CHAPTER 1
INTRODUCTION

Iceberg Detection In Satellite Images Using IBM Watson Studio

1.1 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:

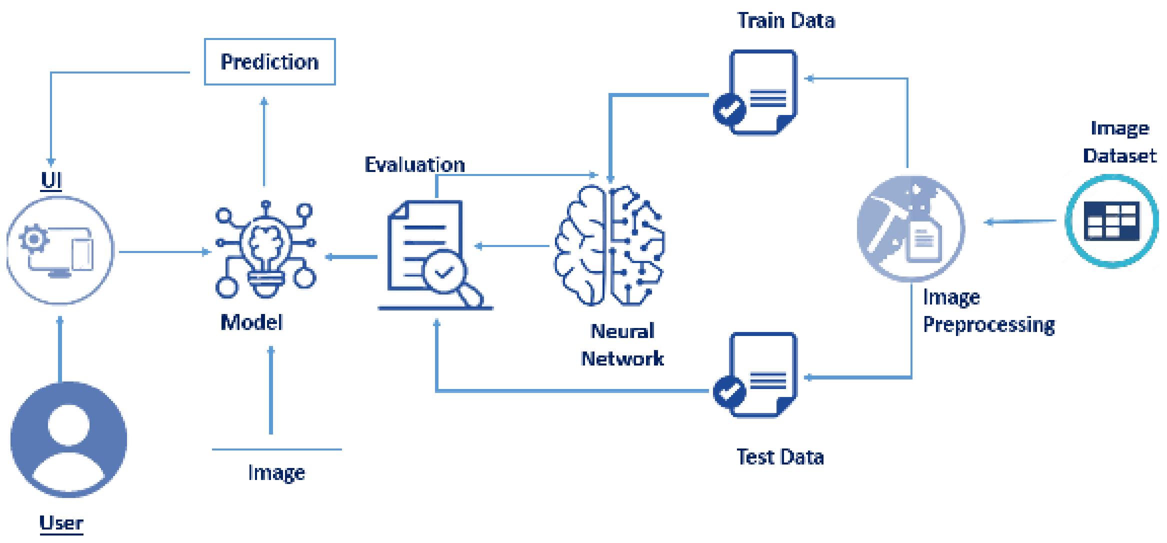


FIG 1.1 TECHNICAL ARCHITECTURE

1.2 **Pre-Requisites:**

1. To complete this project, you must require the following software's, and packages

   · Anaconda (IDLE / Spyder / PyCharm)(Python 3.7):

- o Refer to the link below to download anaconda
- o Link : https://www.youtube.com/watch?v=5mDYijMfSzs
- Link: https://www.anaconda.com/products/individual

2. Python Packages

- Tensorflow- This package is used as backend support to Keras
- Keras-This package is used for building Neural Network layers
- opencv-This package is used for image processing
- Flask- To build a web application

If you are using anaconda navigator, follow the below steps to download the required packages:

- · Open anaconda prompt as administrator.
- · Type "pip install tensorflow==1.14.0" and click enter.
- · Type "pip install keras=2.2.4" and click enter.
- · Type "pip install opencv-python" and click enter.
- · Type "pip install imutils" and click enter.
- · Type "pip install flask" and click enter.

CHAPTER 2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 AIM:

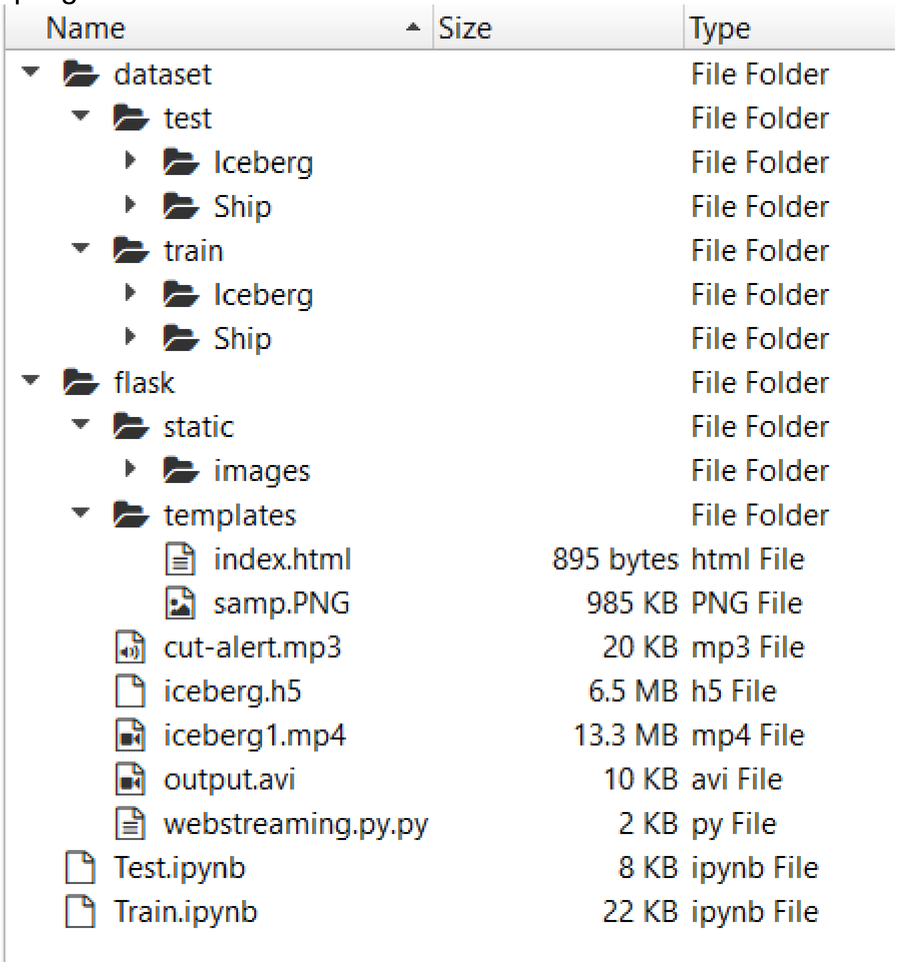
The main aim of this project is to build a model that automatically identifies whether a remotely sensed target is an iceberg or not. Often times an iceberg is wrongly classified as a ship. The algorithm had to be extremely accurate because lives and billions of dollars in energy infrastructure are at stake.

2.2 PROJECT FLOW:

- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once model analyses the input the detect the iceberg
- To accomplish this, we have to complete all the activities and tasks listed below
- Data collection
  Create train and test folders
- Image preprocessing
  Import image data generator library and configure it
  Apply imagedatagenerator functionality to train and test set
- Model building
  Import The Required Model Building Libraries
  Initialize The Model
  Add The Convolution Layer
  Add The Pooling Layer
  Add The Flatten Layer
  Adding The Dense Layers
  Compile The Model
  Fit And Save The Model
- Test the model
  Import the packages and load the saved model
  Load the test image,pre-process it and predict
- Application building
  Build A Flask Application
  Initialize Graph, Load The Model, Initialize The Flask App
  And Load The Video
  Pre-Process The Frame
  Pre-Process The Frame Part 2
  Build The HTML Page

2.3 PROJECT STRUCTURE:

To complete this project the following structure is used which consists of dataset and main program



Name	Size	Type
dataset		File Folder
test		File Folder
Iceberg		File Folder
Ship		File Folder
train		File Folder
Iceberg		File Folder
Ship		File Folder
flask		File Folder
static		File Folder
images		File Folder
templates		File Folder
index.html	895 bytes	html File
samp.PNG	985 KB	PNG File
cut-alert.mp3	20 KB	mp3 File
iceberg.h5	6.5 MB	h5 File
iceberg1.mp4	13.3 MB	mp4 File
output.avi	10 KB	avi File
webstreaming.py.py	2 KB	py File
Test.ipynb	8 KB	ipynb File
Train.ipynb	22 KB	ipynb File

FIG 2.1 DATASET

CHAPTER 3
EXPERIMENTAL OR MATERIAL AND METHODS; ALGORITHMS USED

3.1Data Collection:

In this, we will be collecting data for building our project. We will be creating two folders one for training and the other for testing. Images present in the training folder will be used for building the model and the testing images will be used for validating our model.

3.2 Create Train And Test Folders

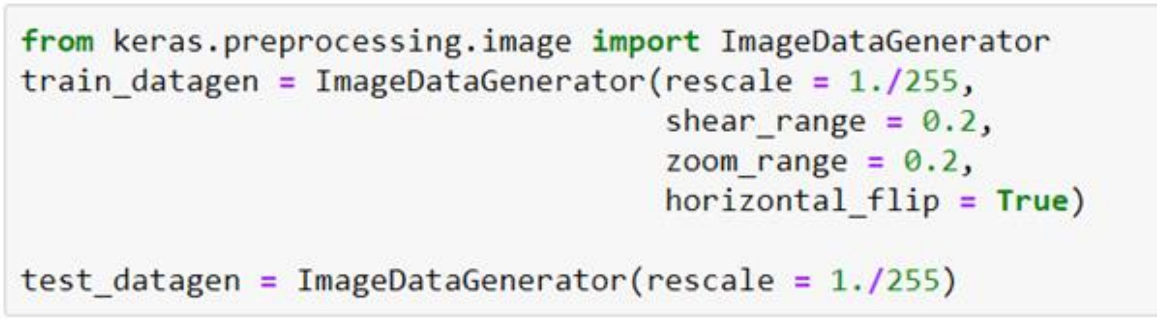
Step1: Create Train and Test folders with each folder having folders with images of icebergs and ships. A minimum of 100 images need to be present in each category folder to get the maximum no of features.

3.3 Image Preprocessing

In this, we will pre-process the images which will be used for building the model. Image pre-processing includes zooming, shearing, flipping to increase the robustness of the model after it is built. We will be using the Keras package for pre-processing images.

3.4 Import ImageDataGenerator Library And Configure It

Import ImageDataGenerator and create an instance for which include shearing, rescale, zooming, etc to make the model robust with a different type of images



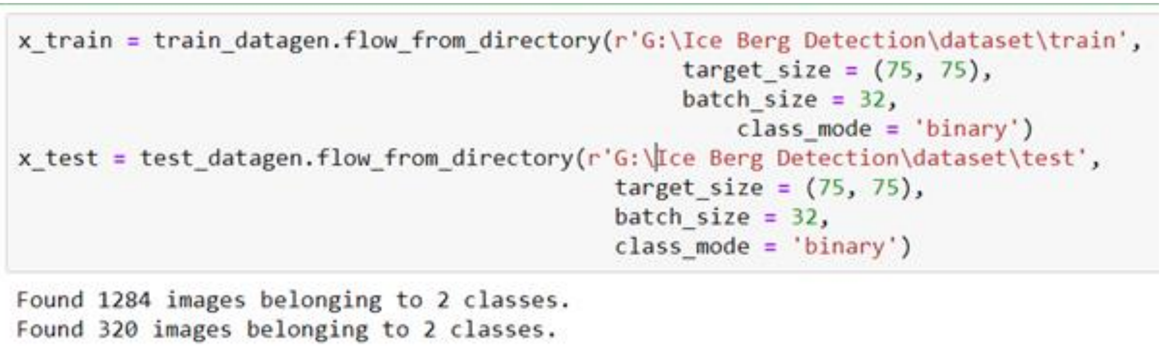
```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

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

FIG 3.1 IMAGEDATAGENERATOR LIBRARY

Reference:Keras documentation..**Keras documentation..https://keras.io/api/preprocessing/image/**

3.5 Apply ImageDataGenerator Functionality To Train And Test Set

Specify the path of both the folders in the flow_from_directory method.



```
x_train = train_datagen.flow_from_directory(r'G:\Ice Berg Detection\dataset\train',
                                                target_size = (75, 75),
                                                batch_size = 32,
                                                    class_mode = 'binary')
x_test = test_datagen.flow_from_directory(r'G:\Ice Berg Detection\dataset\test',
                                              target_size = (75, 75),
                                              batch_size = 32,
                                              class_mode = 'binary')
```

```
Found 1284 images belonging to 2 classes.
Found 320 images belonging to 2 classes.
```

FIG 3.2 ImageDataGenerator Train And Test Set

Reference: Keras documentation..

Keras documentation..

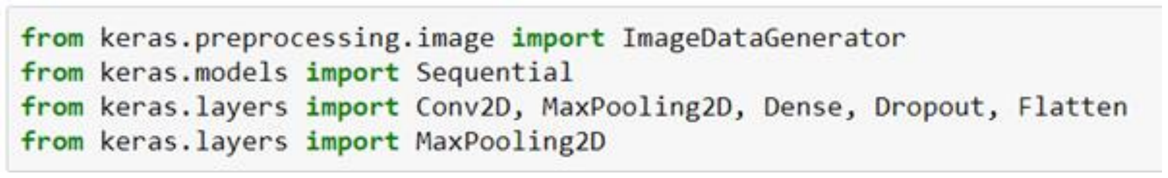
https://keras.io/api/preprocessing/image/#imagedatasetfromdirectory-function

3.6 Model Building

In this milestone, we start building our model by:
1.Initializing the mode
2.Adding Convolution layers
3. Adding Pooling layers
4.Flatten layer
5.Full connection layers which include hidden layers
At last, we compile the model with layers we added to complete the neural network structure

3.7 Import The Required Model Building Libraries

Import the libraries that are required to initialize the neural network layer, create and add different layers to the neural network model.



```
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Flatten
from keras.layers import MaxPooling2D
```

FIG 3.3 Model Building Libraries

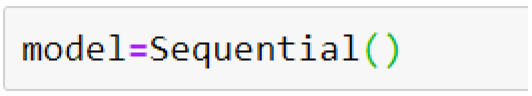
The code referenced in this video is from https://YouTube.com/Sentdex ..

The code referenced in this video is from https://YouTube.com/Sentdex and https://pythonprogramming.net/convolutional-neural-network-kats-vs-dogs-mach..

https://youtu.be/umGJ30-15_A

3.8 Initialize The Model

Initialize the neural network layer by creating a reference/object to the Sequential class.

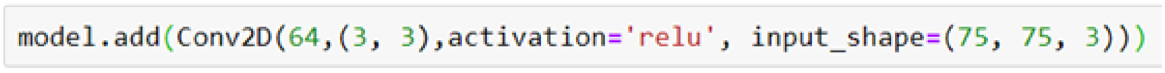


```
model=Sequential()
```

FIG 3.4 MODEL

3.9 Add The Convolution Layer

The first layer of the neural network model, the convolution layer will be added. To create a convolution layer, Convolution2D class is used. It takes the number of feature detectors, feature detector size, expected input shape of the image, activation function as arguments. This layer applies feature detectors on the input image and returns a feature map (features from the image).

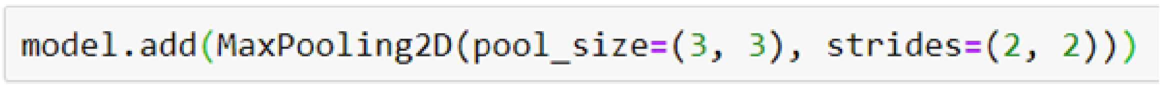


```
model.add(Conv2D(64,(3, 3),activation='relu', input_shape=(75, 75, 3)))
```

FIG 3.5 CONVOLUTION LAYER

3.2.1 Add The Pooling Layer

After the convolution layer, usually, the pooling layer is added. Max pooling layer can be added using MaxPooling2D class. It takes the pool size as a parameter. The efficient size of the pooling matrix is (2,2). It returns the pooled feature maps. (Note: Any number of convolution layers, pooling and dropout layers can be added)

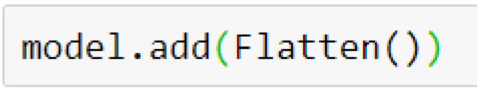


```
model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
```

FIG 3.6 POOLING LAYER

3.2.2 Add The Flatten Layer

The flatten layer is used to convert an n-dimensional array to a 1-dimensional array. This 1D array will be given as input to ANN layers.

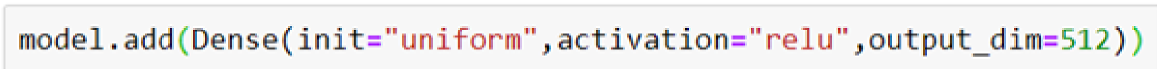


```
model.add(Flatten())
```

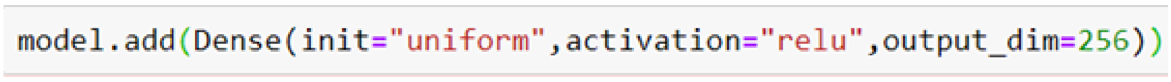
FIG 3.7 FLATTEN LAYER

3.2.3 Adding The Dense Layers

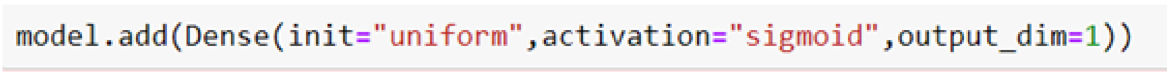
Three dense layers are added which usually takes the number of units/neurons. Specifying the activation function, kind of weight initialization is optional.



```
model.add(Dense(init="uniform",activation="relu",output_dim=512))
```



```
model.add(Dense(init="uniform",activation="relu",output_dim=256))
```

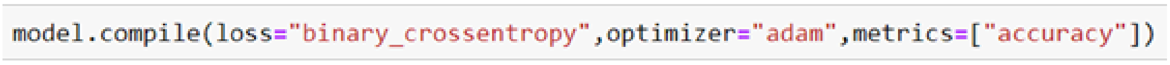


```
model.add(Dense(init="uniform",activation="sigmoid",output_dim=1))
```

FIG 3.8 DENSE LAYER

3.2.4 Compile The Model

After adding all the required layers, the model is to be compiled. For this step, loss function, optimizer, and metrics for evaluation can be passed as arguments.



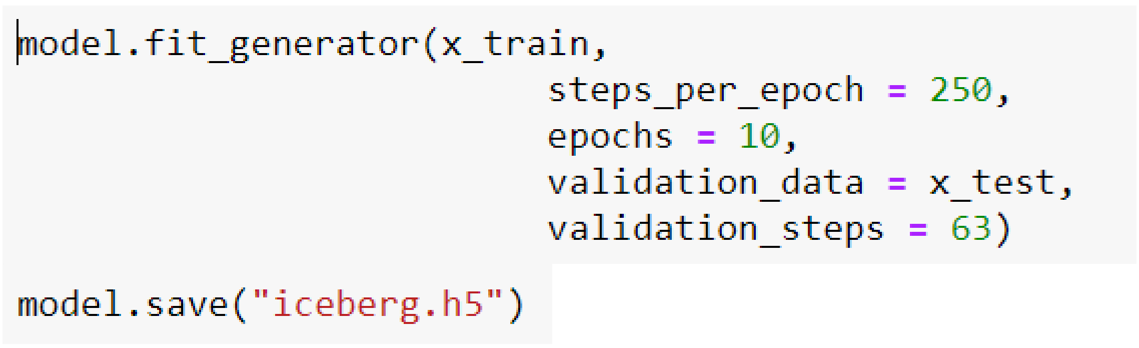
```
model.compile(loss="binary_crossentropy",optimizer="adam",metrics=["accuracy"])
```

FIG 3.9 COMPILE THE MODEL

3.2.5 Fit And Save The Model

Fit the neural network model with the train and test set, number of epochs and validation steps.

The weights are to be saved for future use. The weights are saved in as .h5 file using save().



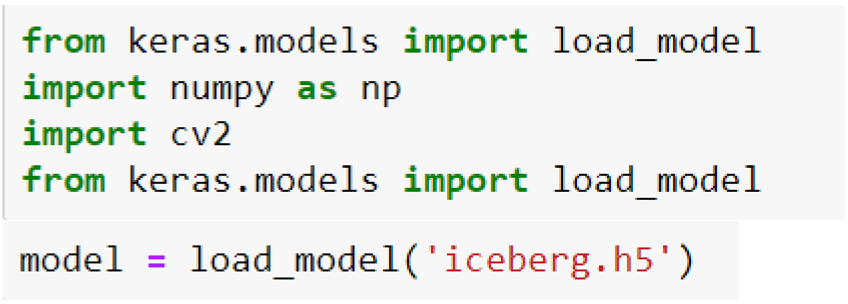
```
model.fit_generator(x_train,
                         steps_per_epoch = 250,
                         epochs = 10,
                         validation_data = x_test,
                         validation_steps = 63)
model.save("iceberg.h5")
```

FIG 3.2.1 SAVE THE MODEL

3.2.6Test The Model

Now we test the model by passing an image to get predictions Import The Packages And Load The Saved Model

Import the packages that are used to load the model and get the predictions

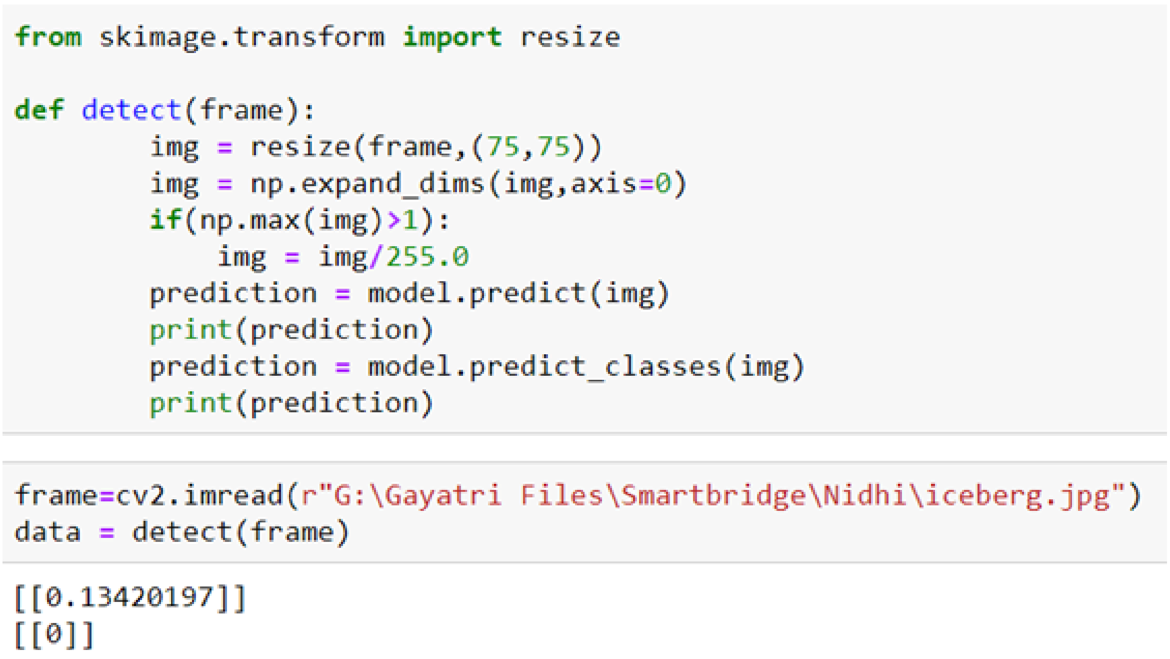


```
from keras.models import load_model
import numpy as np
import cv2
from keras.models import load_model
model = load_model('iceberg.h5')
```

FIG 3.2.2 TEST THE MODEL

3.2.7 Load The Test Image, Pre-Process It And Predict

Pre-processing the image includes converting the image to array and resizing according to the model. Give the pre-processed image to the model to know to which class your model belongs to.



```
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)
```

```
frame=cv2.imread(r"G:\Gayatri Files\Smartbridge\Nidhi\iceberg.jpg")
data = detect(frame)
```

```
[[0.13420197]]
[[0]]
```

FIG 3.2.3 PREDICT

CHAPTER 4

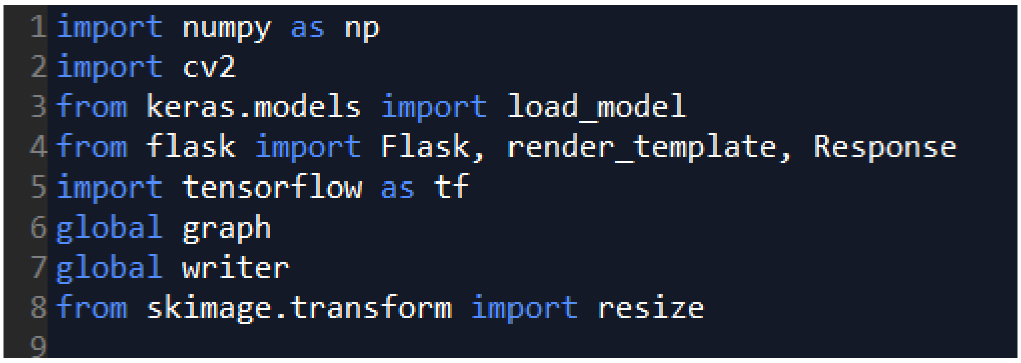
RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

4.1 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.

4.2 Build A Flask Application

Step 1: Load the required packages



```
import numpy as np
import cv2
from keras.models import load_model
from flask import Flask, render_template, Response
import tensorflow as tf
global graph
global writer
from skimage.transform import resize

```

FIG 4.1 FLASK APP

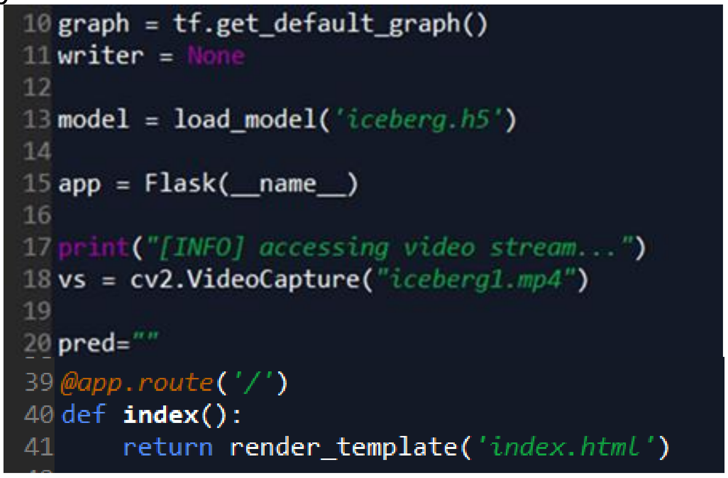
Full-Stack Web Development Internship Program: http://bit.ly/fs-int..

Full-Stack Web Development Internship Program: http://bit.ly/fs-internship This Edureka Python Flask tutorial will cover all the fundamentals of Fl..

https://www.youtube.com/watch?v=lj4I_CvBnt0

4.3 Initialize Graph, Load The Model, Initialize The Flask App And Load The Video

Graph element is required to work with TensorFlow. So, the graph element is created explicitly.



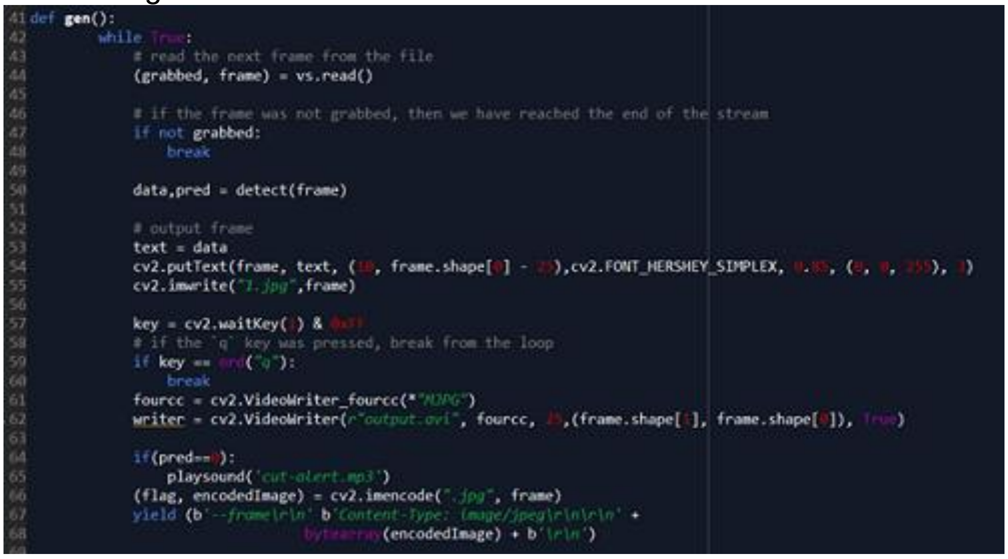
```
graph = tf.get_default_graph()
writer = None

model = load_model('iceberg.h5')

app = Flask(__name__)

print("[INFO] accessing video stream...")
vs = cv2.VideoCapture("iceberg1.mp4")

pred=""
@app.route('/')
def index():
    return render_template('index.html')
```

FIG 4.2 FLASK APP CONT

Step 3: Configure the home page

4.4 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.

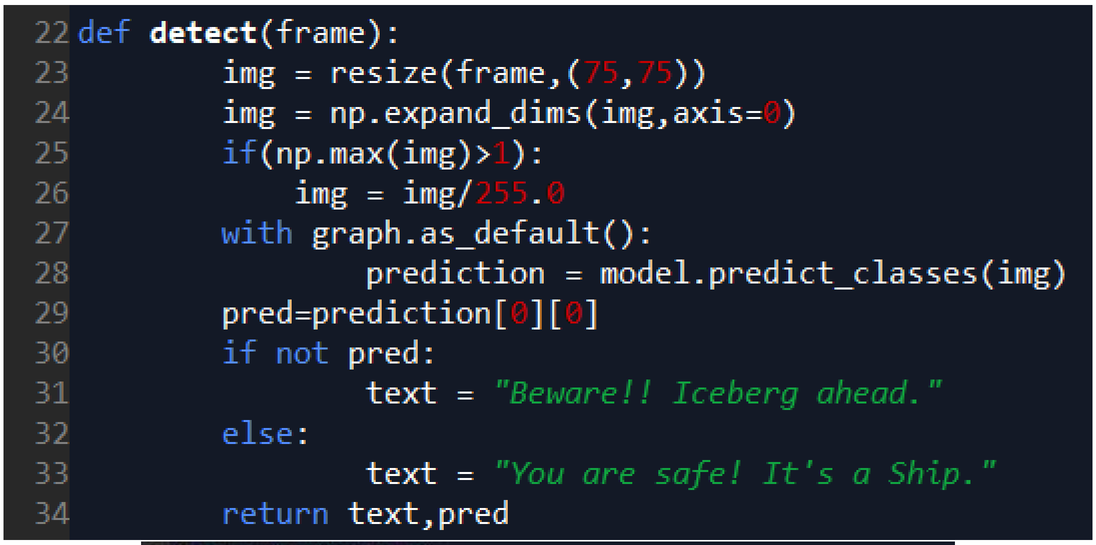


```python
def gen():
        while True:
            # read the next frame from the file
            (grabbed, frame) = vs.read()

            # if the frame was not grabbed, then we have reached the end of the stream
            if not grabbed:
                break

            data,pred = detect(frame)

            # output frame
            text = data
            cv2.putText(frame, text, (10, frame.shape[0] - 25),cv2.FONT_HERSHEY_SIMPLEX, 0.85, (0, 0, 255), 3)
            cv2.imwrite("1.jpg",frame)

            key = cv2.waitKey(1) & 0xFF
            # if the `q` key was pressed, break from the loop
            if key == ord("q"):
                break
            fourcc = cv2.VideoWriter_fourcc(*"MJPG")
            writer = cv2.VideoWriter(r"output.avi", fourcc, 25,(frame.shape[1], frame.shape[0]), True)

            if(pred==0):
                playsound('cut-alert.mp3')
            (flag, encodedImage) = cv2.imencode(".jpg", frame)
            yield (b'--frame\r\n' b'Content-Type: image/jpeg\r\n\r\n' +
                                 bytearray(encodedImage) + b'\r\n')
```

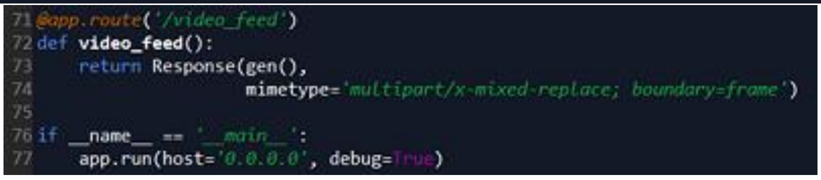
FIG 4.3 THE FRAME

4.5 Pre-Process The Frame Part -2

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



```python
def detect(frame):
        img = resize(frame,(75,75))
        img = np.expand_dims(img,axis=0)
        if(np.max(img)>1):
            img = img/255.0
        with graph.as_default():
                prediction = model.predict_classes(img)
        pred=prediction[0][0]
        if not pred:
                text = "Beware!! Iceberg ahead."
        else:
                text = "You are safe! It's a Ship."
        return text,pred
```



```python
@app.route('/video_feed')
def video_feed():
    return Response(gen(),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)
```

FIG 4.4 THE FRAME PART 2

4.6 Result: The algorithm have extremely accurate because lives and billions of dollars in energy infrastructure are at stake.

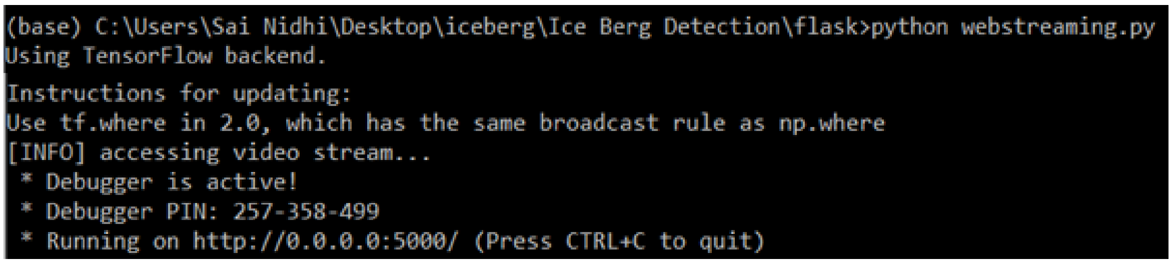
CHAPTER 5
SUMMARY AND CONCLUSIONS

5.1 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 5000 port and can be accessed through it.



```
(base) C:\Users\Sai Nidhi\Desktop\iceberg\Ice Berg Detection\flask>python webstreaming.py
Using TensorFlow backend.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
[INFO] accessing video stream...
 * Debugger is active!
 * Debugger PIN: 257-358-499
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

FIG 5.1 HTML PAGE

5.2 SUMMARY:

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.

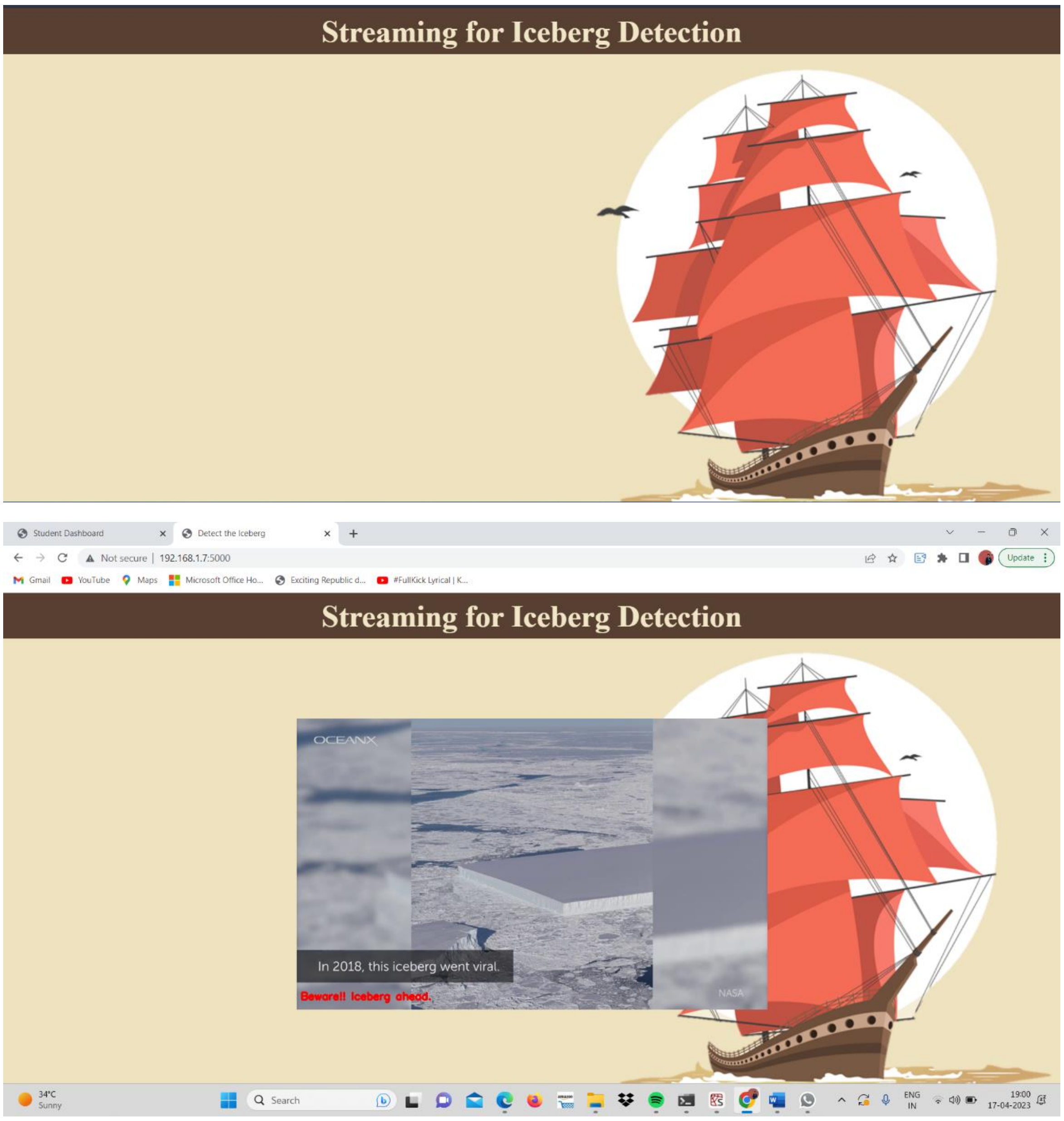
5.3 CONCLUSION:

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.

5.4 OUTPUT:

Open the browser and navigate to localhost:5000 to check your application

A.SCREENSHOTS



In 2018, this iceberg went viral.

Beware!! Iceberg ahead.

FIG 5.2 SCREENSHOT

B.SOURCE CODE

```
import numpy as np

import cv2

from keras.models import load_model

from flask import Flask, render_template, Response

#import tensorflow as tf

global graph

global writer
```

```
from skimage.transform import resize

#graph = tf.get_default_graph()

writer = None


model = load_model('C:/spyder/flask/iceberg.h5')




app = Flask(_name_)


print("[INFO] accessing video stream...")

vs = cv2.VideoCapture("iceberg1.mp4")


pred=""

def detect(frame):

        img = resize(frame,(128,128))

        img = np.expand_dims(img,axis=0)

        if(np.max(img)>1):

            img = img/255.0

        #with graph.as_default():

        prediction = model.predict(img)

        pred = [prediction[0][0]]


        if pred:

                text = "Beware!! Iceberg ahead."

        else:
            text = "You are safe! It's a Ship."
        return text


@app.route('/')
def index():
    return render_template('index.html')


def gen():
        while True:


            (grabbed, frame) = vs.read()




            if not grabbed:
                break


            data = detect(frame)


            # output frame
            text = data
            cv2.putText(frame, text, (10, frame.shape[0] -
25),cv2.FONT_HERSHEY_SIMPLEX, 0.85, (0, 0, 255), 3)
            cv2.imwrite("1.jpg",frame)




            key = cv2.waitKey(1) & 0xFF
            # if the `q` key was pressed, break from the loop
            if key == ord("q"):
                break
            fourcc = cv2.VideoWriter_fourcc(*"MJPG")
            writer = cv2.VideoWriter(r"output.avi", fourcc, 25,(frame.shape[1],
frame.shape[0]), True)


            if(pred==0):
                playsound('cut_alert.mp3')
            (flag, encodedImage) = cv2.imencode(".jpg", frame)
            yield (b'--frame\r\n' b'Content-Type: image/jpeg\r\n\r\n' +
                        bytearray(encodedImage) + b'\r\n')
        #cv2.destroyAllWindows()


@app.route('/video_feed')
def video_feed():
    return Response(gen(),
                mimetype='multipart/x-mixed-replace; boundary=frame')


if _name_ == '_main_':
    app.run(host='0.0.0.0', debug=False)
```