Iceberg Detection In Satelite Images

Project Report

Department of Computer Science and Engineering

Chalapathi institute of engineering and technology

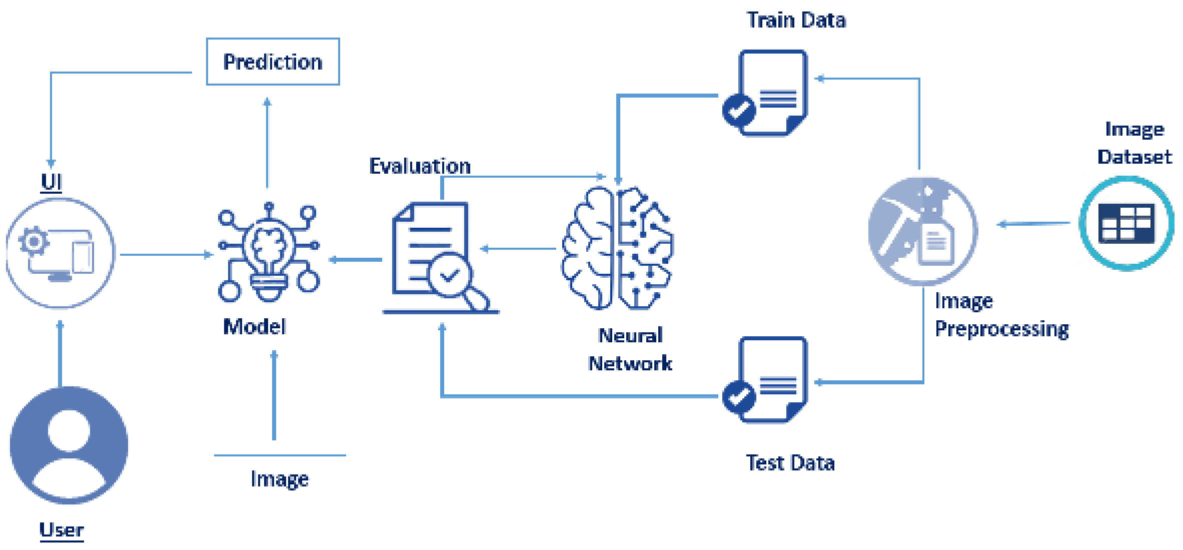
TEAM MEMBERS

1.Gonuguntla Sravani
2.Kamireddy Anitha
3.Gangavarapu Meghana
4.Gogula Madhuri

Introduction

Icebergs present serious hazards for ship navigation and offshore installations. Consequently, there is a large interest to localize them in timely and over vast areas. Because of their independence of cloud cover and daylight, satellite Synthetic Aperture Radar (SAR) images are among the preferred data sources for operational ice conditions and iceberg occurrences. The image spatial resolution mostly used for iceberg monitoring varies between a few and 100 m. Processed SAR data are characterized by speckle noise, which causes a grainy appearance of the images making the identification of icebergs extremely difficult. The methods of satellite monitoring of dangerous ice formations, like icebergs in the Arctic seas represent a threat to the safety of navigation and economic activity on the Arctic shelf.
The main aim of this project is to build a model that automatically identifies whether a remotely sensed target is an iceberg or not. Often times an iceberg is wrongly classified as a ship. The algorithm had to be extremely accurate because lives and billions of dollars in energy infrastructure are at stake.

Architecture:



The Challenge

Build an model to automatically identify whether a remotely sensed target is an iceberg or not.Often times an icebergs is wrongly classified as a ship.To avoid that confusion we are training model with ship dataset and Iceberg dataset.

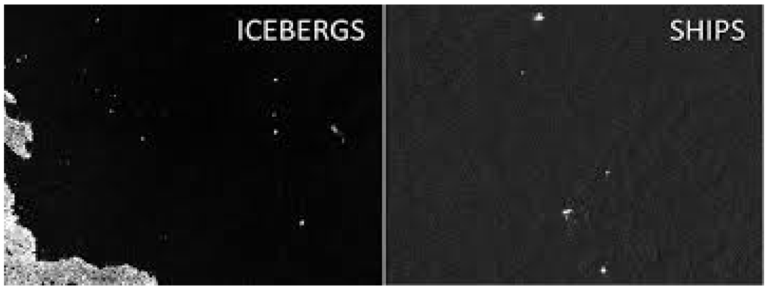
Environment and tools

1.scikit-learn
2.numpy
3.keras
4.pandas
5.matplotlib

Datasets

we are taking the two datasets

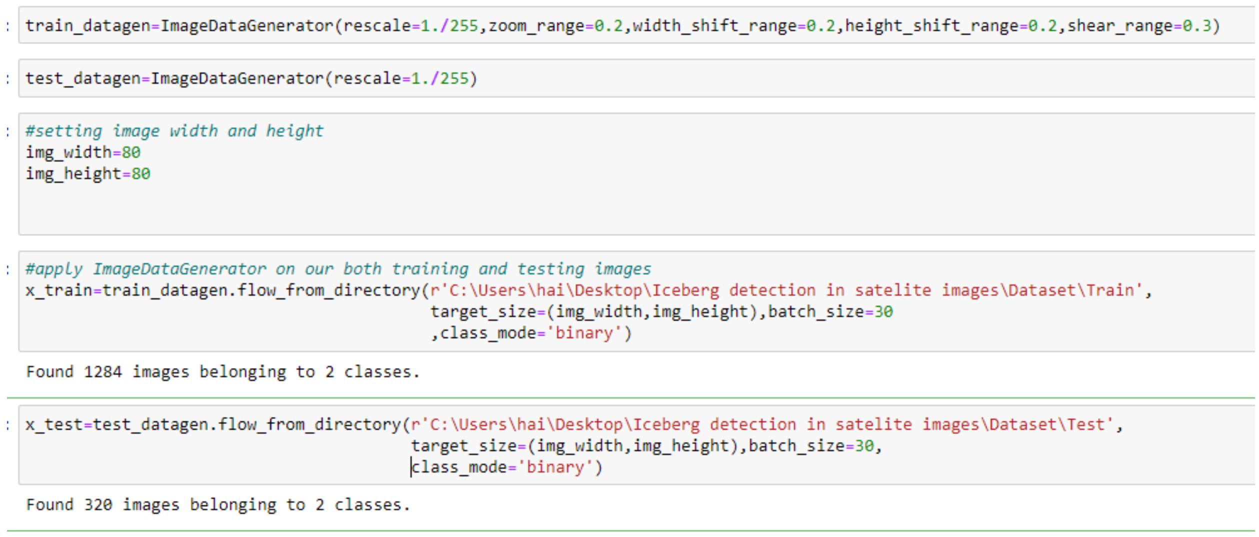
1.Iceberg
2.ship



Training and testing the model

Before training the model we have to preprocess the images.For that we use convultion 2D and Maxpooling.Next load the Training images dataset and Test images dataset.

Then defined training and test set data.we split the data into 3:1 that is 75% as training and the rest as test set.



```
train_datagen=ImageDataGenerator(rescale=1./255,zoom_range=0.2,width_shift_range=0.2,height_shift_range=0.2,shear_range=0.3)
```

```
test_datagen=ImageDataGenerator(rescale=1./255)
```

```
#setting image width and height
img_width=80
img_height=80
```

```
#apply ImageDataGenerator on our both training and testing images
x_train=train_datagen.flow_from_directory(r'C:\Users\hai\Desktop\Iceberg detection in satelite images\Dataset\Train',
                                          target_size=(img_width,img_height),batch_size=30
                                          ,class_mode='binary')
```

```
Found 1284 images belonging to 2 classes.
```

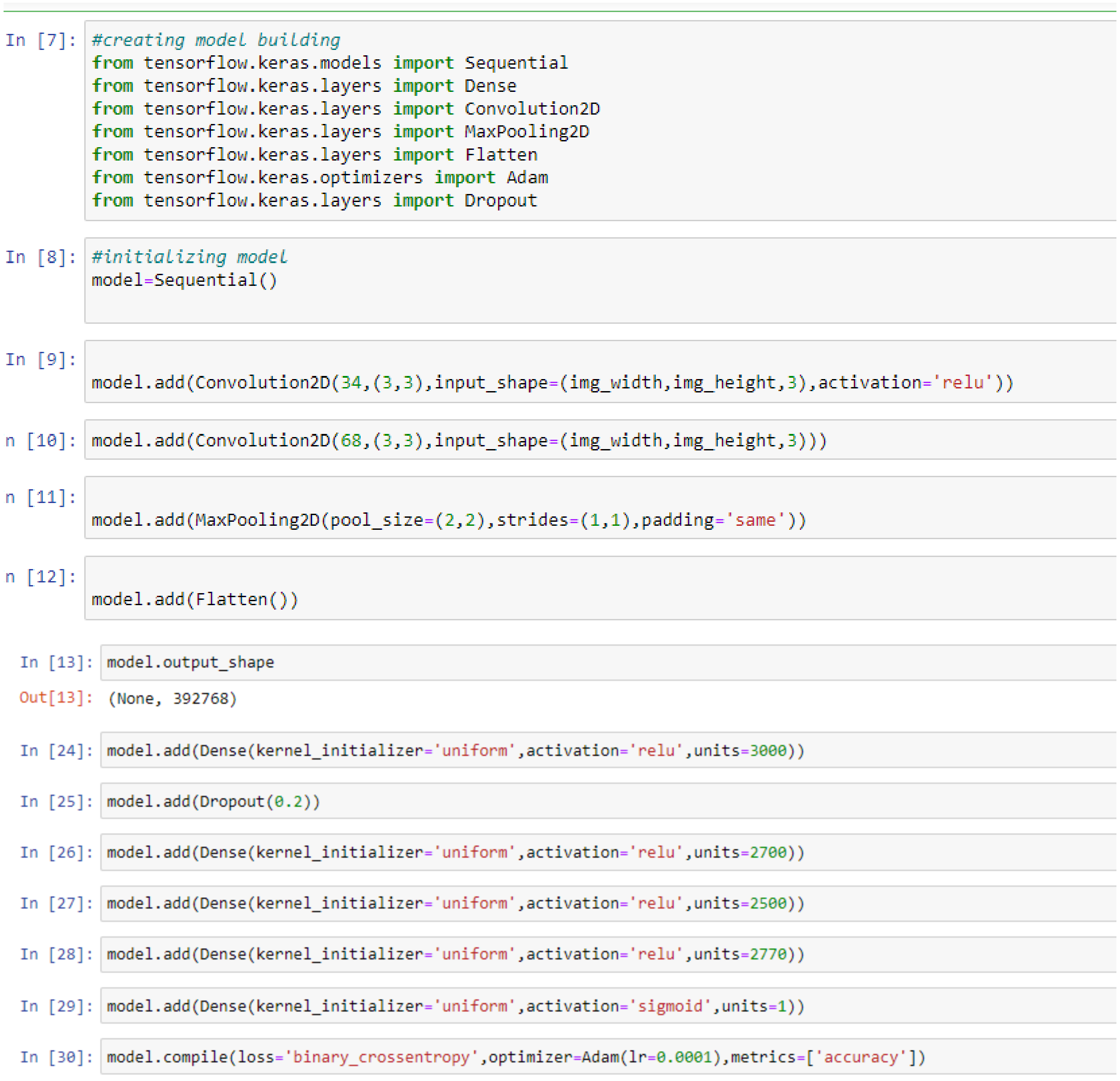
```
x_test=test_datagen.flow_from_directory(r'C:\Users\hai\Desktop\Iceberg detection in satelite images\Dataset\Test',
                                        target_size=(img_width,img_height),batch_size=30,
                                        class_mode='binary')
```

```
Found 320 images belonging to 2 classes.
```

Fitting the model

We first creating the model and initialize that with sequential.

Sequence models are the machine learning models that input or output sequences of data. Sequential data includes text streams, audio clips, video clips, time-series data and etc. Recurrent Neural Networks (RNNs) is a popular algorithm used in sequence models. Applications of Sequence Models.The sequential API allows you to create models layer-by-layer for most problems.



```
In [7]:  #creating model building
         from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import Dense
         from tensorflow.keras.layers import Convolution2D
         from tensorflow.keras.layers import MaxPooling2D
         from tensorflow.keras.layers import Flatten
         from tensorflow.keras.optimizers import Adam
         from tensorflow.keras.layers import Dropout

In [8]:  #initializing model
         model=Sequential()

In [9]:
         model.add(Convolution2D(34,(3,3),input_shape=(img_width,img_height,3),activation='relu'))

n [10]:  model.add(Convolution2D(68,(3,3),input_shape=(img_width,img_height,3)))

n [11]:
         model.add(MaxPooling2D(pool_size=(2,2),strides=(1,1),padding='same'))

n [12]:
         model.add(Flatten())

In [13]: model.output_shape

Out[13]: (None, 392768)

In [24]: model.add(Dense(kernel_initializer='uniform',activation='relu',units=3000))

In [25]: model.add(Dropout(0.2))

In [26]: model.add(Dense(kernel_initializer='uniform',activation='relu',units=2700))

In [27]: model.add(Dense(kernel_initializer='uniform',activation='relu',units=2500))

In [28]: model.add(Dense(kernel_initializer='uniform',activation='relu',units=2770))

In [29]: model.add(Dense(kernel_initializer='uniform',activation='sigmoid',units=1))

In [30]: model.compile(loss='binary_crossentropy',optimizer=Adam(lr=0.0001),metrics=['accuracy'])
```

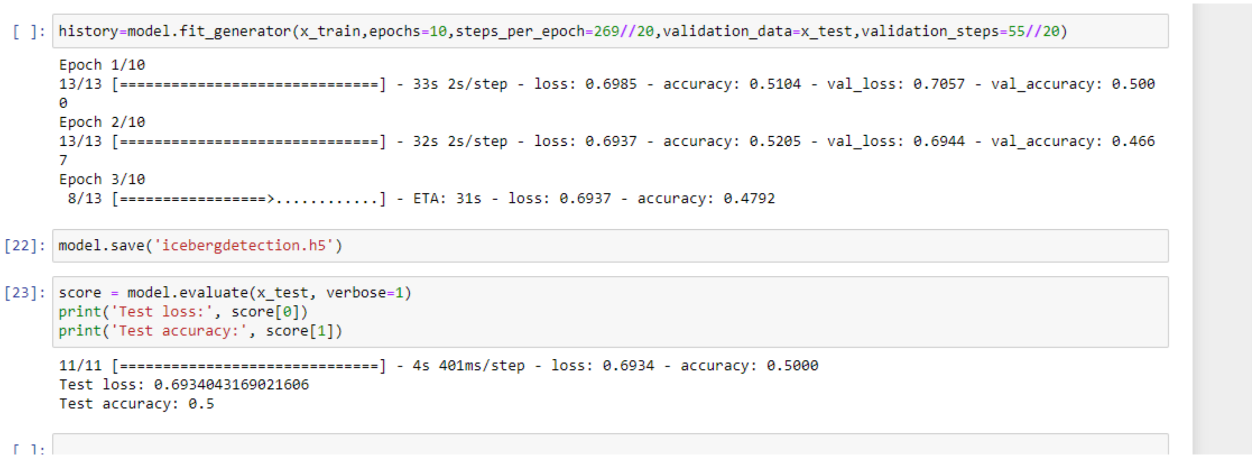
1.We use two convolution layer followed by Maxpooling layer which in turn is

followed by convolution layer.We use 20% dropout to reduce the overfitting.

2. Dense layer is used with relu and sigmoid function and with kernel initializer uniform.

3.Finally we compile the model with Adam and metrics accuracy binary_crossentropy in loss function.

Let's train our model with 10 epochs with batch size 13 and validation on x test and validation steps are 4.

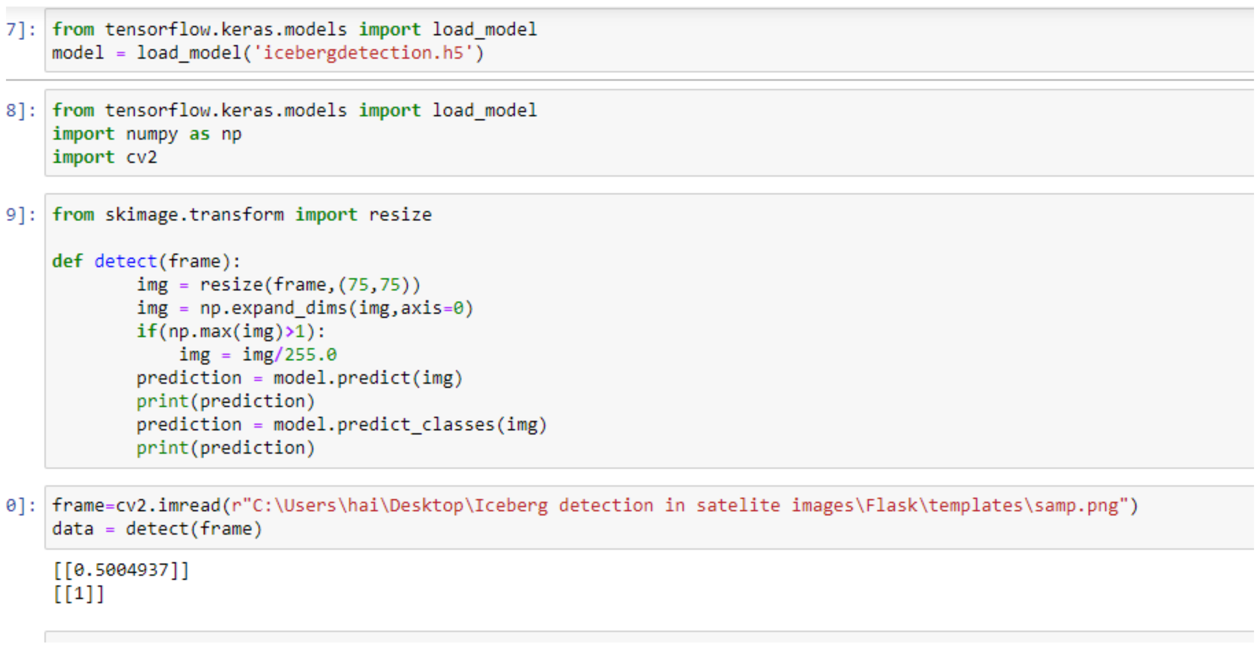


```
[ ]: history=model.fit_generator(x_train,epochs=10,steps_per_epoch=269//20,validation_data=x_test,validation_steps=55//20)

     Epoch 1/10
     13/13 [==============================] - 33s 2s/step - loss: 0.6985 - accuracy: 0.5104 - val_loss: 0.7057 - val_accuracy: 0.500
     0
     Epoch 2/10
     13/13 [==============================] - 32s 2s/step - loss: 0.6937 - accuracy: 0.5205 - val_loss: 0.6944 - val_accuracy: 0.466
     7
     Epoch 3/10
      8/13 [=================>...........] - ETA: 31s - loss: 0.6937 - accuracy: 0.4792

[22]: model.save('icebergdetection.h5')

[23]: score = model.evaluate(x_test, verbose=1)
      print('Test loss:', score[0])
      print('Test accuracy:', score[1])

     11/11 [==============================] - 4s 401ms/step - loss: 0.6934 - accuracy: 0.5000
     Test loss: 0.6934043169021606
     Test accuracy: 0.5

[ ]:
```

We save our model with extension icebergdetection.h5 if we want to predict an image either it is iceberg or ship we can directly load the model and we can upload image and can get prediction.After saving model we evaluate the model.we get accuracy as 0.5 and test loss is 0.6.

Testing the model

We test the saved model to verify either it is giving correct prediction or not.For that we play a video about icebergs and load that into frames .In any frame if iceberg is present it will predict it.



```
7]: from tensorflow.keras.models import load_model
    model = load_model('icebergdetection.h5')
```

```
8]: from tensorflow.keras.models import load_model
    import numpy as np
    import cv2
```

```
9]: from skimage.transform import resize

    def detect(frame):
            img = resize(frame,(75,75))
            img = np.expand_dims(img,axis=0)
            if(np.max(img)>1):
                img = img/255.0
            prediction = model.predict(img)
            print(prediction)
            prediction = model.predict_classes(img)
            print(prediction)
```

```
0]: frame=cv2.imread(r"C:\Users\hai\Desktop\Iceberg detection in satelite images\Flask\templates\samp.png")
    data = detect(frame)

    [[0.5004937]]
    [[1]]
```

Application Building

Now we will be building a Flask application that is used for building our UI which in backend can be interfaced to the model to get predictions. Flask application requires an HTML page for Frontend and a Python file for the backend which takes care of the interface with the model.

Build A Flask Application

Step 1: Load the required packages

Initialize Graph, Load The Model, Initialize The Flask And Load the video

step 2:Graph element is required to work with TensorFlow. So, the graph element is created explicitly.
Step 3: Configure the home page

Pre-Process The Frame

Pre-process the captured frame and give it to the model for prediction. Whenever the iceberg is detected an alarm is raised.

Pre-process the image by resizing it and increasing the dimension of the image to meet the model .

Build The HTML Page

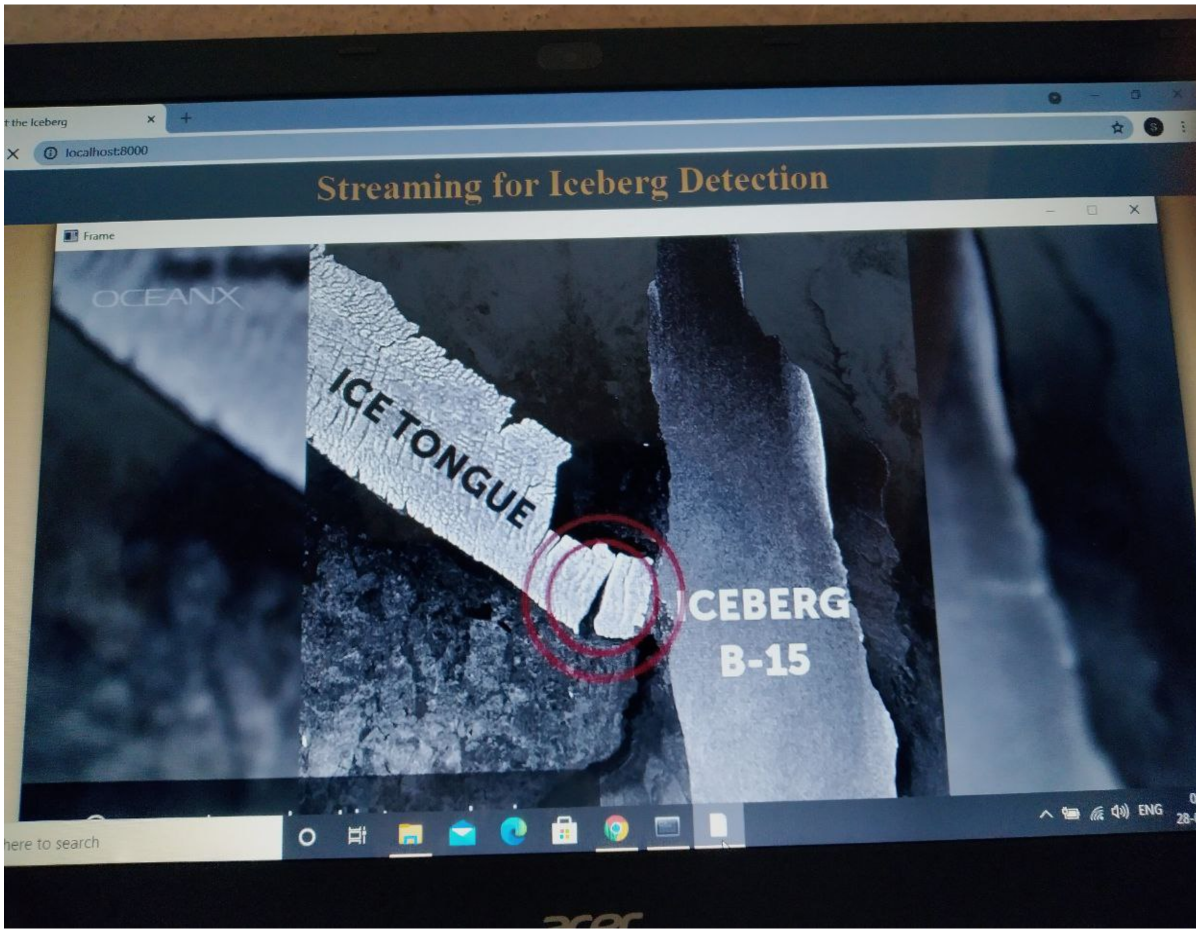
Build an HTML page to display the processed video on the screen, so that the concerned person can monitor.

Step 1: Run the application

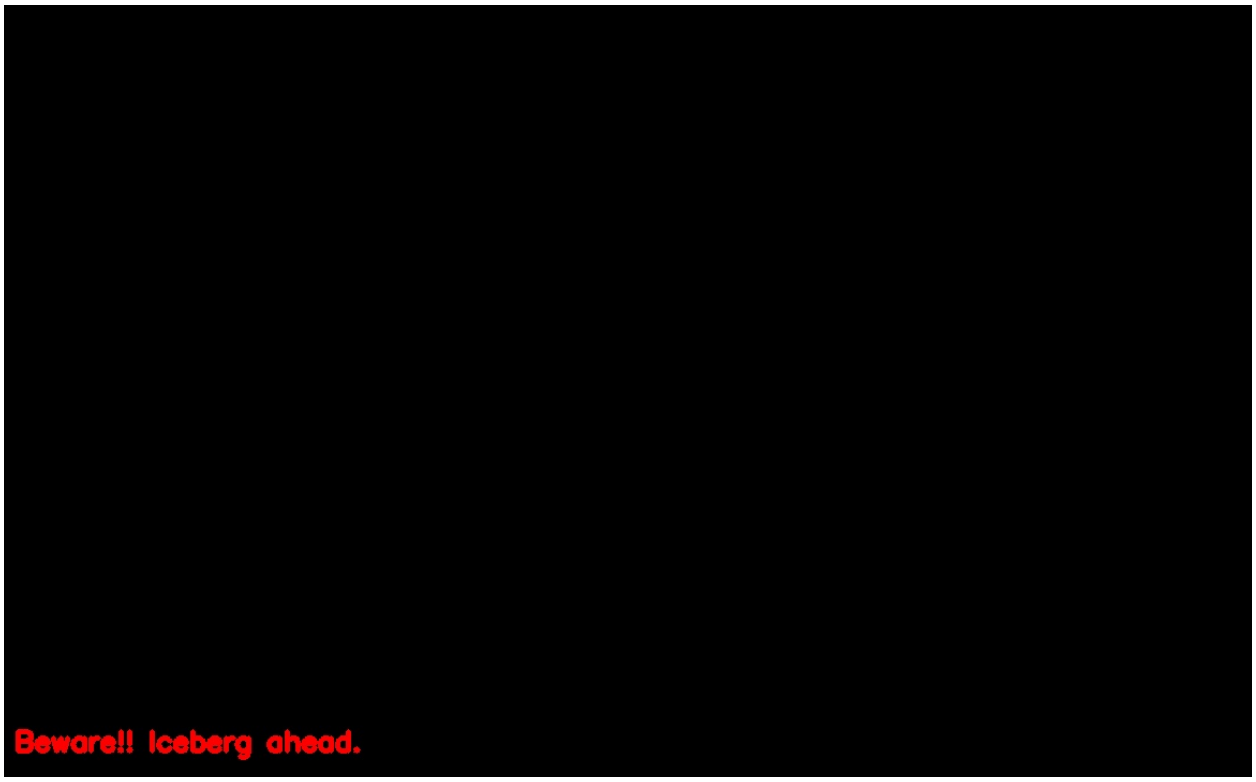
When the python file is executed the localhost is activated on 8000 port and can be accessed through it.

Output

Open the browser and navigate to localhost:8000 to check your application



when we run the the flask file and open the browser with localhost:8000 video will played about iceberg if any iceberg appear our model will give alert sound with one frame displaying beware iceberg ahead .If no iceberg appear it will predict a iceberg it is ship it is safe.



```
Beware!! Iceberg ahead.
```

Advantages

Because of this iceberg detection ship can know with remote sensing either it is ship or iceberg thus escape from an accident

Conclusion

We have done project on Iceberg detection in satelite imges which is detecting an iceberg or not with remote sensors.For doing that we have created a model which can capable to classify the image is ship or iceberg using some Artificial intelligence Techniques.

From 3 CSE-A