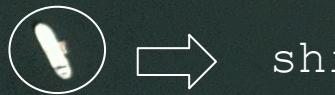
Identifying ships in satellite imagery

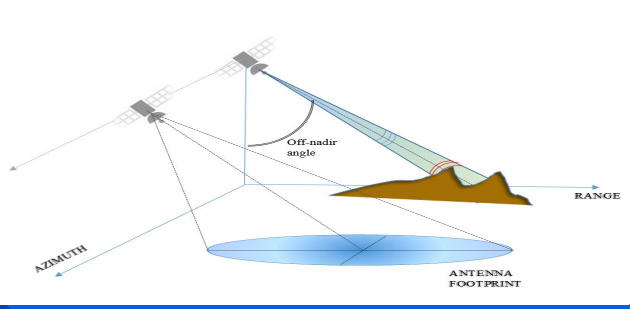
Big Data for Official Statistics
2019/2020

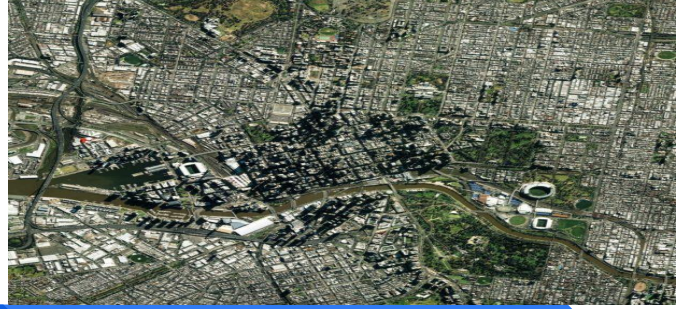
Valerio Antonini
1611556

Task:

Build an algorithm to automatically identify whether a remotely sensed is a ship or not







Images production

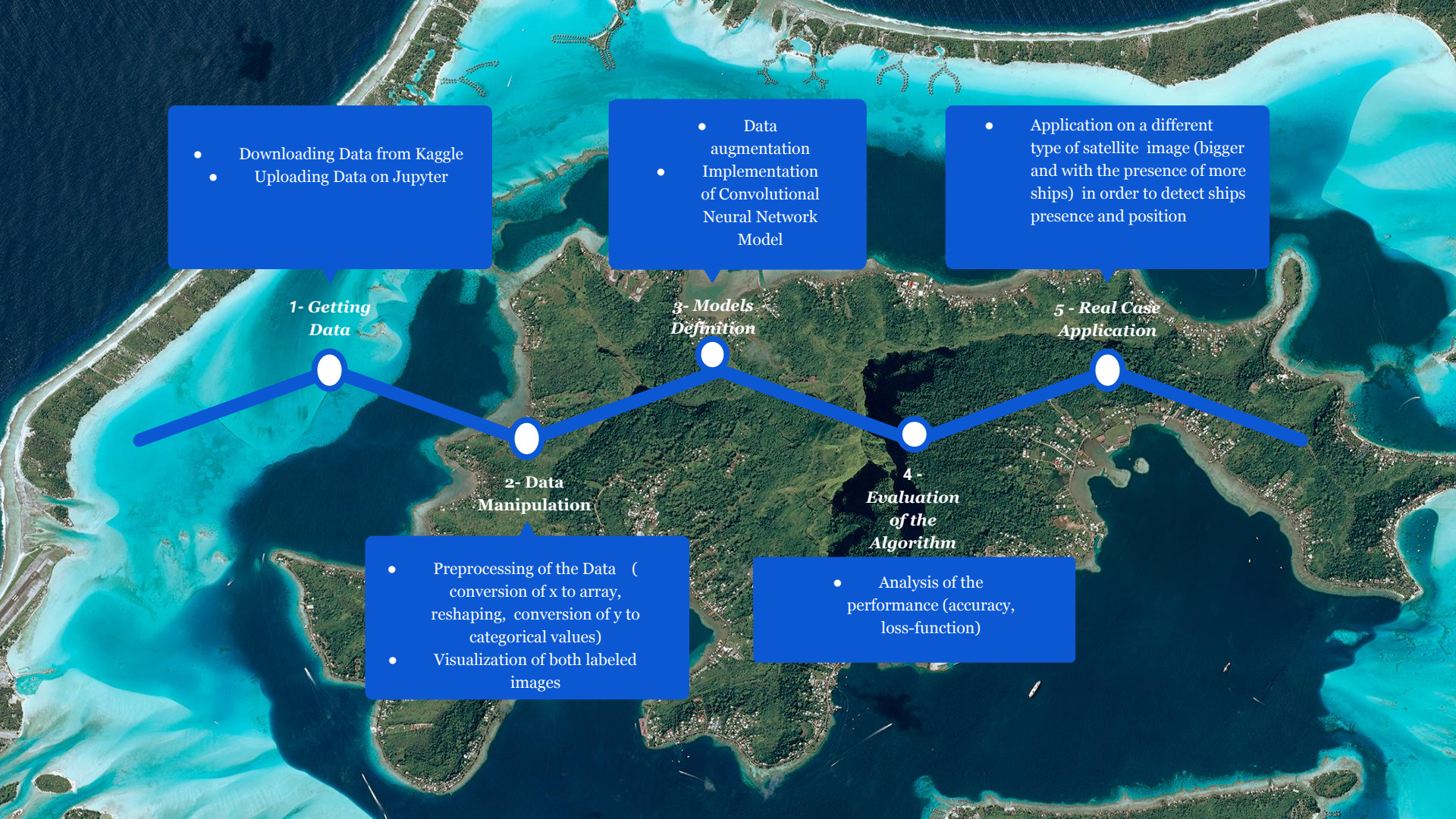
Synthetic Aperture Radar (SAR) imagery uses radio waves to image the Earth's surface. Unlike optical imagery, the wavelengths which the instruments use are not affected by the time of day or meteorological conditions, enabling imagery to be obtained day or night, with cloudy, or clear skies.

Images exploitation

Object detection in satellite image analysis is a fundamental problem which plays a significant role for different types of applications, such as detection of geological hazard, urban planning, Land use and cover mapping, environmental monitoring, updation of geographic information system and agriculture.

Deep Learning utilization

After the huge success of Deep learning methods in computer vision field they are currently being studied in the context of satellite imagery for different purposes like object identification, object tracking, object classification, semantic segmentation of aerial/satellite images.


1- Getting Data
Downloading Data from Kaggle
Uploading Data on Jupyter
2- Data Manipulation
Preprocessing of the Data (conversion of x to array, reshaping, conversion of y to categorical values)
Visualization of both labeled images
3- Models Definition
Data augmentation
Implementation of Convolutional Neural Network Model
4 - Evaluation of the Algorithm
Analysis of the performance (accuracy, loss-function)
5 - Real Case Application
Application on a different type of satellite image (bigger and with the presence of more ships) in order to detect ships presence and position

Data

The source

Dataset taken from Kaggle challenge: Ships on Satellite Imagery (source: Planet).
The dataset includes 4000 80x80 RGB images labeled with either a “ship” or “no-ship” classification

The “non-ship” class

The "no-ship" class includes 3000 images. Some of these do not include any portion of an ship, some are contain only a portion of an ship and the last third are images that have previously been mislabeled by machine learning models

The “ship” class

The "ship" class includes 1000 images. Ships of different sizes, orientations, and atmospheric collection conditions are included

The pixel value

The pixel value data for each 80x80 RGB image is stored as a list of 19200 integers within the data list. The first 6400 entries contain the red channel values, the next 6400 the green, and the final 6400 the blue

Data Manipulation

```python
def preprocessing(dataset):
    data = np.array(dataset['data']).astype('uint8')
    labels = np.array(dataset['labels']).astype('uint8')
    x = data.reshape([-1, 3, 80, 80]).transpose([0,2,3,1])
    x = x / 255.0
    y = to_categorical(labels, num_classes=2)
    return x,y,labels
```

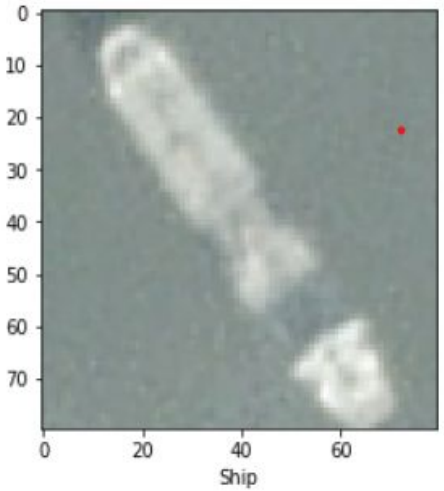
```python
data, labels, y = preprocessing(dataset)
```

```
Total number of images: 4000
Number of NoShip Images: 3000
Number of Ship Images: 1000
```

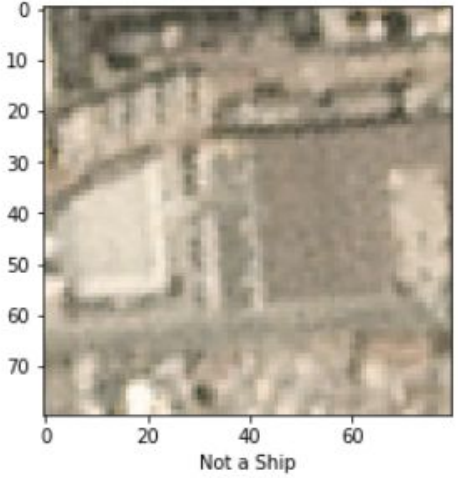
Data Visualization

```
def show_immage(position):
    index = random.choice(position)   #Select a random element from the list
    pixel_vals = data[index]
    im = pixel_vals.reshape((80,80,3))
    plt.imshow(im)
    plt.xlabel(classes[y[index]])
```

show_immage(ship)



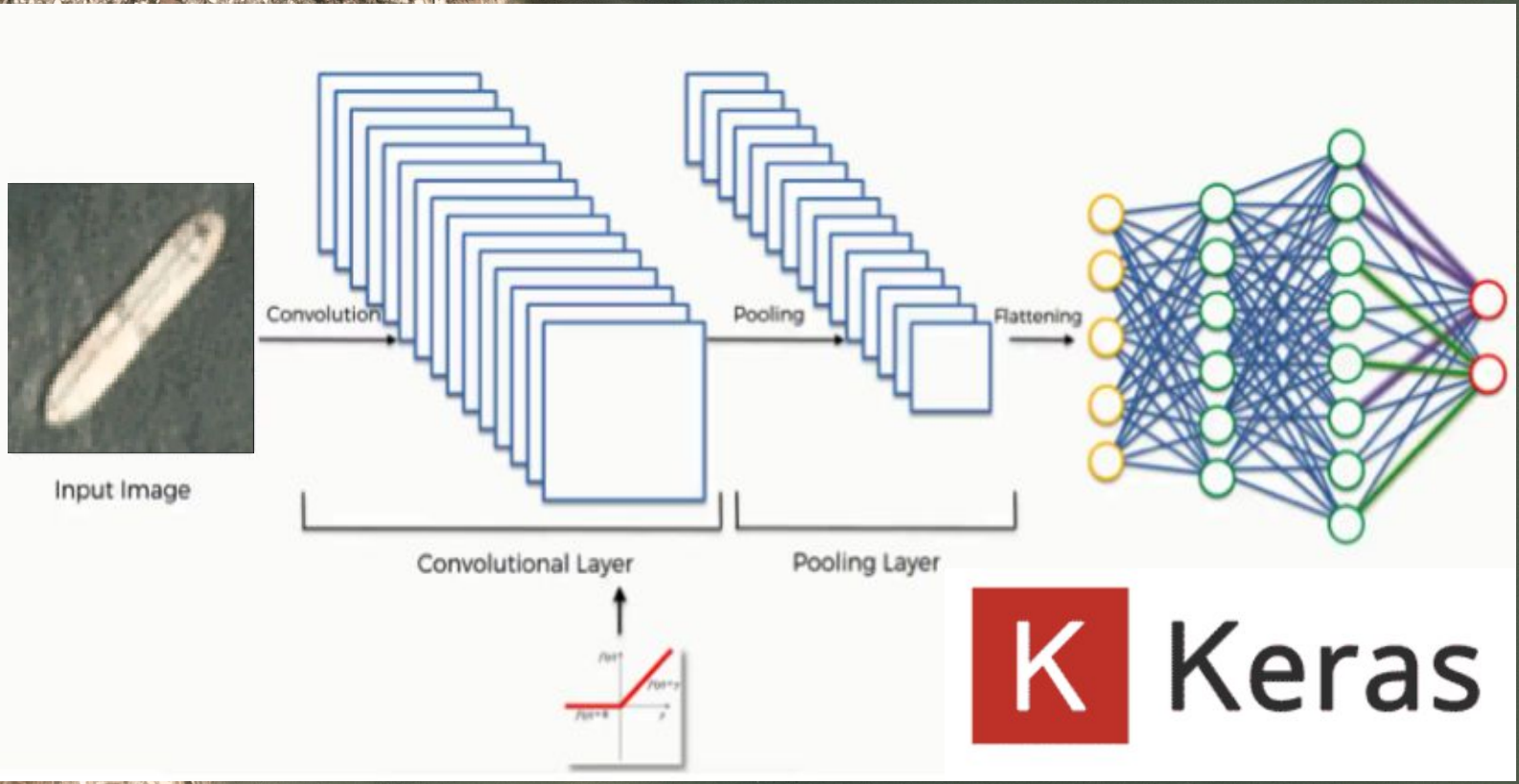
show_immage(no_ship)



Data Augmentation

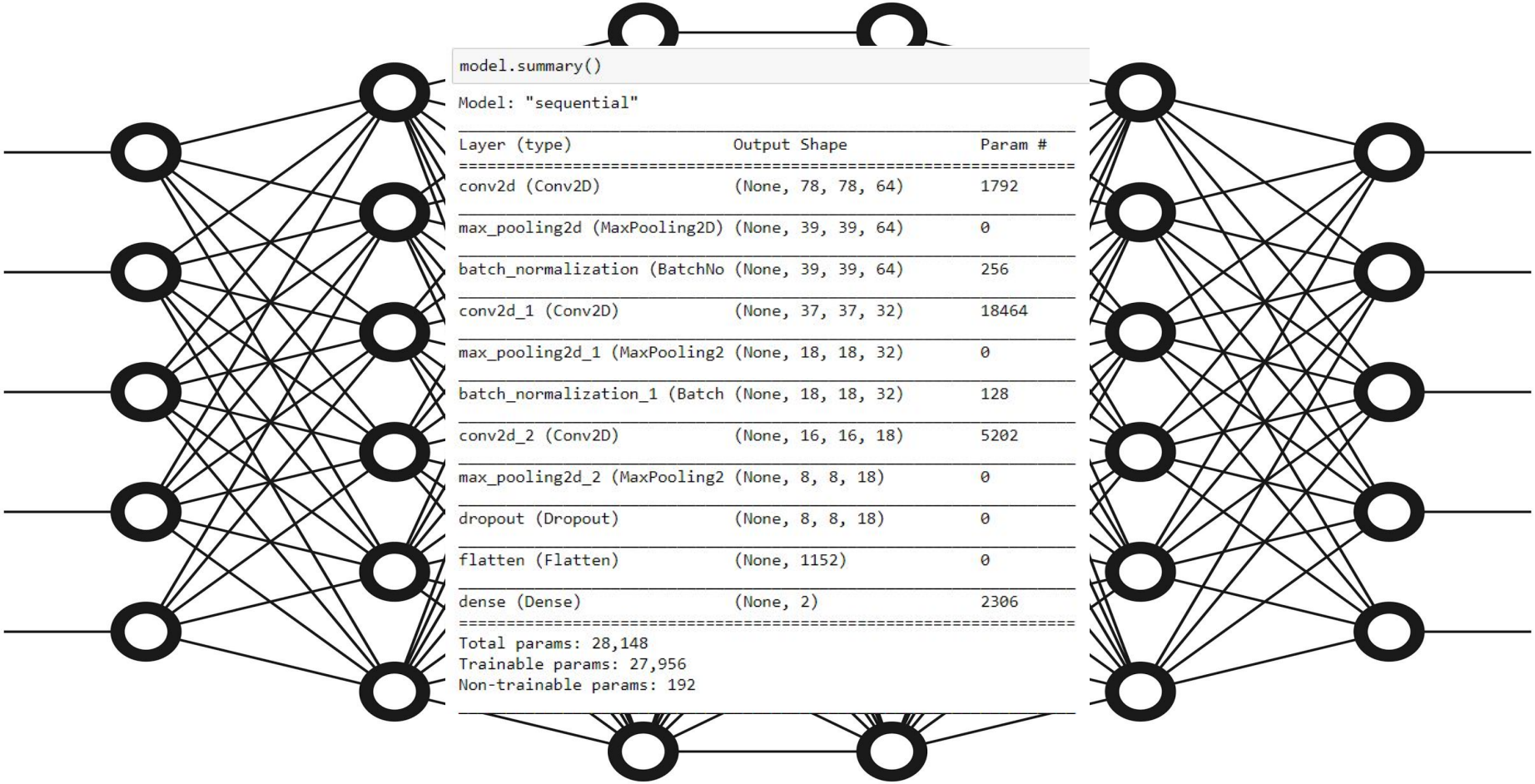
```
aug = ImageDataGenerator(shear_range=0.1,
                         width_shift_range=0.15,
                         height_shift_range=0.05,
                         zoom_range=0.3,
                         horizontal_flip=True,
                         vertical_flip=True,
                         fill_mode="nearest")
```

Convolutional Neural Network



Model Definition

```
model =  Sequential()

model.add(Conv2D(64, kernel_size=3, activation='relu', input_shape=(80,80,3)))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(BatchNormalization())


model.add(Conv2D(32, kernel_size=3, activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())

model.add(Conv2D(18, kernel_size=3, activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(2, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```



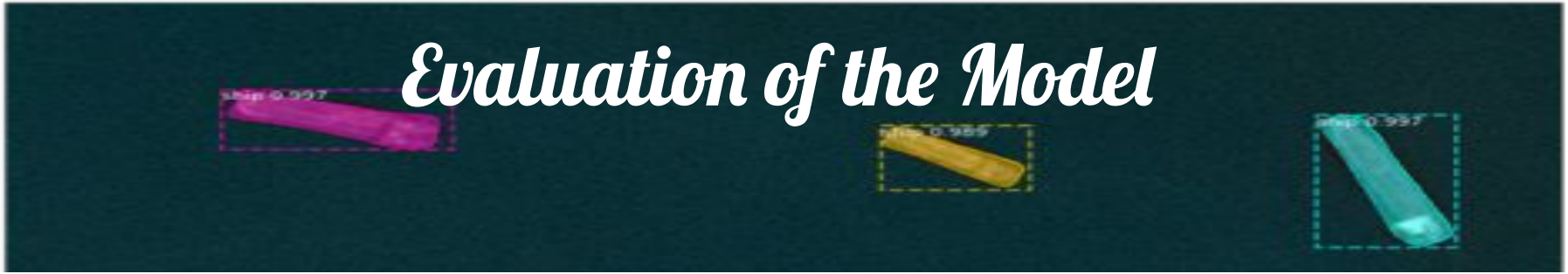
```
model.summary()
```

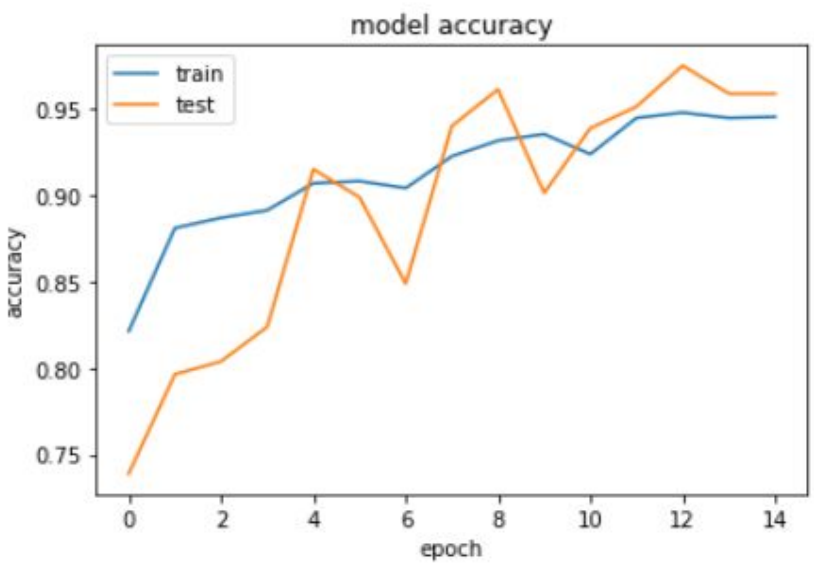
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 78, 78, 64)        1792
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 39, 39, 64)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 39, 39, 64)        256
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 37, 37, 32)        18464
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 18, 18, 32)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 18, 18, 32)        128
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 18)        5202
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 18)          0
_________________________________________________________________
dropout (Dropout)            (None, 8, 8, 18)          0
_________________________________________________________________
flatten (Flatten)            (None, 1152)              0
_________________________________________________________________
dense (Dense)                (None, 2)                 2306
=================================================================
Total params: 28,148
Trainable params: 27,956
Non-trainable params: 192
_________________________________________________________________
```

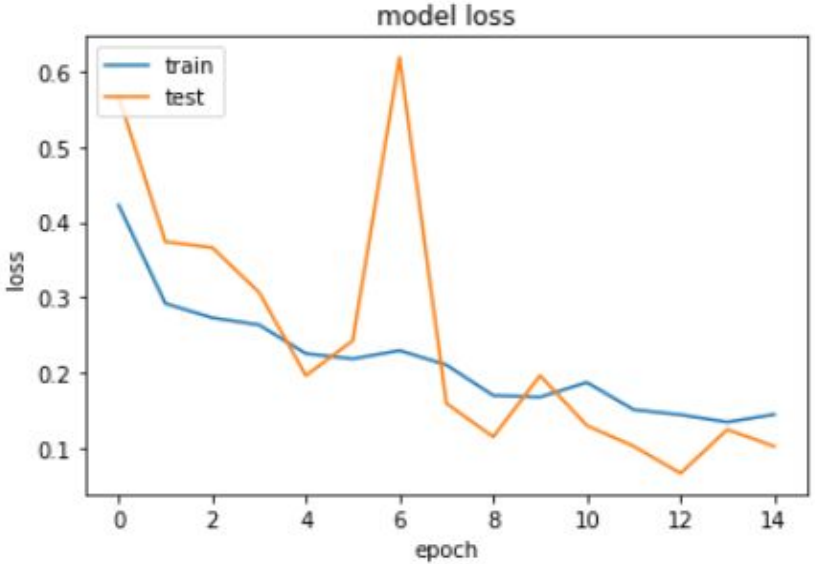
```
batch__size = 32
epochs_ = 15
history = model.fit_generator(aug.flow(X_train, y_train, batch_size = batch__size),
          validation_data = (X_test, y_test),
          steps_per_epoch = len(X_train) // batch__size,
          epochs = epochs_, verbose=2)
```

```
Epoch 1/15
100/100 - 64s - loss: 0.4224 - accuracy: 0.8213 - val_loss: 0.5659 - val_accuracy: 0.7387
Epoch 2/15
100/100 - 61s - loss: 0.2921 - accuracy: 0.8809 - val_loss: 0.3739 - val_accuracy: 0.7962
Epoch 3/15
100/100 - 63s - loss: 0.2732 - accuracy: 0.8869 - val_loss: 0.3662 - val_accuracy: 0.8037
Epoch 4/15
100/100 - 60s - loss: 0.2639 - accuracy: 0.8913 - val_loss: 0.3069 - val_accuracy: 0.8238
Epoch 5/15
100/100 - 60s - loss: 0.2258 - accuracy: 0.9069 - val_loss: 0.1967 - val_accuracy: 0.9150
Epoch 6/15
100/100 - 60s - loss: 0.2189 - accuracy: 0.9081 - val_loss: 0.2429 - val_accuracy: 0.8988
Epoch 7/15
100/100 - 60s - loss: 0.2299 - accuracy: 0.9041 - val_loss: 0.6182 - val_accuracy: 0.8487
Epoch 8/15
100/100 - 61s - loss: 0.2108 - accuracy: 0.9225 - val_loss: 0.1599 - val_accuracy: 0.9400
Epoch 9/15
100/100 - 60s - loss: 0.1704 - accuracy: 0.9316 - val_loss: 0.1156 - val_accuracy: 0.9613
Epoch 10/15
100/100 - 61s - loss: 0.1682 - accuracy: 0.9353 - val_loss: 0.1967 - val_accuracy: 0.9013
Epoch 11/15
100/100 - 61s - loss: 0.1876 - accuracy: 0.9237 - val_loss: 0.1305 - val_accuracy: 0.9388
Epoch 12/15
100/100 - 60s - loss: 0.1515 - accuracy: 0.9447 - val_loss: 0.1031 - val_accuracy: 0.9513
Epoch 13/15
100/100 - 61s - loss: 0.1449 - accuracy: 0.9478 - val_loss: 0.0673 - val_accuracy: 0.9750
Epoch 14/15
100/100 - 62s - loss: 0.1350 - accuracy: 0.9447 - val_loss: 0.1252 - val_accuracy: 0.9588
Epoch 15/15
100/100 - 61s - loss: 0.1452 - accuracy: 0.9453 - val_loss: 0.1030 - val_accuracy: 0.9588
```

Evaluation of the Model







Confusion Matrix

Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class

C.M. is a specific table layout that allows visualization of the performance of an algorithm

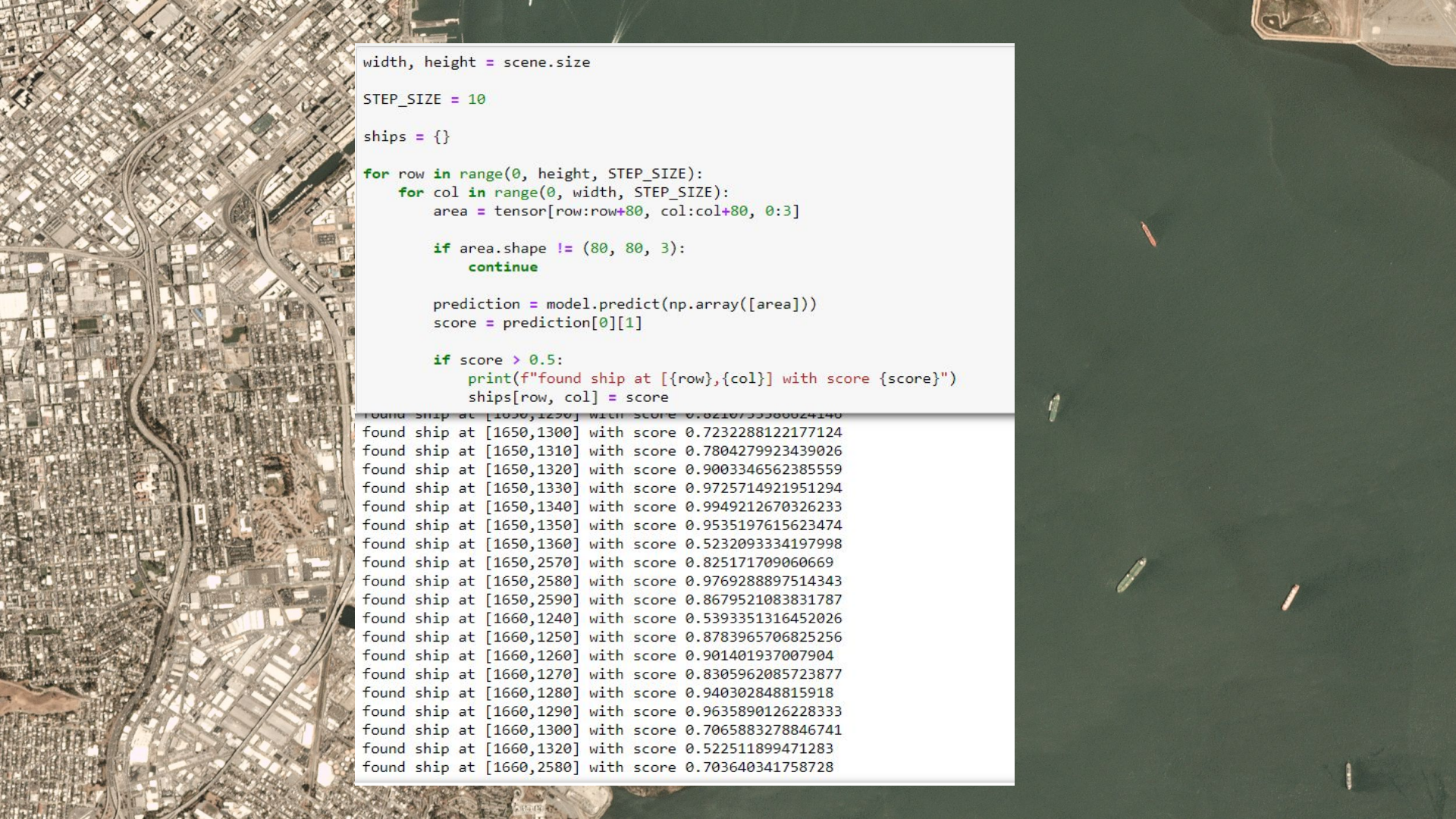
Confusion matrix

Real Case Application

1. Next step is to investigate a different image in order to detect the ships presence. The new image is bigger than the previous ones and represents San Francisco bay viewed from a satellite.

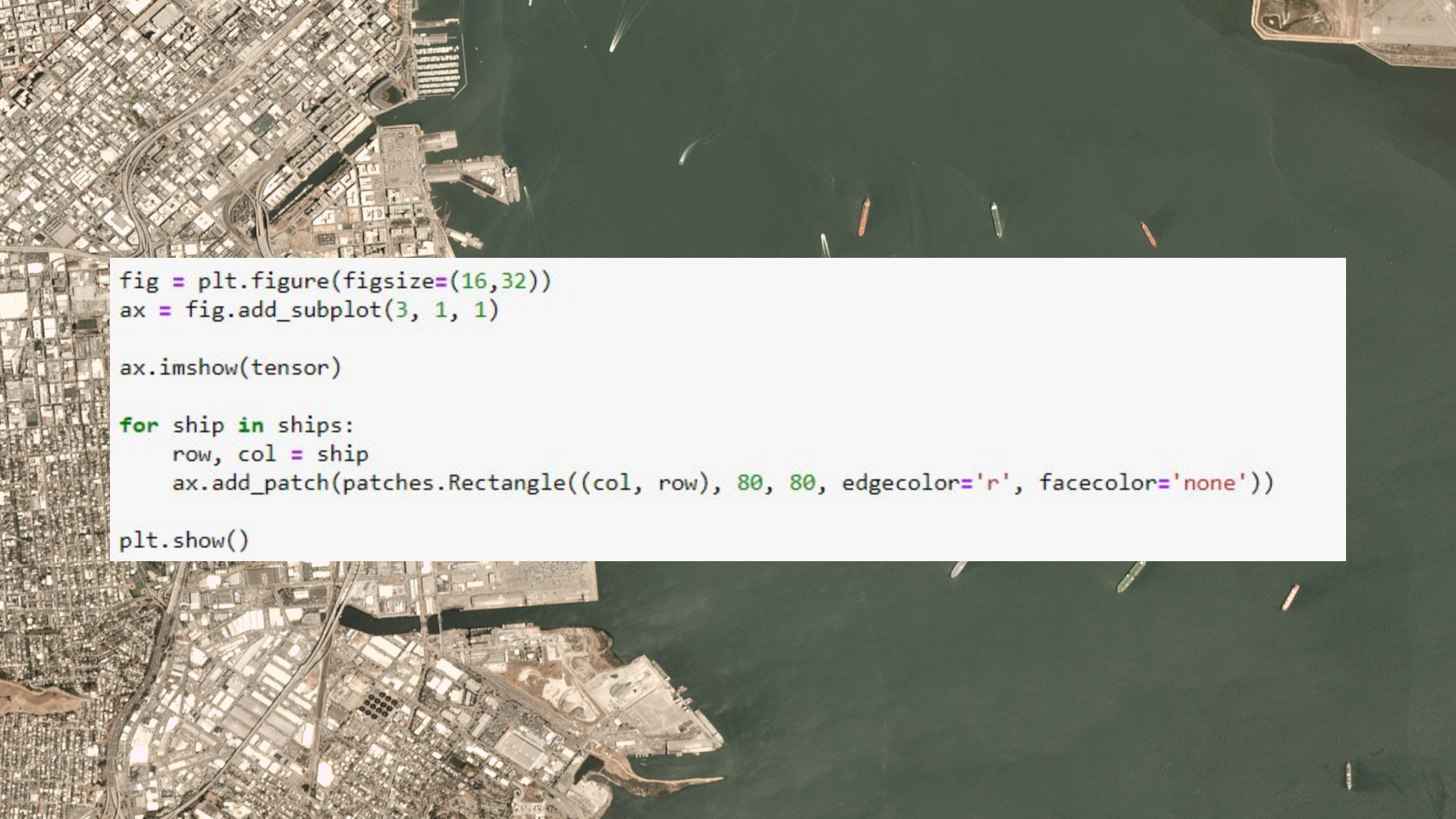
2. The algorithm's going to check a small window of pixel to detect the ships presence appling the previous convolutional neural network model.

3. Therefore it's going to move on the right and check window by window until cover the whole image. The last step is to show the whole image of San Francisco bay where the retrieval ships are highlighted by a rectangle.



```python
width, height = scene.size

STEP_SIZE = 10

ships = {}

for row in range(0, height, STEP_SIZE):
    for col in range(0, width, STEP_SIZE):
        area = tensor[row:row+80, col:col+80, 0:3]

        if area.shape != (80, 80, 3):
            continue

        prediction = model.predict(np.array([area]))
        score = prediction[0][1]

        if score > 0.5:
            print(f"found ship at [{row},{col}] with score {score}")
            ships[row, col] = score
```

```
found ship at [1650,1300] with score 0.7232288122177124
found ship at [1650,1310] with score 0.7804279923439026
found ship at [1650,1320] with score 0.9003346562385559
found ship at [1650,1330] with score 0.9725714921951294
found ship at [1650,1340] with score 0.9949212670326233
found ship at [1650,1350] with score 0.9535197615623474
found ship at [1650,1360] with score 0.5232093334197998
found ship at [1650,2570] with score 0.825171709060669
found ship at [1650,2580] with score 0.9769288897514343
found ship at [1650,2590] with score 0.8679521083831787
found ship at [1660,1240] with score 0.5393351316452026
found ship at [1660,1250] with score 0.8783965706825256
found ship at [1660,1260] with score 0.901401937007904
found ship at [1660,1270] with score 0.8305962085723877
found ship at [1660,1280] with score 0.940302848815918
found ship at [1660,1290] with score 0.9635890126228333
found ship at [1660,1300] with score 0.7065883278846741
found ship at [1660,1320] with score 0.522511899471283
found ship at [1660,2580] with score 0.703640341758728
```

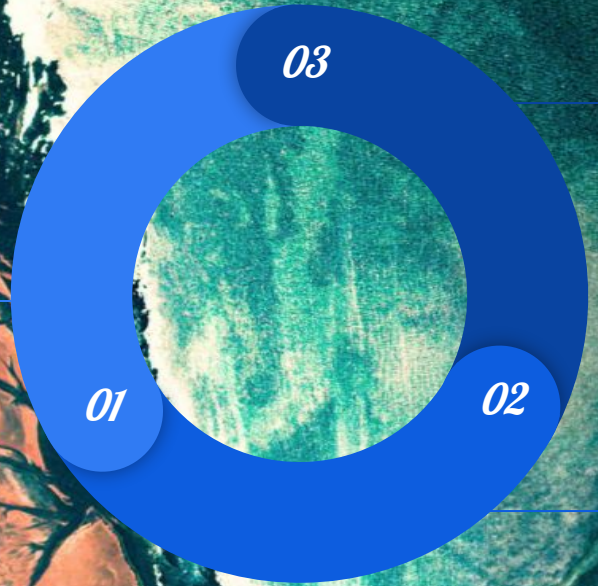


```
fig = plt.figure(figsize=(16,32))
ax = fig.add_subplot(3, 1, 1)

ax.imshow(tensor)

for ship in ships:
    row, col = ship
    ax.add_patch(patches.Rectangle((col, row), 80, 80, edgecolor='r', facecolor='none'))

plt.show()
```


0
200
400
600
800
1000
1200
1400
1600
0
500
1000
1500
2000
2500

Project purpose

Ships detection from satellite is very important for several reasons: intelligence, commercial, ships retrieval..



Different applications

In general the use of satellite images in combination with deep learning could be very helpful for several applications. A second set of tasks is of change detection, where the goal is to identify regions which have changed when contrasting two satellite images. This has applications in determining extent of damage during natural disasters, understanding the growth of cities, and also determining deforestation

New tasks

In the future, it might be possible to even apply these techniques to live drones or airships. The industry is on the rise, and pretty soon the satellite imageries treatment will reach even more important peaks.


Thanks for your attention!
Identifying ships in satellite imagery
Big Data for Official Statistics
Valerio Antonini
1611556