Image Segmentation for Object Detection

Overview

Humans can easily detect and identify objects present in an image as compared to a computer.

However, with the availability of large amount of data, refer GPUs, and better algorithms, we can now easily train computers to detect and classify multiple object within an image with high accuracy.

We explore term such as object detection, localization, loss function for object detection and finally explore an object detection algorithm.

Problem Statement

Conventional object detectors use bounding boxes to label the data while training and output an array of bounding boxes around objects detected in the image. This method of detecting objects is old.

In our project, we will use Detectron2 for image segmentation and object detection.

Detectron2 is Facebook’s recently introduced AI Research's (FAIR) framework.

Includes Mask R-CNN, RetinaNet, Faster R-CNN, RPN, Fast R-CNN, TensorMask, PointRend, DensePose, and more.

Methodology

Procedure to train the model and use it for prediction:

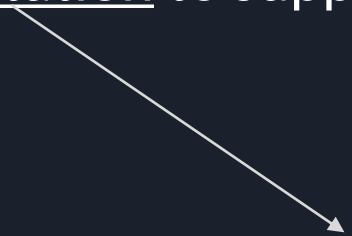
- Building custom dataset
- Registering the Dataset
- Training the model
- Saving the model
- Making Inference

Each team member has taken up two object classes, extracted the images either from Google or from their home environment, and then created the dataset using LabelMe.

Creating datasets Using **LABELME**

GOAL

To create image annotation to supply it to our model for training purposes.



https://viso.ai/computer-vision/image-annotation/

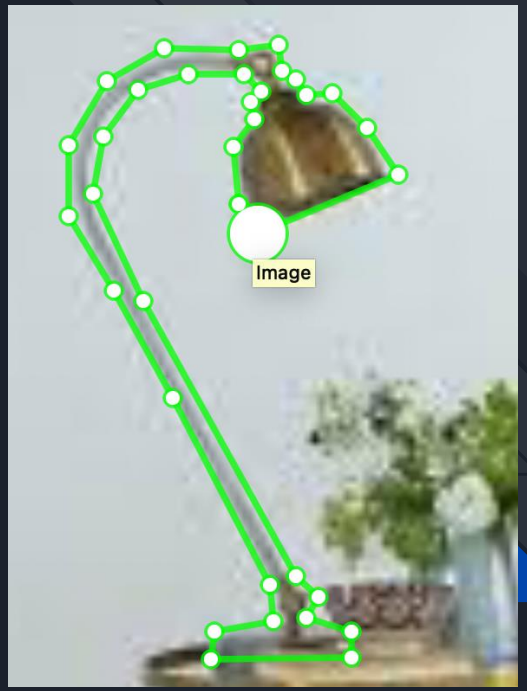
Reason for selecting LabelMe

- Labelme is **a graphical image annotation tool**
- It has a simple user interface
- It allows us to create and edit polygons around object
- Generates corresponding JSON file with all the information about the image

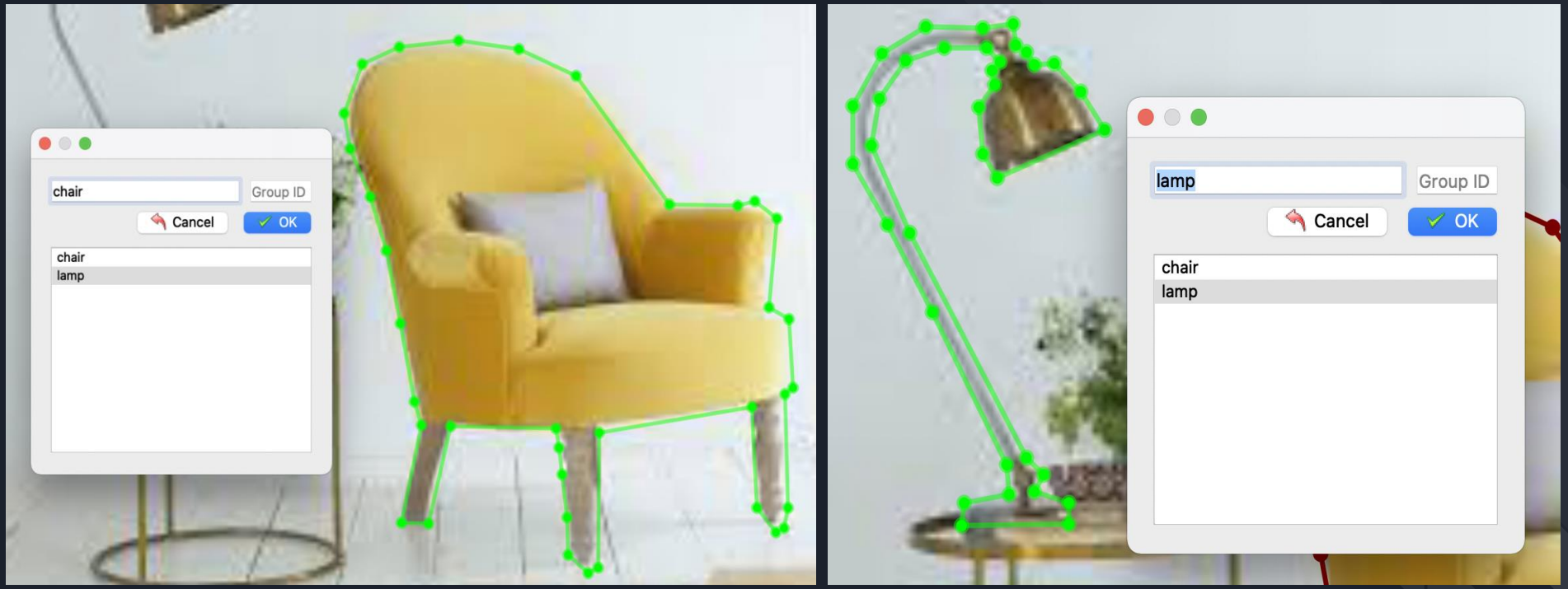
Selecting objects for classes using LABELME

Chair

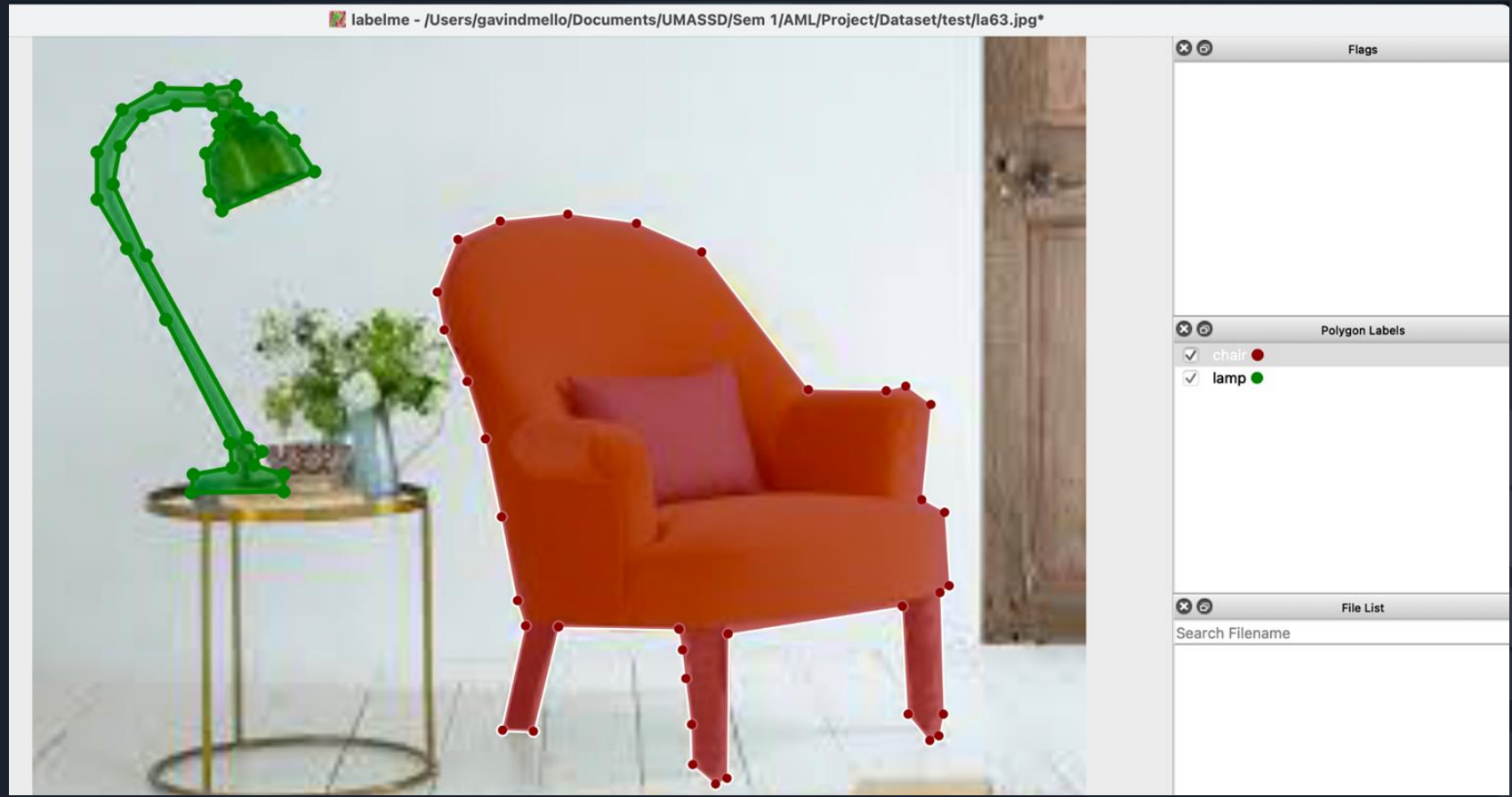
Lamp



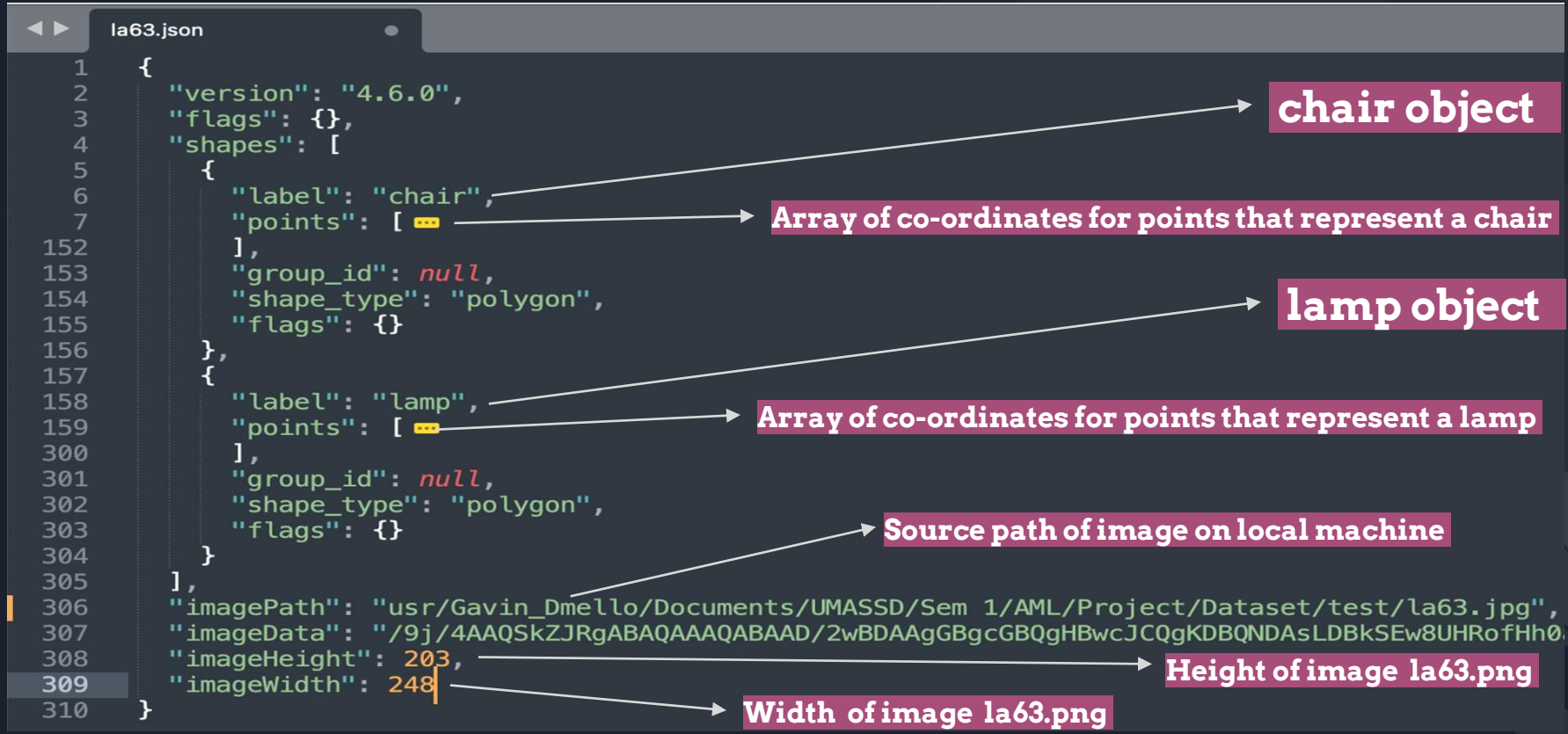
Labelling polygons for classes using LABELME



Multiple polygons in one single image

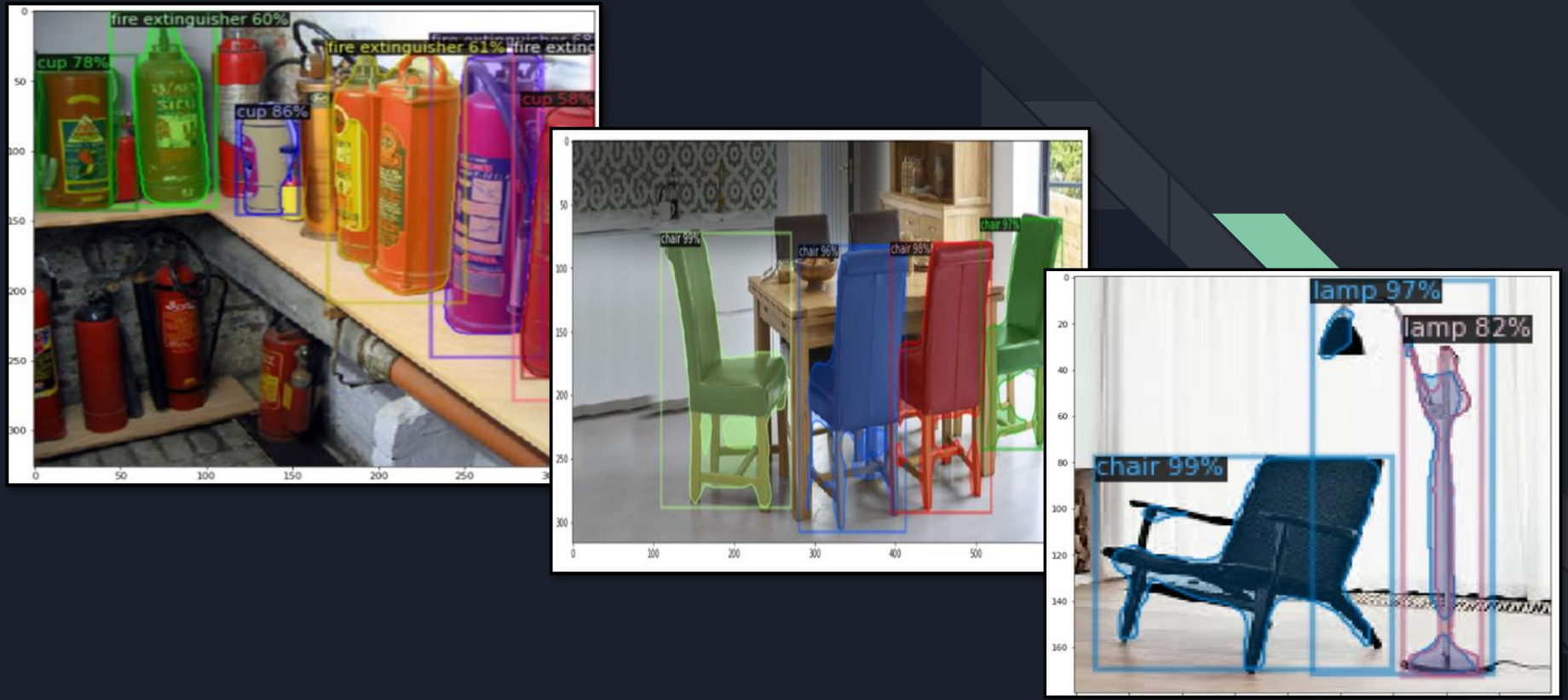


Output JSON File from LABELME for an Image




```
{
  "version": "4.6.0",
  "flags": {},
  "shapes": [
    {
      "label": "chair",
      "points": [ ...
      ],
      "group_id": null,
      "shape_type": "polygon",
      "flags": {}
    },
    {
      "label": "lamp",
      "points": [ ...
      ],
      "group_id": null,
      "shape_type": "polygon",
      "flags": {}
    }
  ],
  "imagePath": "usr/Gavin_Dmello/Documents/UMASSD/Sem 1/AML/Project/Dataset/test/la63.jpg",
  "imageData": "/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRofHh0
  "imageHeight": 203,
  "imageWidth": 248
}
```

Results


fire extinguisher 60%
cup 78%
fire extinguisher 61%
cup 86%
chair 99%
chair 96%
chair 98%
lamp 97%
lamp 82%
chair 99%

Rigorous Testing

Object Number	Object Class	File	Image Size (Pixels)	Image Size (Pixels)	Object orientation (Degrees)	Object Colour	Object size (Pixels)	Object size (Pixels)	Predicted Confidence Score (%)
6	Iron	I72.jpg	900	600	0	RBG	500	500	92
			900	600	0	RBG	300	300	91
			900	600	0	RBG	150	150	86
			900	600	0	RBG	75	75	Did not Predict
			900	600	90	RBG	300	300	65
			900	600	180	RBG	300	300	wrong Predict
			900	600	0	GS	300	300	93
0xFFFFF									

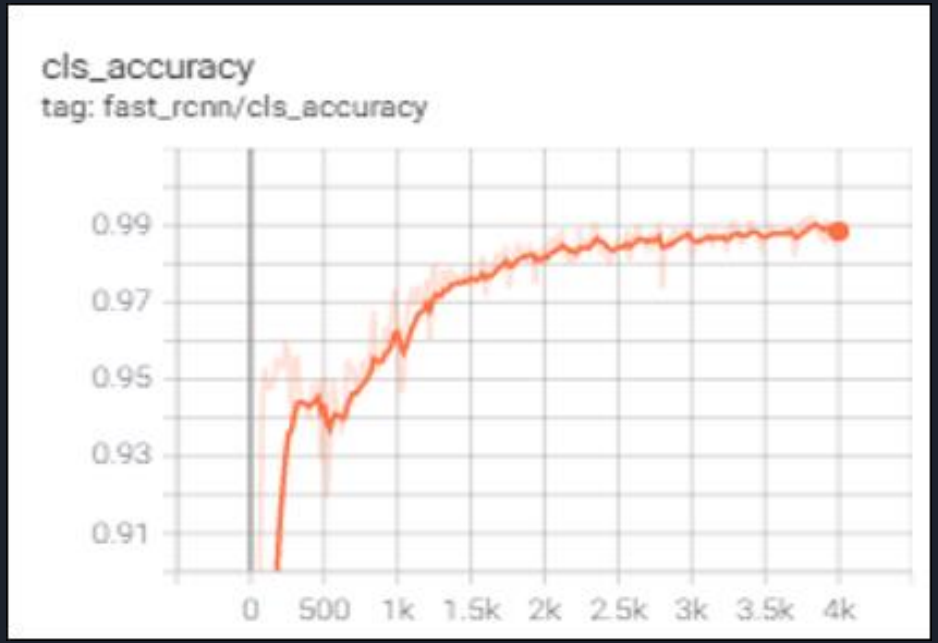
← Keeping the image size, orientation and object colour constant and varying the object size.

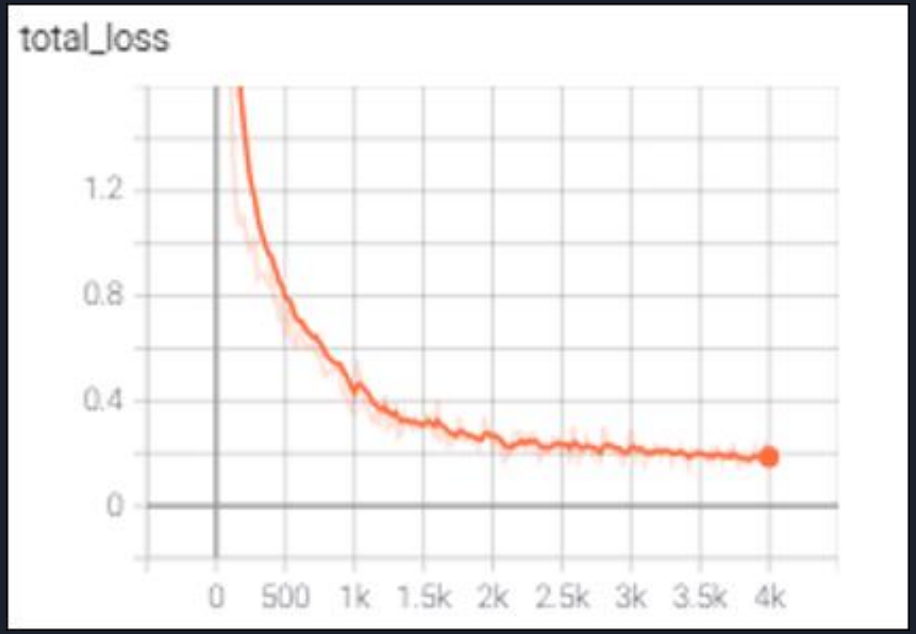
Rigorous Testing

Object Number	Object Class	File	Image Size (Pixels)	Image Size (Pixels)	Object orientation (Degrees)	Object Colour	Object size (Pixels)	Object size (Pixels)	Predicted Confidence Score (%)
6	Iron	I72.jpg	900	600	0	RBG	500	500	92
			900	600	0	RBG	300	300	91
			900	600	0	RBG	150	150	86
			900	600	0	RBG	75	75	Did not Predict
			900	600	**90**	RBG	300	300	65
			900	600	**180**	RBG	300	300	wrong Predict
			900	600	0	GS	300	300	93
0xFFFFF									

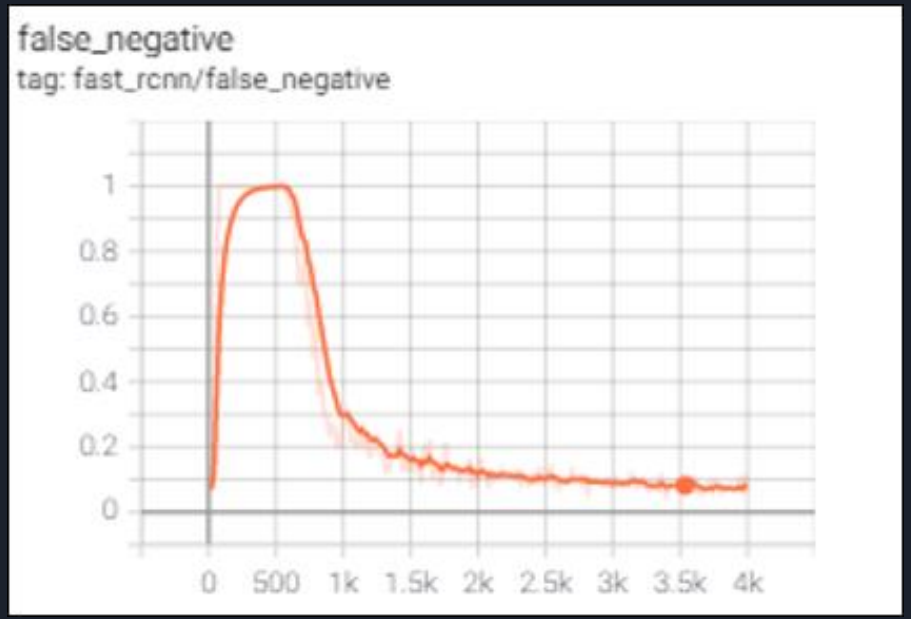
← Keeping the image size, object color and object size constant and varying the *Orientation*.

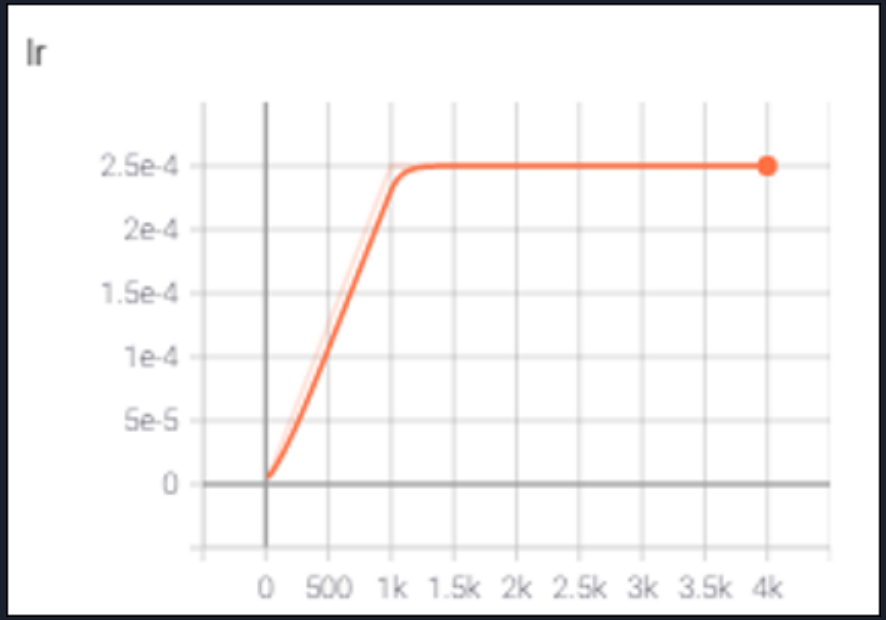
Performance Evaluation
(Training Accuracy)





Performance Evaluation
(Training Accuracy)





Performance Evaluation
(Testing Accuracy)

```
[05/10 06:39:23 d2.evaluation.fast_eval_api]: Evaluate annotation type *bbox*
[05/10 06:39:23 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finished in 0.01 seconds.
[05/10 06:39:23 d2.evaluation.fast_eval_api]: Accumulating evaluation results...
[05/10 06:39:23 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finished in 0.02 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.709
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.923
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.815
 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 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.743
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.649
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.753
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.753
 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 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.786
[05/10 06:39:23 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs  |  APm  |  APl   |
|:------:|:------:|:------:|:-----:|:-----:|:------:|
| 70.858 | 92.331 | 81.518 |  nan  | 0.000 | 74.329 |
```

Figure displays the Average Precision(AP) and Recall of the trained model

Performance Evaluation
(Testing Accuracy)

```
[05/10 06:39:23 d2.evaluation.coco_evaluation]: Some metrics cannot be computed and is shown as NaN.
[05/10 06:39:23 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category   | AP     | category   | AP     | category        | AP     |
|:-----------|:-------|:-----------|:-------|:----------------|:-------|
| chair      | 57.723 | lamp       | 74.950 | cup             | 93.399 |
| clock      | 86.634 | fan        | 24.257 | fire extinguisher | 59.703 |
| kettle     | 90.000 | iron       | 68.529 | laptop          | 82.525 |
| cell phone | nan    | table      | nan    |                 |        |
```

Figure displaying the Average Precision(AP) of individual Classes


THANK YOU.