Preliminary test with open resource : Introduction, Mask R-CNN

- Algorithm: Mask R-CNN(R-CNN, Regional convolutional neural networks)
- Pros and cons of mask R-CNN to object detection (compare to previous approach, i.e. faster R-CNN)
  - ✓ Pros : Reduce misalignment after RoIPooling by “RoIAlign” => Increase AP

		AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}	AP^{kp}_{M}	AP^{kp}_{L}
Previous RoI Pooling →	*RoIPool*	59.8	86.2	66.7	55.1	67.4
Novel RoI Pooling →	*RoIAlign*	**64.2**	**86.6**	**69.7**	**58.7**	**73.0**

* Ref.> K. He et al., Mask R-CNN, *The IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2961-2969

  - ✓ Cons: No benefit to reduce time & complex

Method	time (fps)
Faster RCNN	200ms
Mask RCNN	2.5

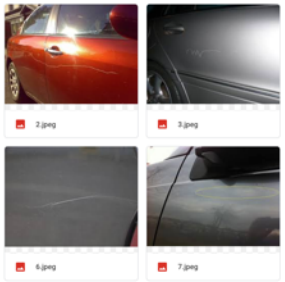
* Ref.> Gang Yu, http://www.skicyyu.org/Presentation/Intro_Object%20Detection.pdf

Preliminary test with open resource : Experiment details, Mask R-CNN

- Open resource: by Nicolas Metallo (https://github.com/nicolasmetallo/car-damage-detector)
  - ✓ Note: Edited version of Mask R-CNN algorithm known as “Balloon model” by Matterport,Inc, CA, USA (https://github.com/matterport/Mask_RCNN/tree/master/samples/balloon)

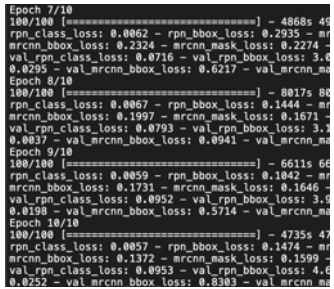
- Implementation detail:

Image collection	Modify & Adjusting the Model	Image annotation	Train the machine with our images	Test with splashing



- ✓ Modify the code to adjust with our running Env.
- ✓ Fix the errors
  - Scikit-image
  - Tensorflow placeholder etc..

- ✓ Train: 29 images
- ✓ Val.: 4 images
- ✓ W/O category



Challenges and improvement :MR-CNN

Challenges

- Speed and training time

Solution

- AWS-Free tier
- GPU with G-Colab

Result

- 10x slower
- 10x faster

Training condition of MR-CNN

Training images	Validation images	Training Epoch	Steps per Epoch	Callbacks
• 330	• 52	• 50	• 1000	• mAp • precision • recall • loss

Training system	Annotation	class	weight	Backbone
• Laptops • AWS EC2 • G- Colab	• polygon • via VIA	• 1 • ‘scratch’	• coco	• ResNet 101

Modification of MR-CNN package

@ model.py from mrcnn package

```
def train(self, train_dataset, val_dataset, learning_rate, epochs, layers,
          augmentation=None, custom_callbacks=None):
```

```
    # Callbacks
    callbacks = [
        keras.callbacks.TensorBoard(log_dir=self.log_dir,
                                    histogram_freq=0, write_graph=True, write_images=False),
        keras.callbacks.ModelCheckpoint(self.checkpoint_path,
                                        verbose=0, save_weights_only=True),
    ]

    # Add custom callbacks to the list
    if custom_callbacks:
        callbacks += custom_callbacks
```

@ Our algorithm

```
class Metrics(Callback):

    def on_train_begin(self, logs={}):
        self.image_id = []
        self.add_class = []
        self.add_image =[]
        self.image_info = []

        # Training dataset.
        dataset_train = CustomDataset()
        dataset_train.load_custom(args.dataset, "train")
        dataset_train.prepare()


    def get_ax(rows=1, cols=1, size=8):
        """Return a Matplotlib Axes array to be used in
        all visualizations in the notebook. Provide a
        central point to control graph sizes.

        Change the default size attribute to control the size
        of rendered images
        """
        _, ax = plt.subplots(rows, cols, figsize=(size*cols, size*rows))
        return ax

    def on_epoch_end(self, epoch, logs={}):

        #dataset = CustomDataset()
```

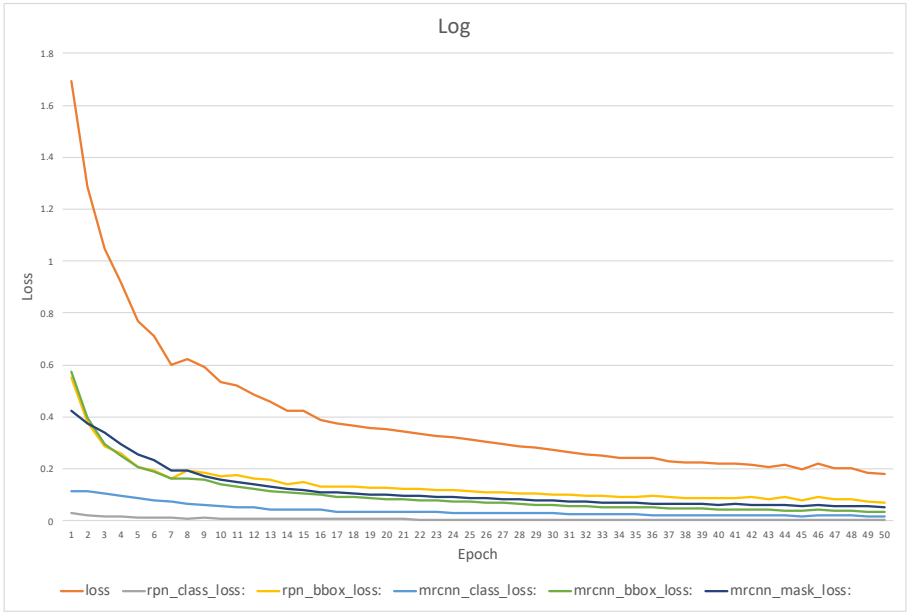
```
model.train(train_dataset=dataset_train,
            val_dataset=dataset_val,
            learning_rate=config.LEARNING_RATE,
            epochs=50,
            layers='heads',
            custom_callbacks=[metrics])
```

@ Result

```
AP @0.50:         0.000
AP @0.55:         0.000
AP @0.60:         0.000
AP @0.65:         0.000
AP @0.70:         0.000
AP @0.75:         0.000
AP @0.80:         0.000
AP @0.85:         0.000
AP @0.90:         0.000
AP @0.95:         0.000
AP @0.50-0.95:    0.000
```

Result of MR-CNN

Log



Result of MR-CNN : segmentation







Result of MR-CNN : Errors