduce the bounding box regression error but unable to fit objects accurately due to lack of objects with varying scales/sizes
  - **mrcnn\_class\_loss** – this metric has a lot of ups and downs indicating the model is unable to learn the class labels well due to less number of training samples and lack of variation (like varying camera angles, sizes and lighting) – the model is unable to distinguish between the classes of unseen samples due to the less number of discriminating features
  - **mrcnn\_mask\_loss** – this metric has a steady decline indicating the model is able to fit the mask of an object and able to efficiently reduce the per-pixel binary cross-entropy loss
  - **mrcnn\_bbox\_loss** - this metric has a steady decline indicating the model is able to reduce the bounding box regression error but unable to fit objects accurately due to lack of objects with varying scales/sizes

# Mask R-CNN



## Task 3: Loss Functions

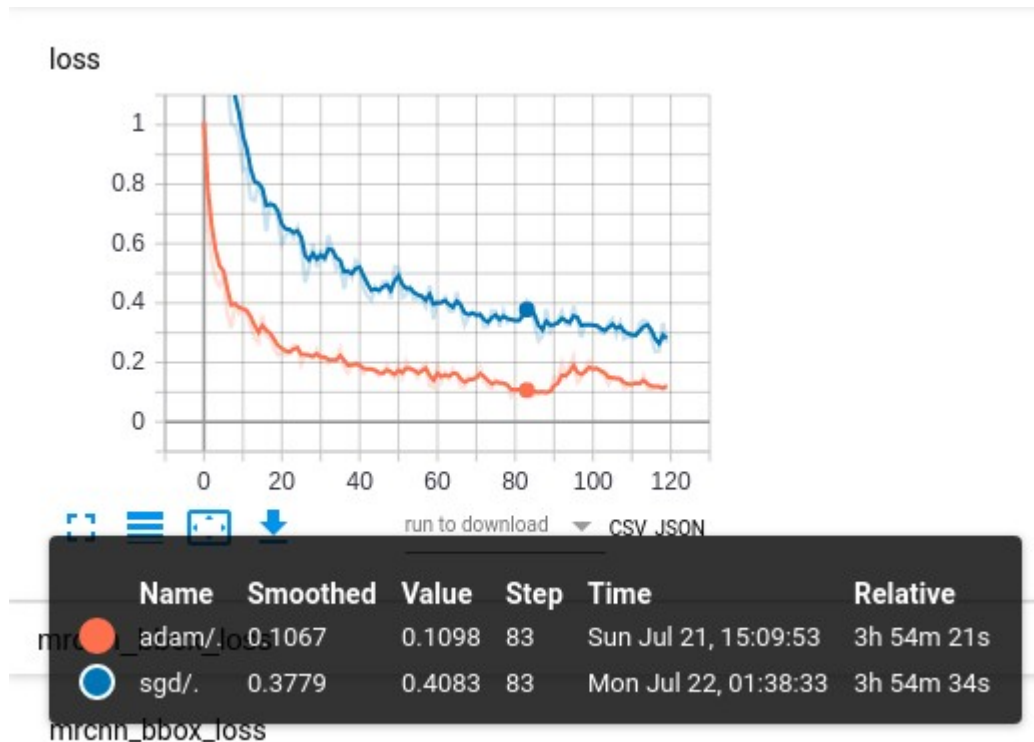
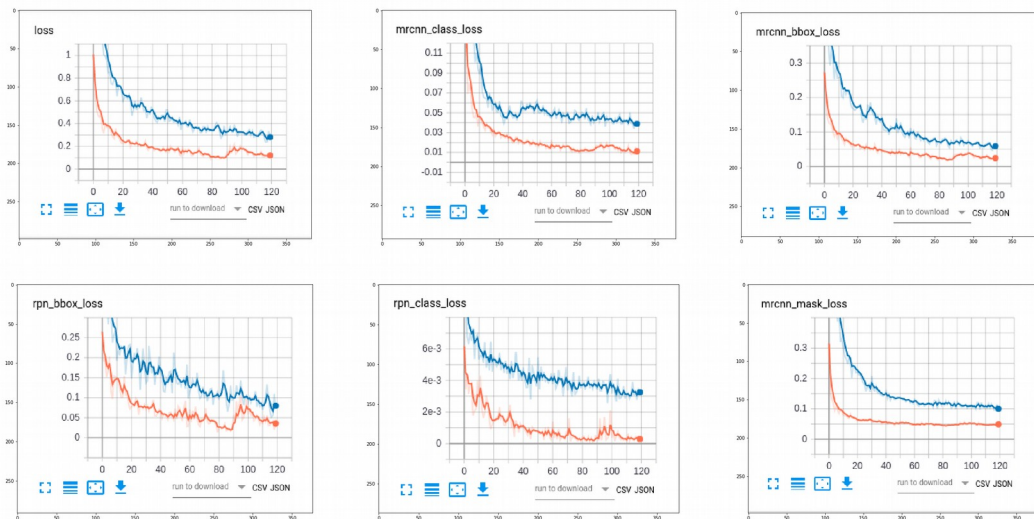
- Current loss function is ( $w1 \cdot \text{rpn\_class\_loss} + w2 \cdot \text{rpn\_bbox\_loss} + w3 \cdot \text{mrcnn\_class\_loss} + w4 \cdot \text{mrcnn\_mask\_loss} + w5 \cdot \text{mrcnn\_bbox\_loss}$ )
- We can tune the weights of individual loss metrics and penalize the ones that most affect the total loss thereby improving the overall performance
- We can add a regularization to the overall loss function to prevent the model from overfitting

## Task 4: Adam vs SGD Optimizer

- As expected, Adam is converging faster compared to SGD (default for Mask R-CNN)

# Mask R-CNN

- We can clearly see around epoch 89, Adam's loss started to increase indicating the model might have overshoot the areas of low loss while SGD has a steady decline in the loss
- Adam definitely performed better for this model configuration and training data – not so much for the validation data due to insufficiency in the training samples



Task 5:

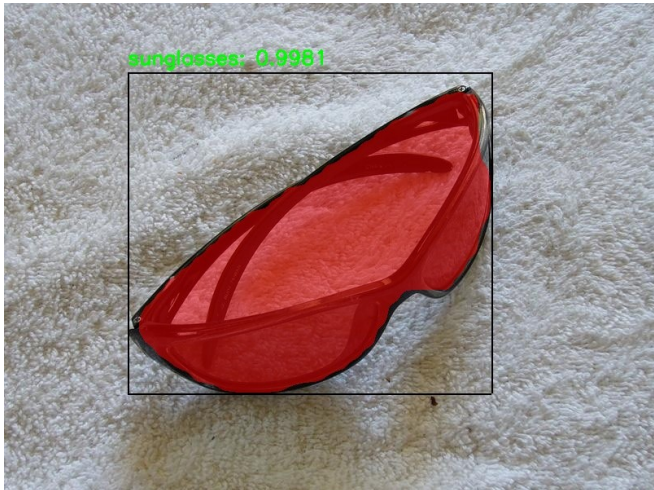

# Mask R-CNN

- Code is hosted at: <https://github.com/vamsimocherla/ObjectDetection>
- Code used to augment images: data\_gt\_aug.py
- Code used to perform training and inference: home\_objects\_aug.py

Task 6:

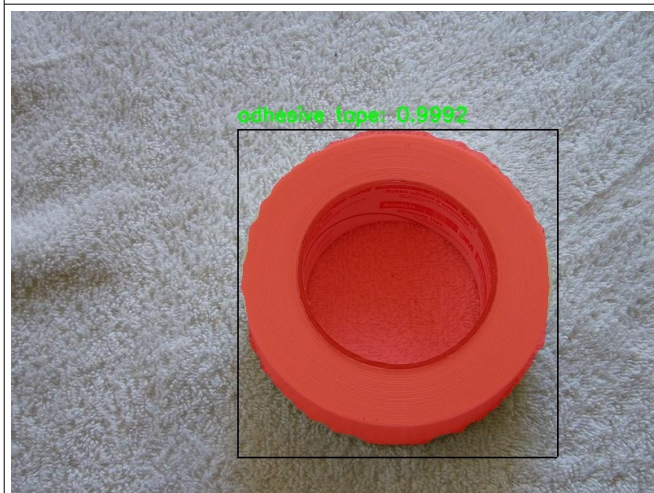
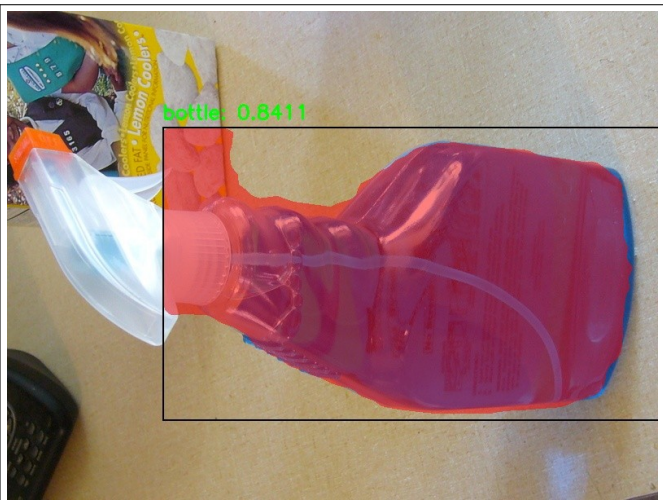
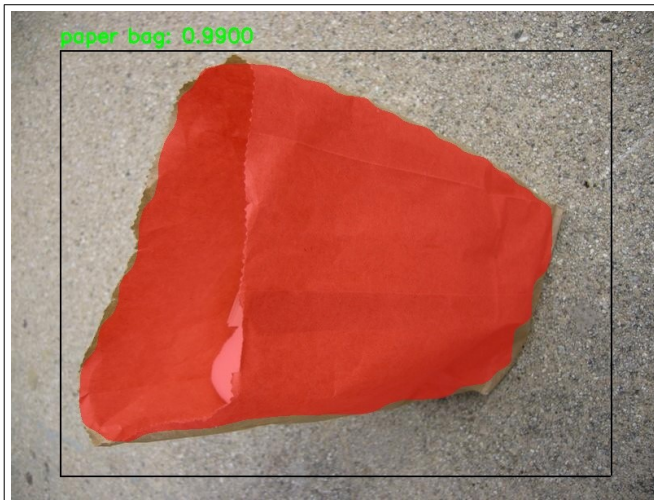
- Augmented training images are hosted at:  
<https://github.com/vamsimocherla/ObjectDetection/tree/master/datasets/HomeObjects06/AugmentedTrain>
- Corresponding masks hosted at:  
<https://github.com/vamsimocherla/ObjectDetection/tree/master/datasets/HomeObjects06/AugmentedMasks>
- Training image annotations hosted at: [https://github.com/vamsimocherla/ObjectDetection/blob/master/datasets/HomeObjects06/home\\_objects\\_train.json](https://github.com/vamsimocherla/ObjectDetection/blob/master/datasets/HomeObjects06/home_objects_train.json)
- Augmentation metrics of each image hosted at:  
[https://github.com/vamsimocherla/ObjectDetection/blob/master/datasets/HomeObjects06/home\\_objects\\_train\\_aug.json](https://github.com/vamsimocherla/ObjectDetection/blob/master/datasets/HomeObjects06/home_objects_train_aug.json)

Some Results:

Train Set	Test Set
	



# Mask R-CNN





## Mask R-CNN

