Configuration Manual

MSc Research Project
Data Analytics

Devashish Vijay Rayate
Student ID: X19232616

School of Computing
National College of Ireland

Supervisor: Prof. Hicham Rifai

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Devashish Vijay Rayate
Student ID:	X19232616
Programme:	Data Analytics
Year:	2020
Module:	MSc Research Project
Supervisor:	Prof. Hicham Rifai
Submission Due Date:	16/08/2021
Project Title:	Configuration Manual
Word Count:	1057
Page Count:	9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Devashish Vijay Rayate
Date:	15th August 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Devashish Vijay Rayate
X19232616

1 Introduction

The objective of this documentation is to list out all the activities to be performed during the project implementation stage. In order to recreate the project in the future, software and hardware requirements are outlined. This article covers the coding and deployment processes, as well as the procedures that must be completed to run the code.

2 System Configurations

2.1 Hardware Configuration

Figure 1 below shows the hardware configuration of the system on which the code was implemented.

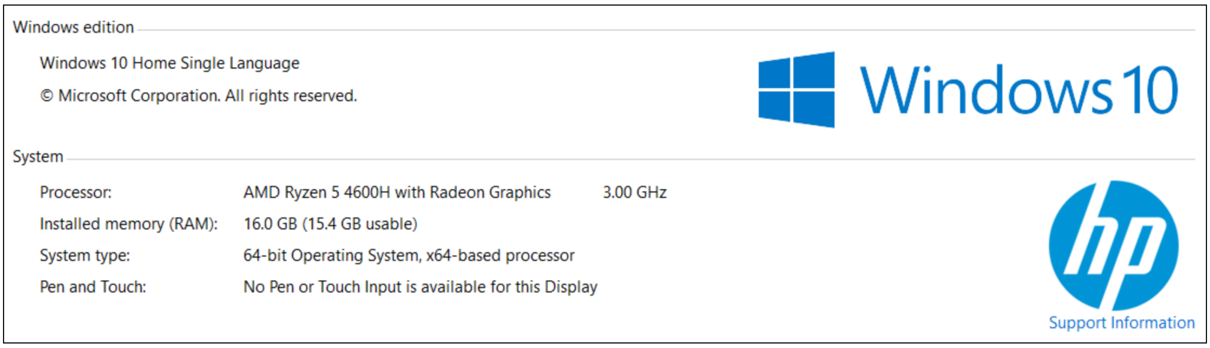


Figure 1: Hardware configuration

2.2 Software Configuration

This section contains information about the software that were used to implement the research, as well as their specifications.

2.2.1 Google Colaboratory

Google's computing infrastructure, also called as Google Colab, is used for the project. All of the libraries have been loaded, and the model is being coded in Google Colab.

The dataset is uploaded on Google Drive which is then connected to Google Colab using following code:

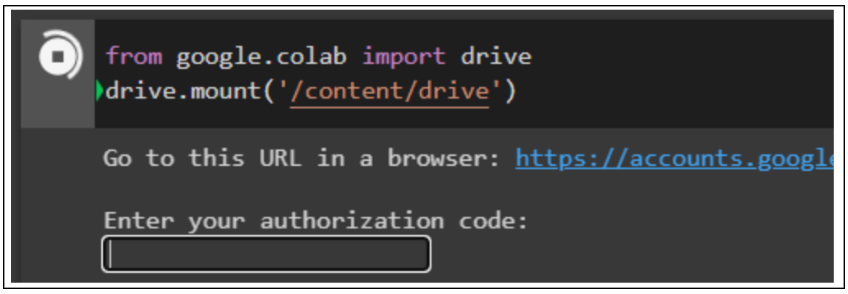


Figure 2: Mounting Google Drive on Google Colab

Following the execution of the command in Figure 2, a hyperlink to get authorization code is displayed, and if we click on it, an authorization key is produced. Copy that code and paste it into the colab's input box, and the drive will be mounted successfully.

We'll switch the Colab notebook's runtime to GPU because it makes image processing models run faster. Colab's runtime can be changed by going to the 'Runtime' menu, then 'Change Runtime Type,' and selecting the GPU option.

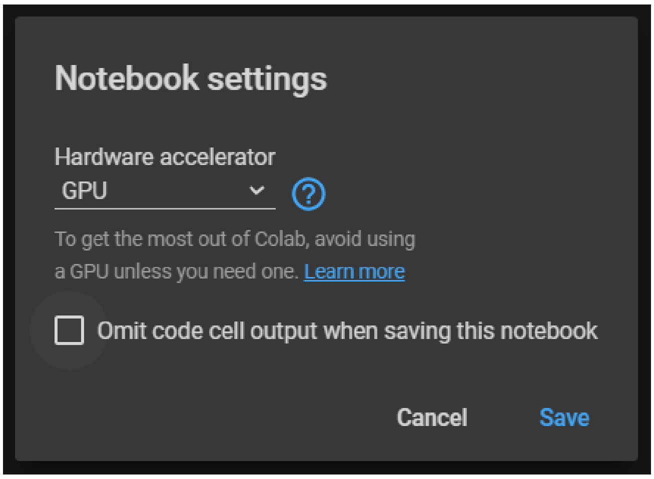


Figure 3: Changing Runtime Type to GPU on Google Colab

2.2.2 Other Software Used

Google chrome web browser was used to access Google Colab. TeXstudio is used for project report documentations which supports creation of documents using Latex. The software is user friendly and it helps to create latex documents very easily. Figure 4 below shows UI of TeXstudio.

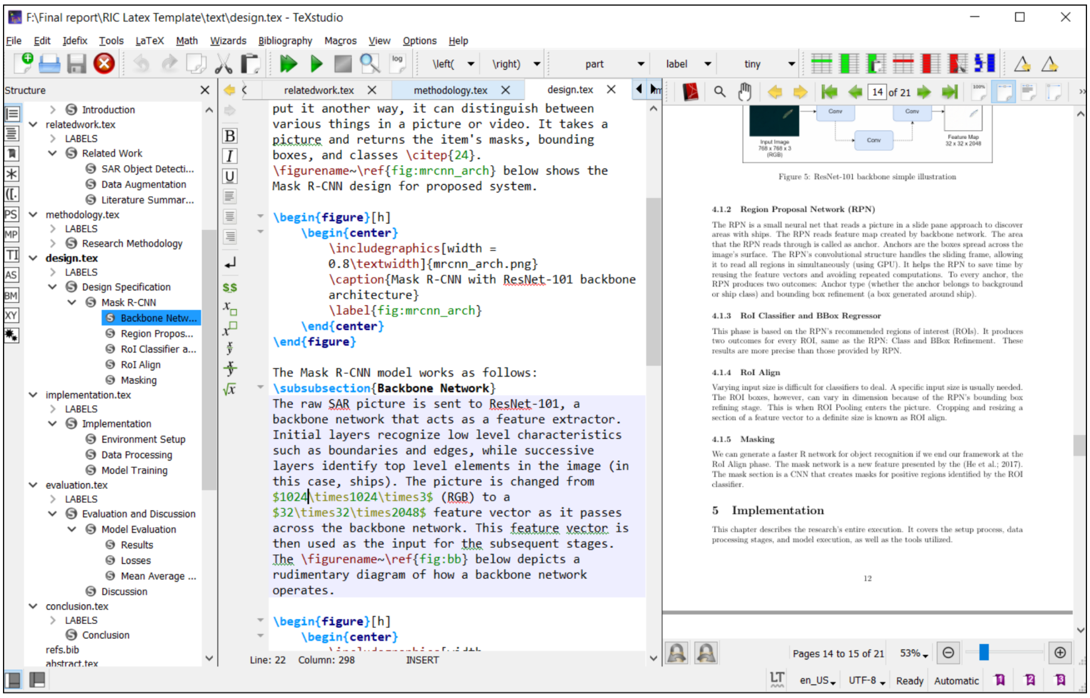


Figure 4: Documentation using TeXstudio

3 Data Preparation

Dataset used for the research is downloaded from Kaggle's 'Airbus Ship Detection challenge'[1] shown in Figure 5

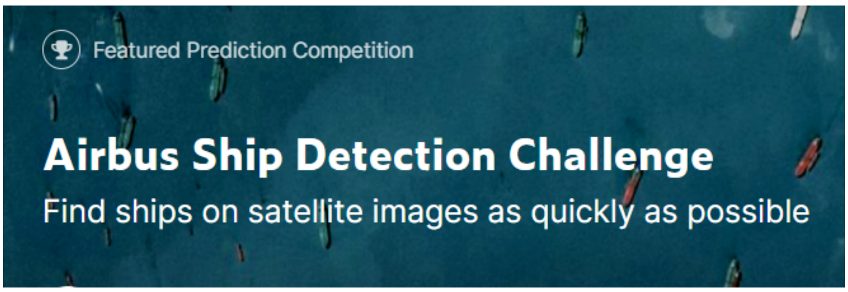


Figure 5: 'Airbus Ship Detection' dataset for project implementation

The dataset contains 2 folders: train_v2 and test_v2. It also contains one annotation file which has ImageIds and EncodedPixels information. The annotations are provided in RLE format. As shown in Figure 6, we uploaded our dataset on Google Drive.

[1]https://www.kaggle.com/c/airbus-ship-detection

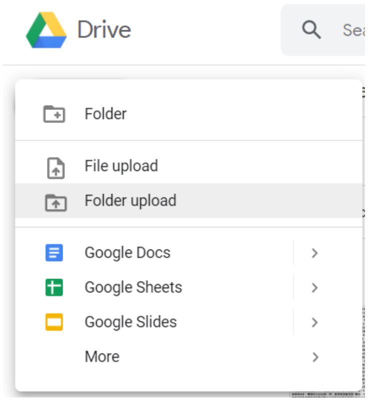


Figure 6: Upload dataset on Google Drive

After uploading the folders on Google Drive, we read the data using below code and split the data under train_v2 folder into training and testing dataset with ratio of 75:25 as shown in Figure 7.

```
#ignore images that do not contain any ships and create train and validation dataset
train_names1 = anns[anns.EncodedPixels.notnull()].ImageId.unique().tolist()  ## override with ships
train_names = [f for f in train_files if f not in exclude_list and f in train_names1]

#test_size = config.VALIDATION_STEPS * config.IMAGES_PER_GPU
image_fps_train, image_fps_val = train_test_split(train_names, train_size = 0.75, test_size=0.25)
print(len(train_names1))
if debug:
    image_fps_train = image_fps_train[:100]
    image_fps_val = image_fps_val[:100]
    test_names = test_names[:100]

print(len(image_fps_train), len(image_fps_val))

8499 2833
```

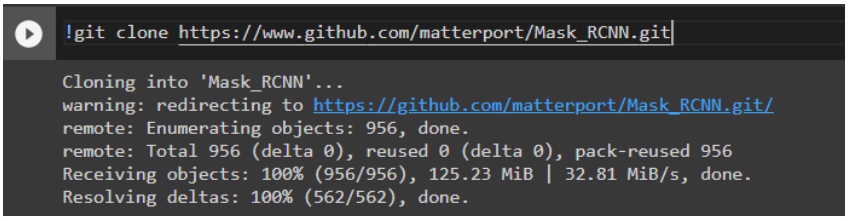
Figure 7: Splitting the dataset into training and testing set

4 Model Implementation

Implementing Mask R-CNN along with data augmentation is novelty of the project. Mask R-CNN is a straightforward and effective object segmentation approach for object recognition applications. It achieved first position at COCO 2016 Competition (He et al.; 2017). In this research we used the Mask R-CNN library which is publicly available on github[2]. It allows us to create an object detection model using Mask R-CNN. We can also add our own code into those script as required.

[2]https://github.com/matterport/Mask_RCNN

The code snippet of 'git clone' command shown in Figure 8 below is used to clone the github repository into our colab environment.

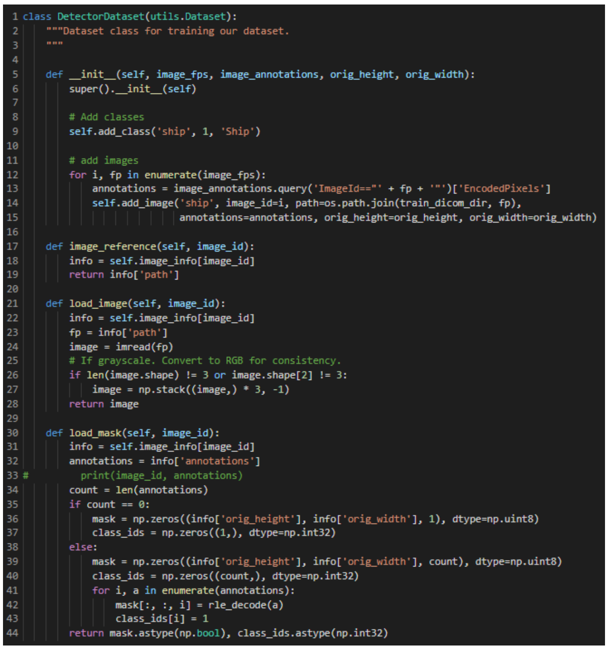


```
!git clone https://www.github.com/matterport/Mask_RCNN.git

Cloning into 'Mask_RCNN'...
warning: redirecting to https://github.com/matterport/Mask_RCNN.git/
remote: Enumerating objects: 956, done.
remote: Total 956 (delta 0), reused 0 (delta 0), pack-reused 956
Receiving objects: 100% (956/956), 125.23 MiB | 32.81 MiB/s, done.
Resolving deltas: 100% (562/562), done.
```

Figure 8: Splitting the dataset into training and testing set

We will write a few functions to decode the RLE encodings present in annotation file. We will also create a class for associating these encodings to each image and preparing them for training.



```
class DetectorDataset(utils.Dataset):
    """Dataset class for training our dataset.
    """

    def __init__(self, image_fps, image_annotations, orig_height, orig_width):
        super().__init__(self)

        # Add classes
        self.add_class('ship', 1, 'Ship')

        # add images
        for i, fp in enumerate(image_fps):
            annotations = image_annotations.query('ImageId=="' + fp + '"')['EncodedPixels']
            self.add_image('ship', image_id=i, path=os.path.join(train_dicom_dir, fp),
                           annotations=annotations, orig_height=orig_height, orig_width=orig_width)

    def image_reference(self, image_id):
        info = self.image_info[image_id]
        return info['path']

    def load_image(self, image_id):
        info = self.image_info[image_id]
        fp = info['path']
        image = imread(fp)
        # If grayscale. Convert to RGB for consistency.
        if len(image.shape) != 3 or image.shape[2] != 3:
            image = np.stack((image,) * 3, -1)
        return image

    def load_mask(self, image_id):
        info = self.image_info[image_id]
        annotations = info['annotations']
#         print(image_id, annotations)
        count = len(annotations)
        if count == 0:
            mask = np.zeros((info['orig_height'], info['orig_width'], 1), dtype=np.uint8)
            class_ids = np.zeros((1,), dtype=np.int32)
        else:
            mask = np.zeros((info['orig_height'], info['orig_width'], count), dtype=np.uint8)
            class_ids = np.zeros((count,), dtype=np.int32)
            for i, a in enumerate(annotations):
                mask[:, :, i] = rle_decode(a)
                class_ids[i] = 1
        return mask.astype(np.bool), class_ids.astype(np.int32)
```

Figure 9: Creating a class for preparing dataset

We will create dataset object for training and testing datasets and invoke prepare() method for them respectively (shown in Figure 10).

```
%%time
# prepare the training dataset
dataset_train = DetectorDataset(image_fps_train, image_annotations, ORIG_SIZE, ORIG_SIZE)
dataset_train.prepare()
```

```
CPU times: user 55.8 s, sys: 756 ms, total: 56.6 s
Wall time: 56.6 s
```

```
%%time
# prepare the validation dataset
dataset_val = DetectorDataset(image_fps_val, image_annotations, ORIG_SIZE, ORIG_SIZE)
dataset_val.prepare()
```

```
CPU times: user 18.2 s, sys: 210 ms, total: 18.4 s
Wall time: 18.4 s
```

Figure 10: Preparing training and testing dataset

We are using ImageAug library for data augmentation. Figure 11 below shows different data augmentations that were used in the project.

```
# Image augmentation
augmentation = iaa.Sequential([
    iaa.OneOf([ ## rotate
        iaa.Affine(rotate=0),
        iaa.Affine(rotate=90),
        iaa.Affine(rotate=180),
        iaa.Affine(rotate=270),
    ]),
    iaa.Fliplr(0.5),
    iaa.Flipud(0.5),
    iaa.OneOf([ ## brightness or contrast
        iaa.Multiply((0.9, 1.1)),
        iaa.ContrastNormalization((0.9, 1.1)),
    ]),
    iaa.OneOf([ ## blur or sharpen
        iaa.GaussianBlur(sigma=(0.0, 0.1)),
        iaa.Sharpen(alpha=(0.0, 0.1)),
    ]),
])
```

Figure 11: Data augmentation

We will specify the configurations for training our Mask R-CNN model. The config.py file which is cloned from github contains default configuration and hyperparameter values which can be manipulated using code mentioned in Figure 10) below.

```python
class DetectorConfig(Config):
    NAME = 'airbus'
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1
    NUM_CLASSES = 2  # background and ship classes
    IMAGE_MIN_DIM = 768
    IMAGE_MAX_DIM = 768
    RPN_ANCHOR_SCALES = (8, 16, 32, 64)
    DETECTION_NMS_THRESHOLD = 0.0
    STEPS_PER_EPOCH = 15 if debug else  500
    VALIDATION_STEPS = 10 if debug else 500

config = DetectorConfig()
config.display()
```

Figure 12: Data augmentation

In order to implement transfer learning, we have used COCO pre-trained weights while training. The coco weights file is also downloaded from Mask R-CNN's github repository. model.load_weights() function is used to load these COCO weights into our model.

```python
COCO_WEIGHTS_PATH = os.path.join(ROOT_DIR, "mask_rcnn_coco.h5")

model = modellib.MaskRCNN(mode='training', config=config, model_dir=ROOT_DIR)

# Exclude the last layers because they require a matching number of classes
model.load_weights(COCO_WEIGHTS_PATH, by_name=True, exclude=[
    "mrcnn_class_logits", "mrcnn_bbox_fc",
    "mrcnn_bbox", "mrcnn_mask"])
```

Figure 13: COCO transfer learning

After executing all activities mentioned above, we are ready to train our model. We will set LEARNING_RATE variable to 0.001 and train model heads for 2 epochs using code mentioned in Figure 14)

```python
%%time
model.train(dataset_train, dataset_val,
            learning_rate=LEARNING_RATE,
            epochs=2,
            layers='heads',
            augmentation=None)  ## not augmenting for first 2 epochs

history = model.keras_model.history.history
```

Figure 14: Training model heads

After training the model heads, we will apply some learning rate decay to our model and train its all layers for 50 epochs. Each epoch will maintain information about different losses experienced during training and testing and on the basis of these losses we will select the best epoch which has minimum val_loss for model evaluation. Figure 15) shows how our model executes during training.

```
Epoch 23/36
500/500 [==============================] - 362s 725ms/step - loss: 0.8952 - rpn_class_loss: 0.0059 - rpn_bbox_loss: 0.3797 - mrcnn_clas
Epoch 24/36
500/500 [==============================] - 308s 617ms/step - loss: 0.9570 - rpn_class_loss: 0.0055 - rpn_bbox_loss: 0.4029 - mrcnn_clas
Epoch 25/36
500/500 [==============================] - 311s 621ms/step - loss: 0.8892 - rpn_class_loss: 0.0057 - rpn_bbox_loss: 0.3638 - mrcnn_clas
Epoch 26/36
500/500 [==============================] - 310s 620ms/step - loss: 0.8434 - rpn_class_loss: 0.0054 - rpn_bbox_loss: 0.3429 - mrcnn_clas
Epoch 27/36
500/500 [==============================] - 313s 626ms/step - loss: 0.8720 - rpn_class_loss: 0.0061 - rpn_bbox_loss: 0.3747 - mrcnn_clas
```

Figure 15: Model execution

After selecting best epoch, we will recreate the model in inference mode and load the weights produced by best epoch. Refer the code in Figure 16) for the same.

```
class InferenceConfig(DetectorConfig):
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1

inference_config = InferenceConfig()

# Recreate the model in inference mode
model2 = modellib.MaskRCNN(mode='inference',
                           config=inference_config,
                           model_dir=ROOT_DIR)

# Load trained weights (fill in path to trained weights here)
model_path = "/content/drive/MyDrive/Ship/airbus20210810T0625/mask_rcnn_airbus_0049.h5"
print("Loading weights from ", model_path)
model2.load_weights(model_path, by_name=True)
```

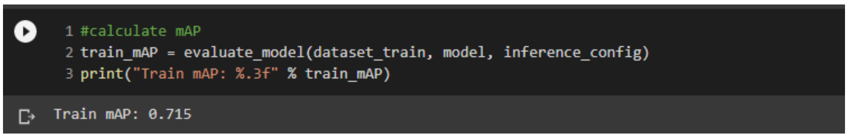
```
WARNING:tensorflow:From /content/drive/MyDrive/Ship/Mask_RCNN/mrcnn/model.py:720: The name tf.sets.set_intersection is

WARNING:tensorflow:From /content/drive/MyDrive/Ship/Mask_RCNN/mrcnn/model.py:722: The name tf.sparse_tensor_to_dense i

WARNING:tensorflow:From /content/drive/MyDrive/Ship/Mask_RCNN/mrcnn/model.py:772: to_float (from tensorflow.python.ops
Instructions for updating:
Use `tf.cast` instead.
Loading weights from  /content/drive/MyDrive/Ship/airbus20210810T0625/mask_rcnn_airbus_0049.h5
Re-starting from epoch 49
```

Figure 16: Model recreation in inference mode

Once the model is recreated in inference mode, we can evaluate its performance. We will use mean average precision (mAP) metrics to evaluate our model. a model is considered as a balanced model if its mAP is between 0.5 to 1.0. We can see from Figure 17) below that our model was able to achieve mAP of 0.715 which denotes that the model is balanced.



```
#calculate mAP
train_mAP = evaluate_model(dataset_train, model, inference_config)
print("Train mAP: %.3f" % train_mAP)
```

```
Train mAP: 0.715
```

Figure 17: Model Evaluation

The testing results can be visualized using visualize.display_instances() method defined in Mask R-CNN library. Figure 18) below shows visualizations for sample images.

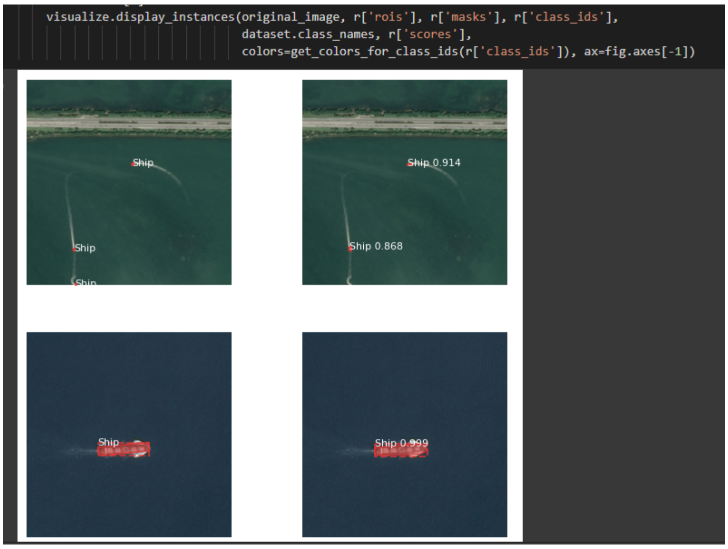


Figure 18: Visualization

In the last stage we can feed our validation dataset to the model which will predict whether or not there is a ship in given sar image. For that, we will define a function predict() which will make predictions on all images present in validation set and write down the results in submissions.csv file.

```
# Make predictions on test images, write out results in submission.csv
def predict(image_fps, filepath='submission.csv', min_conf=config.DETECTION_MIN_CONFIDENCE):
    # assume square image
    resize_factor = ORIG_SIZE / config.IMAGE_SHAPE[0]
    #resize_factor = ORIG_SIZE
    with open(filepath, 'w') as file:
        file.write("ImageId,EncodedPixels\n")
        for image_id in tqdm(image_fps):
            found = False
            if image_id not in test_names_nothing:
                image = imread(os.path.join(test_dicom_dir, image_id))
                # If grayscale. Convert to RGB for consistency.
                if len(image.shape) != 3 or image.shape[2] != 3:
                    image = np.stack((image,) * 3, -1)
                results = model.detect([image])
                r = results[0]
                assert( len(r['rois']) == len(r['class_ids']) == len(r['scores']) )
                if len(r['rois']) == 0:
                    pass  ## no ship
                else:
                    num_instances = len(r['rois'])
                    for i in range(num_instances):
                        if r['scores'][i] > min_conf:
                            file.write(image_id + "," + rle_encode(r['masks'][...,i]) + "\n")
                            found = True
            if not found:
                file.write(image_id + ",\n")  ## no ship

#Prediction
submission_fp = os.path.join(DATA_DIR, 'submission.csv')
predict(test_image_fps, filepath=submission_fp)
```

Figure 19: Making predictions on validation dataset

The scripts and functions mentioned in this document are all provided in the ICT solutions along with this report.

References

He, K., Gkioxari, G., Dollár, P. and Girshick, R. (2017). Mask r-cnn, *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988.