

# Project1 AUV | Vortex

📅 Deadline	@Oct 9, 2020
📄 Requirements	<a href="https://docs.google.com/document/d/1W1vqOUQJrSRdAA-CHnTKE792Ehe4Xv3YyK2AzK5mSYc/edit?usp=sharing">https://docs.google.com/document/d/1W1vqOUQJrSRdAA-CHnTKE792Ehe4Xv3YyK2AzK5mSYc/edit?usp=sharing</a>
📌 Status	Completed

**Ahmed Tarek | Heba AbdelQader**

| We Made a about 7 Meetings to Complete Our Project

## Tasks:

- ✓ Task 1
- ✓ Task 1 Bonus
- ✓ Task 2
- ✓ Task 2 Bonus
- ✓ Creating The Report

## Task Number 1 :

| We Started With Making a Meeting To Discuss Ideas about the 1st Task - Most of the Ideas and Try-Error and Searching Done in an Online Meetings

## Ahmed Tarek

- 1) I Shared Ideas of Filters With Heba and She Started to Code it
- 2) I Codded The Drawing Part For The 2 Rectangles with The Idea Of Drawing
- 3) Finally I developed The Equation for the Rectangles to detect which Gate Wider Than The Other
- 4) I Developed The Code To make the Code Work With Videos

## Heba AbelQader

- 1) I tried different Values in Brightness, Contrast and Sharpness on Enhancement
- 1) I Codded The Filters Part To Reach The Edges of the Gate
- 2) I take the Code of Rectangle with Idea to Develop on it in Contours Detecting to be added to The Filtration Code
- 3) I Made The Final Touch to Make The Rectangle to be Drawn From Different Angles

## Technologies and Algorithms we Used :

- PILLOW >> Image Enhance, Image Filters , Image
- OpenCV >> Median Blur, Thresholding, Canny edges and dilate , Finding Contours
- Drawing Rectangle >> Using (x,y), Width and length to draw a whole Rectangle with Finding the Common sides for 2 Rectangles
- euclidean Equation For Finding the Distance between points to know which gate is Wider than the other

## Final Output

### Detecting Gate (Narrow|Wide) Code

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
from PIL import Image, ImageFilter , ImageEnhance
import math

factor = 1.9 #Brightness Factor
img = cv2.imread('input.jpg') #Reading The Image #path of The Image
im = Image.fromarray(np.uint8(img)) #Passing ndarray to Image.fromarray() to obtain PIL.Image
br_enhancer = ImageEnhance.Brightness(im) #Adjusting Brightness
im_output = br_enhancer.enhance(factor)#Adjusting Brightness
co_enhancer = ImageEnhance.Contrast(im_output)# enhancer for adjusting contrast
factor = 1.5 #increase contrast
im_output = co_enhancer.enhance(factor)# enhancer for adjusting sharpening
sh_enhancer = ImageEnhance.Sharpness(im_output) #adjusting sharpening
factor = 5 #increase sharpness
im_output = sh_enhancer.enhance(factor)#increase sharpness
processed_image = np.asarray(im_output)
r, g, b = cv2.split(processed_image) # splitting the image into different channels #b is the best choice for underwater images
_, img_binary = cv2.threshold(b,150, 255, cv2.THRESH_BINARY) #converting the pixels with values between 150 to 255 to white color
new_image = cv2.medianBlur(img_binary, 3)# using median filter
edges = cv2.Canny(new_image,10,10) # applying canny edges
edges = cv2.dilate(edges, None)
contours, hierarchy = cv2.findContours(edges,cv2.RETR_TREE,cv2.CHAIN_APPROX_NONE) #Detection Contors and hierarchy
circles = []
min_x = 0
min_y = 0
max_x = 0
max_y = 0

for iteration, contour in enumerate(contours): # Find bounding rectangles
    area = cv2.contourArea(contour)
    if area > 50 and area < 999999999999:
        x,y,w,h = cv2.boundingRect(contour)

        if (min_x == 0 and min_y == 0 and max_x == 0 and max_y == 0 ):
            min_x = x
            min_y = y
            max_x = x + w
            max_y = y + h
        if min_x > x:
            min_x = x
        if min_y > y:
            min_y = y
        if max_x < ( x + w ):
            max_x = x + w
        if max_y < ( y + h ):
            max_y = y + h

    mid_min_y = min_y
    mid_max_y = max_y
    width_thrid = (max_x - min_x)/3
    idx1 = min_x + width_thrid
    idx2 = max_x - width_thrid
    mid_min_x = min_x
    mid_max_x = max_x
    for iteration, contour in enumerate(contours):# Find bounding rectangles
        x,y,w,h = cv2.boundingRect(contour)
        if x > idx1 and x < idx2:
            if mid_min_y < y:
                mid_min_y = y
                mid_min_x = x
            if mid_max_y > ( y + h ):
                mid_max_y = y + h
                mid_max_x = x
    cv2.rectangle(img,(int(min_x),int(min_y)),(int(mid_min_x),int(max(max_y,mid_max_y))), (255,0,0),20)
```

```

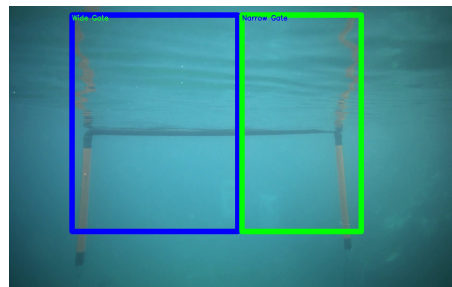
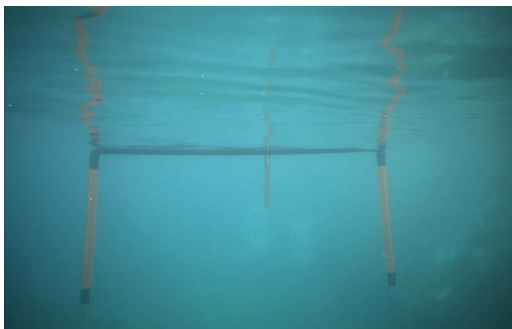
p1=[int(min_x),int(min_y)]
p2=[mid_max_x,mid_max_y]
gate1_width = math.sqrt( ((p1[0]-p2[0])**2)+((p1[1]-p2[1])**2) )
p1=[mid_min_x,mid_min_y]
p2=[int(max_x),int(max_y)]
gate2_width = math.sqrt( ((p1[0]-p2[0])**2)+((p1[1]-p2[1])**2) )
if (gate1_width > gate2_width):
    cv2.putText(img, 'Wide Gate', (int(min_x),int(min_y)+20), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (36,255,12), 2)
    cv2.rectangle(img,(int(mid_max_x),int(min(mid_min_y,min_y))), (int(max_x),int(max_y)), (0,255,0),20)
    cv2.putText(img, 'Narrow Gate', (int(mid_max_x),int(min(mid_min_y,min_y)+20)), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255,0,0), 2)
else:
    cv2.putText(img, 'Narrow Gate', (int(min_x),int(min_y)+20), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (36,255,12), 2)
    cv2.rectangle(img,(int(mid_max_x),int(min(mid_min_y,min_y))), (int(max_x),int(max_y)), (0,255,0),20)
    cv2.putText(img, 'Wide Gate', (int(mid_max_x),int(min(mid_min_y,min_y)+20)), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255,0,0), 2)

plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(edges, cmap = 'gray'),plt.title('contour')
plt.xticks([], plt.yticks([]))
plt.subplot(122), plt.imshow(img),plt.title('Detected Gate')
plt.xticks([], plt.yticks([]))
plt.show()
cv2.imwrite("Detected_Gates.jpg",img)

print("You got an output For Detected_Gate as an OUTPUT 'Detected_Gate.jpg' ")
#if it's Working on IDE
#cv2.imshow("Edges",edges)
#cv2.imshow("Detected Gates",img)
#cv2.waitKey()

```

## Output



## Video Processing Code

```

import numpy as np
import cv2
from matplotlib import pyplot as plt
from PIL import Image, ImageFilter, ImageEnhance
import math
cap = cv2.VideoCapture('tues_6.mp4')

# Get the Default resolutions
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))

# Define the codec and filename.

while (cap.isOpened()):
    ret, img = cap.read()
    factor = 1.9 #Brightness Factor
    im = Image.fromarray(np.uint8(img)) #Passing ndarray to Image.fromarray() to obtain PIL.Image
    br_enhancer = ImageEnhance.Brightness(im) #Adjusting Brightness
    im_output = br_enhancer.enhance(factor)#Adjusting Brightness
    co_enhancer = ImageEnhance.Contrast(im_output)# enhancer for adjusting contrast
    factor = 1.5 #increase contrast
    im_output = co_enhancer.enhance(factor)# enhancer for adjusting sharpening
    sh_enhancer = ImageEnhance.Sharpness(im_output) #adjusting sharpening
    factor = 5 #increase sharpness
    im_output = sh_enhancer.enhance(factor)#increase sharpness
    processed_image = np.asarray(im_output)
    r, g, b = cv2.split(processed_image) # splitting the image into different channels #b is the best choice for underwater im
    _, img_binary = cv2.threshold(b,150, 255, cv2.THRESH_BINARY) #converting the pixels with values between 150 to 255 to white
    new_image = cv2.medianBlur(img_binary, 3)# using median filter
    edges = cv2.Canny(new_image,10,10) # applying canny edges

```

```

edges = cv2.dilate(edges, None)
contours, hierarchy = cv2.findContours(edges,cv2.RETR_TREE,cv2.CHAIN_APPROX_NONE) #Detection Contors and hierarchy
circles = []
min_x = 0
min_y = 0
max_x = 0
max_y = 0
for iteration, contour in enumerate(contours): # Find bounding rectangles
    area = cv2.contourArea(contour)
    if area > 50 and area < 99999999999:
        x,y,w,h = cv2.boundingRect(contour)

        if (min_x == 0 and min_y == 0 and max_x == 0 and max_y == 0 ):
            min_x = x
            min_y = y
            max_x = x + w
            max_y = y + h
        if min_x > x:
            min_x = x
        if min_y > y:
            min_y = y
        if max_x < ( x + w ):
            max_x = x + w
        if max_y < ( y + h ):
            max_y = y + h

    mid_min_y = min_y
    mid_max_y = max_y
    width_thrid = (max_x - min_x)/3
    idx1 = min_x + width_thrid
    idx2 = max_x - width_thrid
    mid_min_x = min_x
    mid_max_x = max_x
    for iteration, contour in enumerate(contours):# Find bounding rectangles
        x,y,w,h = cv2.boundingRect(contour)
        if x > idx1 and x < idx2:
            if mid_min_y < y:
                mid_min_y = y
                mid_min_x = x
            if mid_max_y > ( y + h ):
                mid_max_y = y + h
                mid_max_x = x
    cv2.rectangle(img, (int(min_x),int(min_y)),(int(mid_min_x),int(max(max_y,mid_max_y))), (255,0,0),20)

    p1=[int(min_x),int(min_y)]
    p2=[mid_max_x,mid_max_y]
    gate1_width = math.sqrt( ((p1[0]-p2[0])**2)+((p1[1]-p2[1])**2) )
    p1=[mid_min_x,mid_min_y]
    p2=[int(max_x),int(max_y)]
    gate2_width = math.sqrt( ((p1[0]-p2[0])**2)+((p1[1]-p2[1])**2) )
    if (gate1_width > gate2_width):
        cv2.putText(img, 'Wide Gate', (int(min_x),int(min_y)+20), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (36,255,12), 2)
        cv2.rectangle(img, (int(mid_max_x),int(min(mid_min_y,min_y))), (int(max_x),int(max_y)), (0,255,0),20)
        cv2.putText(img, 'Narrow Gate', (int(mid_max_x),int(min(mid_min_y,min_y)+20)), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255,0,0),2)
    else:
        cv2.putText(img, 'Narrow Gate', (int(min_x),int(min_y)+20), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (36,255,12), 2)
        cv2.rectangle(img, (int(mid_max_x),int(min(mid_min_y,min_y))), (int(max_x),int(max_y)), (0,255,0),20)
        cv2.putText(img, 'Wide Gate', (int(mid_max_x),int(min(mid_min_y,min_y)+20)), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255,0,0),2)
    cv2.imshow("Detected Gates",img)
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()

```

## You Can Check The Google Colab

Google Colaboratory

 <https://colab.research.google.com/drive/1UvYT5tVXMhT5y8IAhbwsc4CHpj4Rd1Xp?usp=sharing>



## Task Number 2 :

We Started With Making a Meeting To Discuss Ideas about the 1st Task - Most of the Ideas and Try-Error and Searching Done in an Online Meetings

## Ahmed Tarek

A) I started To Write The Code For The Model I Wrote it with a different dataset Waiting For Heba to Collect The Data to be Distributed to Training- Validation - Testing

B) We Got an Error While Using it with Our Data So I started to Add Layers and to test

C) I Searched about Percentage of Distributions till I reached the Best >>

64% For Training - 16% For Validation - 20% For Testing

D) I made The Full Documentation Folder

## Heba AbelQader

A) I started to Collect Data from Different Videos and Images and Distributing it to Training - Validation - Testing

B) After Testing Ahmed's Code We Found that we have to add Images so I reached around 5000 Pictures

C) I Edited layers with Changing in Epoch with Ahmed and try and error till I reached The best testing output

D) Deciding the best functions to use (loss, optimizer, activation)

## Technolgies and Algorithms we Used :

We Developed a Model and then we got Problems in the Output and We decided the following functions for our problem:

best loss function is "categorical\_crossentropy" for multi-class classification

best optimizer was "rmsprop"

best activation function for input and hidden layers was Relu

best activation function for output layer was "softmax" function as it consists of multiple sigmoids and suitable for multi class classification

## Final Output

### Neural Networks Detecting Different Shapes UnderWater

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import RMSprop
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import cv2
import os

#img=image.load_img("/content/drive/My Drive/MyDataset/training/bat/bat-201.jpg")#test image path from training
#plt.imshow(img)

#cv2.imread("/content/drive/My Drive/MyDataset/training/bat/bat-201.jpg").shape#test image path from training

train = ImageDataGenerator(rescale= 1/255)
validation = ImageDataGenerator(rescale=1/255)

train_dataset = train.flow_from_directory("/content/drive/My Drive/MyDataset/training", #training folder end with /
                                         target_size= (500,600))
validation_dataset = train.flow_from_directory("/content/drive/My Drive/MyDataset/validation",#validation folder end with /
                                              target_size= (500,600))

model = tf.keras.Sequential([tf.keras.layers.Conv2D(8, (3,3),activation="relu",input_shape=(500,600,3)),
                             tf.keras.layers.MaxPool2D(2,2),
                             #
```

```

        tf.keras.layers.Conv2D(16, (3, 3), activation="relu"),
        tf.keras.layers.MaxPool2D(2, 2),
        #
        tf.keras.layers.Conv2D(32, (3, 3), activation="relu"),
        tf.keras.layers.MaxPool2D(2, 2),
        ##

        tf.keras.layers.Conv2D(64, (3, 3), activation="relu"),
        tf.keras.layers.MaxPool2D(2, 2),
        tf.keras.layers.Conv2D(128, (3, 3), activation="relu"),
        tf.keras.layers.MaxPool2D(2, 2),

        tf.keras.layers.Conv2D(256, (3, 3), activation="relu"),
        tf.keras.layers.MaxPool2D(2, 2),

        ##

        tf.keras.layers.Flatten(),
        ##
        tf.keras.layers.Dense(256, activation="relu"),
        ##
        tf.keras.layers.Dense(8, activation="softmax")
    ])

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

model_fit = model.fit(train_dataset,
                      steps_per_epoch=100,
                      epochs=8,
                      validation_data=validation_dataset)

train_dataset.classes

validation_dataset.class_indices

dir_path = "/content/drive/My Drive/MyDataset/testing" #test path end with /
for i in os.listdir(dir_path):
    img = image.load_img(dir_path+"/"+ i, target_size=(500, 600))
    plt.imshow(img)
    plt.show()

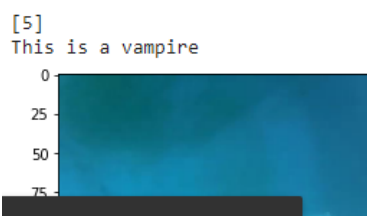
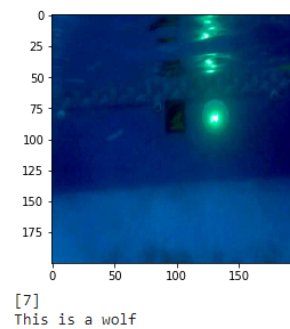
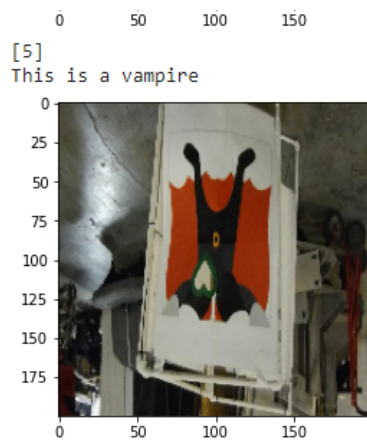
X=image.img_to_array(img)
X=np.expand_dims(X,axis=0)
images=np.vstack([X])
val = model.predict_classes(images)
print(val)
if val == 0: #conditions
    print("This is a bat")
elif val == 1:
    print("This is a draugar")
elif val == 2:
    print("This is a garlic")
elif val == 3:
    print("This is a gate")
elif val == 4:
    print("This is a sunlight")
elif val == 5:
    print("This is a vampire")
elif val == 6:
    print("This is a vetalas")
elif val == 7 :
    print("This is a wolf")

```

## Output

```
model_fit = model.fit(train_dataset,
                      steps_per_epoch=100,
                      epochs=8,
                      validation_data=validation_dataset)

... Epoch 1/8
100/100 [=====] - 1689s 17s/step - loss: 1.5847 - accuracy: 0.4662 - val_loss: 2.2174 - val_accuracy: 0.3316
Epoch 2/8
100/100 [=====] - 687s 7s/step - loss: 0.8787 - accuracy: 0.7347 - val_loss: 2.6510 - val_accuracy: 0.4069
Epoch 3/8
100/100 [=====] - 679s 7s/step - loss: 0.4148 - accuracy: 0.8705 - val_loss: 1.8742 - val_accuracy: 0.5839
Epoch 4/8
100/100 [=====] - 684s 7s/step - loss: 0.2729 - accuracy: 0.9220 - val_loss: 2.2019 - val_accuracy: 0.4069
Epoch 5/8
100/100 [=====] - 677s 7s/step - loss: 0.2723 - accuracy: 0.9327 - val_loss: 3.1868 - val_accuracy: 0.4456
Epoch 6/8
100/100 [=====] - 675s 7s/step - loss: 0.1257 - accuracy: 0.9594 - val_loss: 2.7872 - val_accuracy: 0.5514
Epoch 7/8
9/100 [=>.....] - ETA: 8:11 - loss: 0.2606 - accuracy: 0.9132
```



## You Can Check The Google Colab

Google Colaboratory


 <https://colab.research.google.com/drive/1hc4vVEG0z00GXoOWIraAQsx6FDNVoE0>

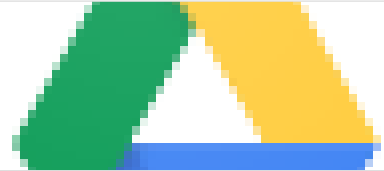


## Project Dataset

### Dataset >

MyDataset - Google Drive

 [https://drive.google.com/drive/folders/1xhyagzZ8x4QJqun7QgjBokSnKIBRPnn?fbclid=IwAR0ZTVg2G2tclrZ8wtF8JAHWcl2MFa\\_xHb6MQIOV6BOFDUB0Qm3P9lphUwk](https://drive.google.com/drive/folders/1xhyagzZ8x4QJqun7QgjBokSnKIBRPnn?fbclid=IwAR0ZTVg2G2tclrZ8wtF8JAHWcl2MFa_xHb6MQIOV6BOFDUB0Qm3P9lphUwk)



## Ahmed's Review

Working in this Project is Different as it's an Online Team Working I didn't imagine that I will Learn all of that things with All of That knowledge actually working with Heba is Awesome as She is available to work most of times only telling everybody are You ready ? Let's Go we Got above 7 Meetings Actually I enjoyed the Output

## Heba's Review

I really enjoyed working on these projects along with Ahmed. I am happy with our trials and results. It was a good chance for learning and better understanding for image processing and neural networks techniques. Projects were the best way to gain the practical experience of the theories. I came through lots of references and links which added to my knowledge.