**Iceberg Detection In Satellite Images Using IBM Watson Studio**

**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.

a.Overview

Apart from local weather effects, such as fog production, icebergs have two main impacts on climate. Iceberg production affects the mass balance of the parent [ice](https://www.britannica.com/science/ice-sheet)sheets, and melting icebergs influence both ocean structure and global sea level.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.

b.  Purpose

The application of satellite images for ship and iceberg monitoring is essential in many ways in Arctic waters. Even though the detection of ships and icebergs in images is well established using Geoscience techniques, the discrimination between those two target classes still represents a challenge for operational scenarios. This thesis project proposes the application of Support Vector Machine (SVM), Convolutional Neural Networks (CNN), and SingleShot Detector (SSD) for ship-iceberg detection in satellite images. The CNN model is compared with SVM and SSD, and the final results indicate not only a superior classification performance of the proposed methods but also the object detection results from SSD.

**2. Literature Survey:**

a. Existing problem

Deep Learning for Iceberg detection in Satellite Images  :-

The main objective of this blog is to develop methods for detecting icebergs using satellite radar data and high spatial resolution images in the visible spectral range. 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 developed method of iceberg detection is based on statistical criteria for finding gradient zones in the analysis of two-dimensional fields of satellite images. The approaches proposed to detect icebergs from satellite data allow improving the quality and efficiency of service for a wide number of users with ensuring the efficiency and safety of Arctic navigation and activities on the Arctic shelf.

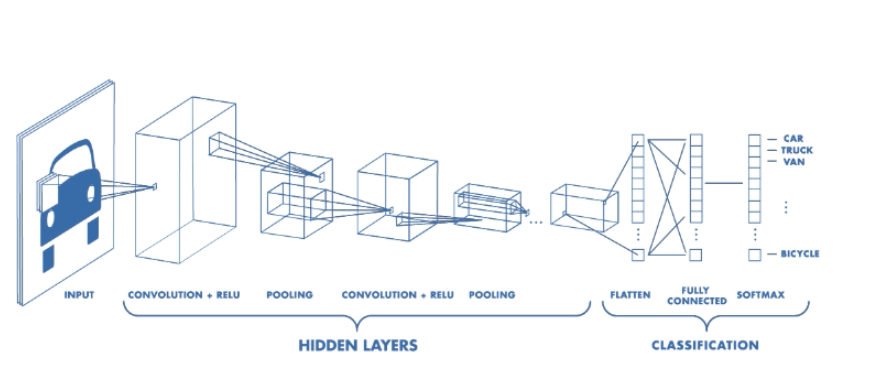
This project is a part of the [Statoil/C-CORE Iceberg Classifier Challenge](https://www.kaggle.com/c/statoil-iceberg-classifier-challenge/data) held on Kaggle.

b.Proposed Solution

The main aim of this proposed solution 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. And the other objective of our study is to estimate the possibility of using different satellite images for iceberg detection among open water, fast ice and drifting ice.

**3. THEORETICAL ANALYSIS**

a. *Architecture:*



**b.Hardware Designing:**

Windows               -         7 or newer

Processor               -         intel core i3 or above or equivalent.

Hard disk               -         20 GB

RAM                     -         1 GB or more

* Skills  Required**:** Python,CNN,Flask Integration,IBM Watson Studio

**c.Software Designing**:

 *Anaconda Navigator :*

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross-platform,  package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook,QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupiter notebook and spyder.

  *Tensor flow:*

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers can easily build and deploy ML powered applications.

 *Keras :*

Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features:

* Consistent, simple and extensible API.
* Minimal structure - easy to achieve the result without any frills.
* It supports multiple platforms and backends.
* It is user friendly framework which runs on both CPU and GPU.
* Highly scalability of computation.

 *Python 3.7 :*

Python is broadly utilized universally and is a high-level programming language. It was primarily introduced for prominence on code, and its language structure enables software engineers to express ideas in fewer lines of code. Python is a programming language that gives you a chance to work rapidly and coordinate frameworks more effectively.

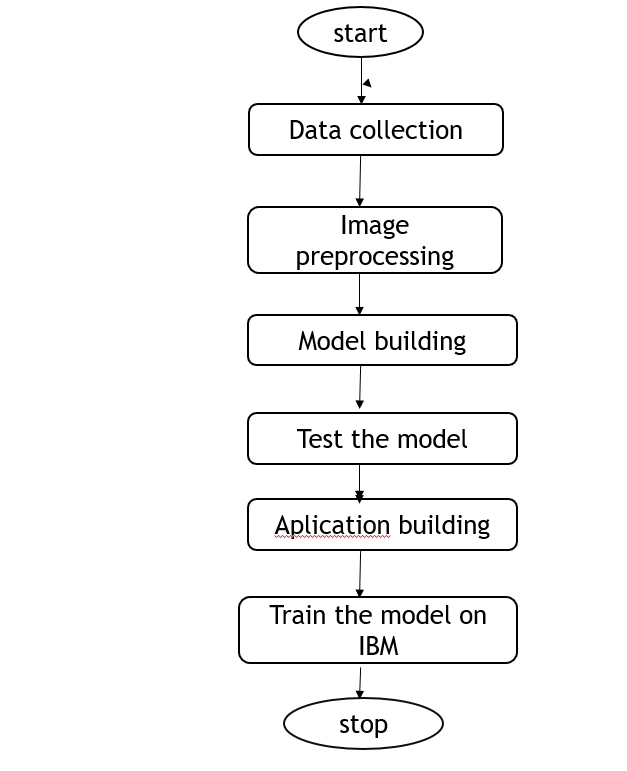
**4. Experimental Investigations**

Many articles started the need and use of iceberd detection in satellite images.It can be developed even with the environment and tools like scikit-learn,numpy,keras,pandas, and matplotlib. Here in this proposed system flask API and IBM Watson studio is used.

Reference:

<https://towardsdatascience.com/deep-learning-for-iceberg-detection-in-satellite-images-c667acf4bad0>

**5.Flowchart**



**6. Result**

The model can make the users aware of the performance of the iceberg detection in satellite imagesbased on the processed inputs and displys the areas(paths).

**7. Advantages & Disadvantages**

**Icebergs are a threat to maritime activities well outside the polar regions. For particular operational activities in areas prone to icebergs, it can be necessary to monitor icebergs upstream of the operational area, giving time for iceberg management or response to take place.**

**a. Advantages:**

* Automatic detection of icebergs in satellite images is regarded a useful tool to provide information necessary for safety in Arctic shipping or operations over large ocean areas in near-real time.
* The main objective of our study is to estimate the possibility of using different satellite images for iceberg detection among open water, fast ice and drifting ice.
* The final detection is performed on the identified blobs using the NN algorithm.
* Accurate predictions.
* Easy usable interface.
* Results shown are without any errors.
* It gives information about iceberg detection in satellite images.

**b.Disadvantages:**

* The user needs patience because the learning process can take longer in some circumstances.
* Requires very large amount of training data.

**8. Applications**

The applications of iceberg detection includes:

☆ **Geological land cover classification (ICEBERG-LandCover)**

**☆** **Polar glaciology and geomorphology using 4D change detection (ICEBERG- ASIFT)**

**☆** **Mapping hydrology (ICEBERG-Rivers)**

**☆** **Biological feature detection (ICEBERG-Seals/Penguins)**

**9.Conclusions**

This is one of the greatest examples where deep learning can be used to solve a challenging real-world problem. If we are able to detect and segment icebergs in an image, it would be of great help to the logistics and transportation team in northern countries like Sweden, Norway and Canada. It could bring a whole new dimension of transport for container ships and vessels by tracking icebergs from satellite images and videos in real-time.

**10. Future Scope**

This system can be further extended by adding new methodologies and different techniques and made it available to all those who are very interested to learn.

* Detect icebergs in satellite images easily with good accuracy .
* Use other algorithms and techniques with less layers to get better accuracy.
* Use other platforms like IBM watson studio for getting  result as quick as possible with great accuracy.

**11. Bibliography**

* <https://geocento.com/satellite-imagery-case-studies/radar-satellite-imagery-detect-icebergs/>
* <https://towardsdatascience.com/deep-learning-for-iceberg-detection-in-satellite-images-c667acf4bad0>
* <https://solidstatetechnology.us/index.php/JSST/article/view/10458>
* <https://www.mdpi.com/2072-4292/11/7/806/htm>

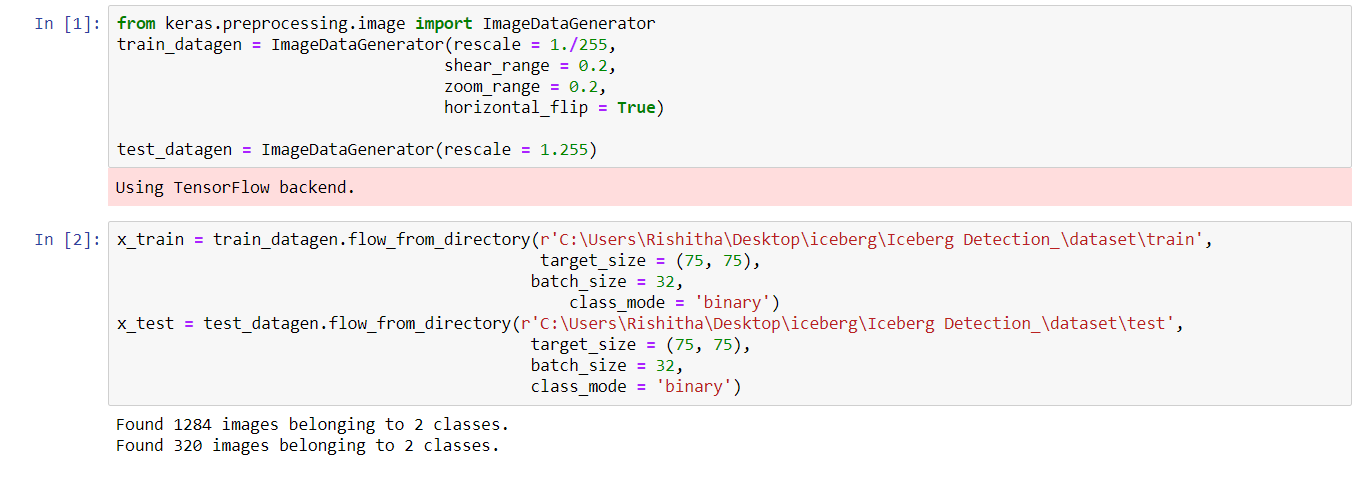
**12. Appendix**

**a. Source code**

This section has the following tasks

* **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.

****

* **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

****

* Test The Model

Now we test the model by passing an image to get predictions

****

And also in this section, we will be building a web application that is integrated into the model we built. A UI is provided for the uses where he has uploaded an image. The uploaded image is given to the saved model and prediction is showcased on the UI.

This section has the following tasks

* Building HTML Pages
* Building server-side script

Let us build the flask file ‘app.py’ which is a web framework written in python for server-side scripting. Let’s see step by step procedure for building the backend application.

**app.py**

from \_\_future\_\_ import division, print\_function

# coding=utf-8

import sys

import os

import glob

import numpy as np

from tensorflow.keras.preprocessing import image

from tensorflow.keras.applications.imagenet\_utils import preprocess\_input, decode\_predictions

from tensorflow.keras.models import load\_model

from tensorflow.keras import backend

import tensorflow as tf

global graph

tf.compat.v1.disable\_eager\_execution()

#graph=tf.compat.v1.get\_default\_graph()

from skimage.transform import resize

# Flask utils

from flask import Flask, redirect, url\_for, request, render\_template

from werkzeug.utils import secure\_filename

from gevent.pywsgi import WSGIServer

# Define a flask app

app = Flask(\_\_name\_\_)

# Model saved with Keras model.save()

model = load\_model('iceberg.h5')

@app.route('/', methods=['GET'])

def index():

# Main page

return render\_template('index.html')

@app.route('/predict', methods=['GET', 'POST'])

def upload():

if request.method == 'POST':

# Get the file from post request

f = request.files['file']

# Save the file to ./uploads

basepath = os.path.dirname(\_\_file\_\_)

file\_path = os.path.join(

basepath, 'uploads', secure\_filename(f.filename))

f.save(file\_path)

img = image.load\_img(file\_path, target\_size=(64, 64))

x = image.img\_to\_array(img)

x = np.expand\_dims(x, axis=0)

#with graph.as\_default():

preds = model.predict\_classes(x)

if preds[0][0]==0:

prediction="Yes,It is Iceberg"

else:

prediction="No,It is not Iceberg"

text = "prediction : "+prediction

# ImageNet Decode

return text

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=False,threaded = False)

**webstreaming.py**

import numpy as np

import imutils

import cv2

import os

from tensorflow.keras.models import load\_model

from flask import Flask, render\_template, url\_for, Response

import tensorflow as tf

global graph

global writer

#graph = tf.get\_default\_graph()

writer = None

model = load\_model('iceberg.h5')

from skimage.transform import resize

vals = ['Ship', 'Iceberg']

app = Flask(\_\_name\_\_)

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

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

pred=""

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=vals[prediction[0][0]]

        if pred:

                text = "Beware!! Iceberg ahead."

        else:

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

        return text

# initialize the video stream and pointer to output video file

@app.route('/')

def index():

    return render\_template('index.html')

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

            # resize the frame and then de

            #print(ix)

            #for x in vals:

            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)

            # check to see if the output frame should be displayed to our

            # screen

            # show the output frame

            #cv2.imshow("Frame", 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 an output video file path has been supplied and the video

            # writer has not been initialized, do so now

#if writer is None:

                # initialize our video writer

            # if the video writer is not None, write the frame to the output

            # video file

            #if writer is not None:

            #    writer.write(frame)

            if(pred==1):

                playsound(r'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=True)

if \_\_name\_\_ == '\_\_main\_\_':

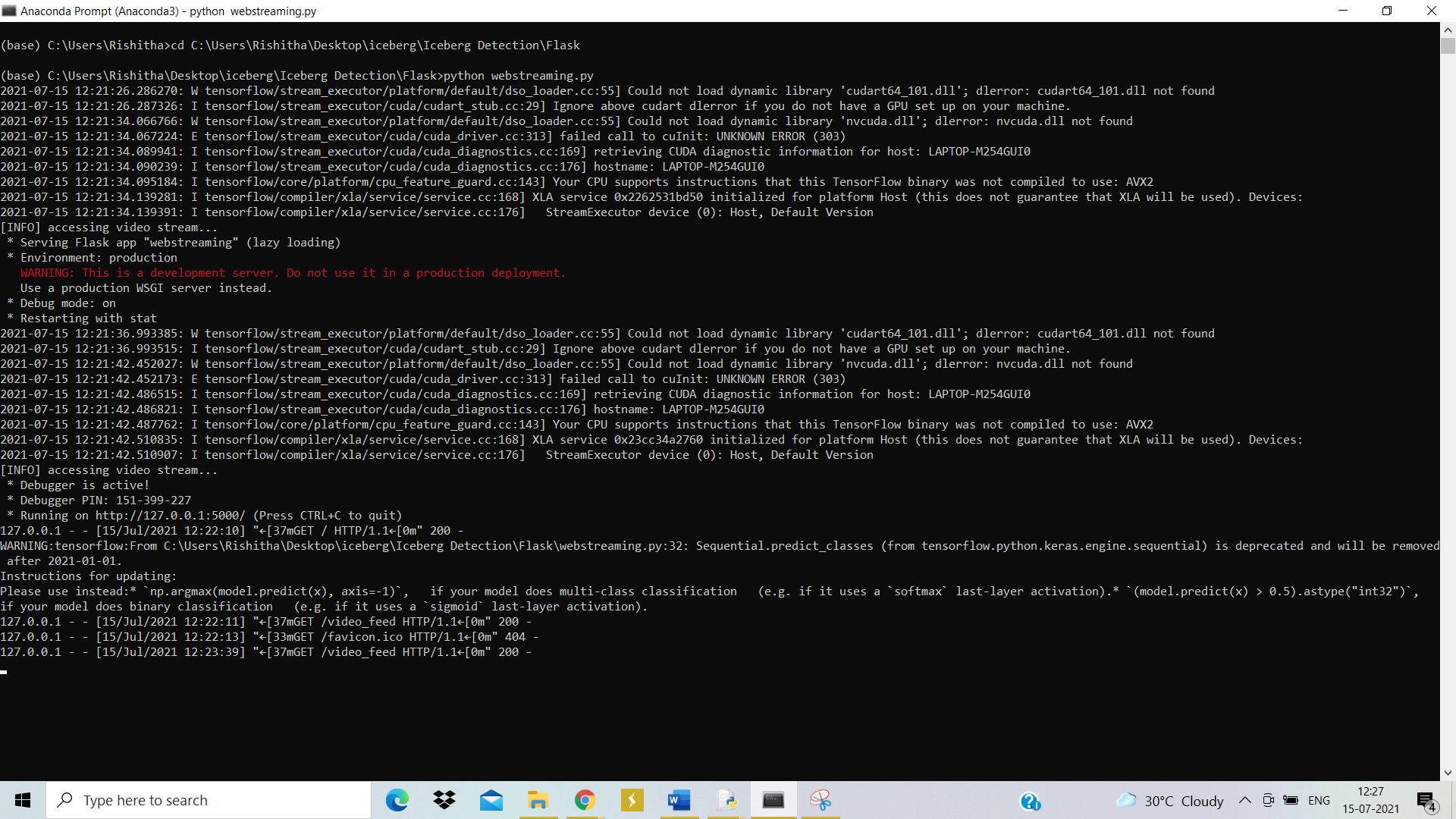
    app.run(debug=True)

**b.UI output Screenshot**

* Application Building

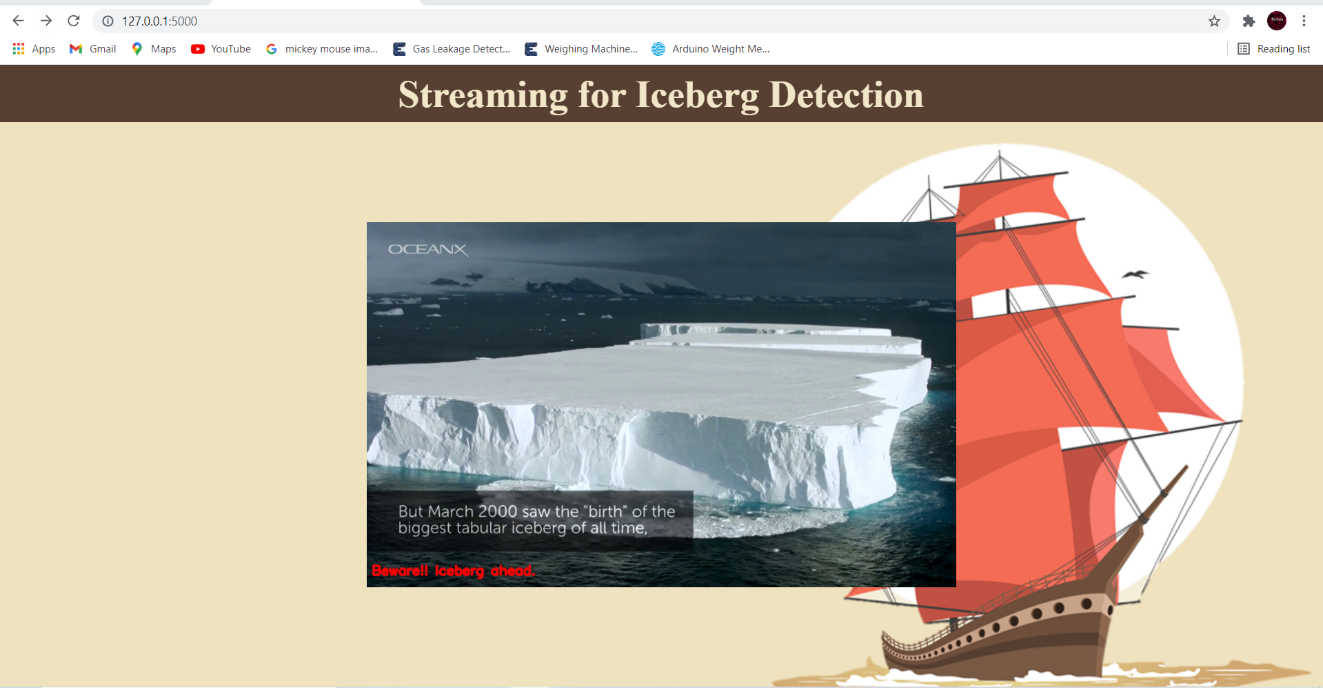
Now we will be building a Flask application that is used for building our UI which in backend can be interfaced to the model to get predictions. Flask application requires an HTML page for Frontend and a Python file for the backend which takes care of the interface with the model.

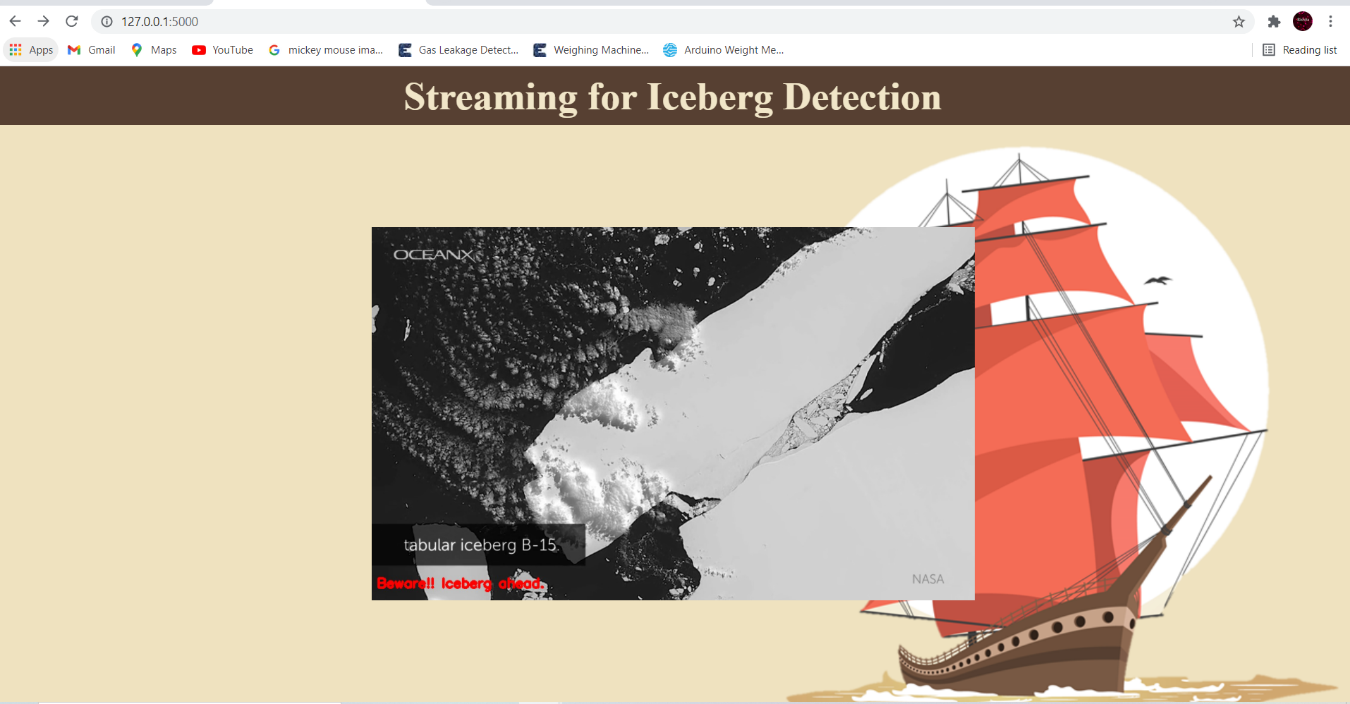
* Open anaconda prompt from the start menu
* Navigate to the folder where your python script is.
* Now type “webstreaming.py” command

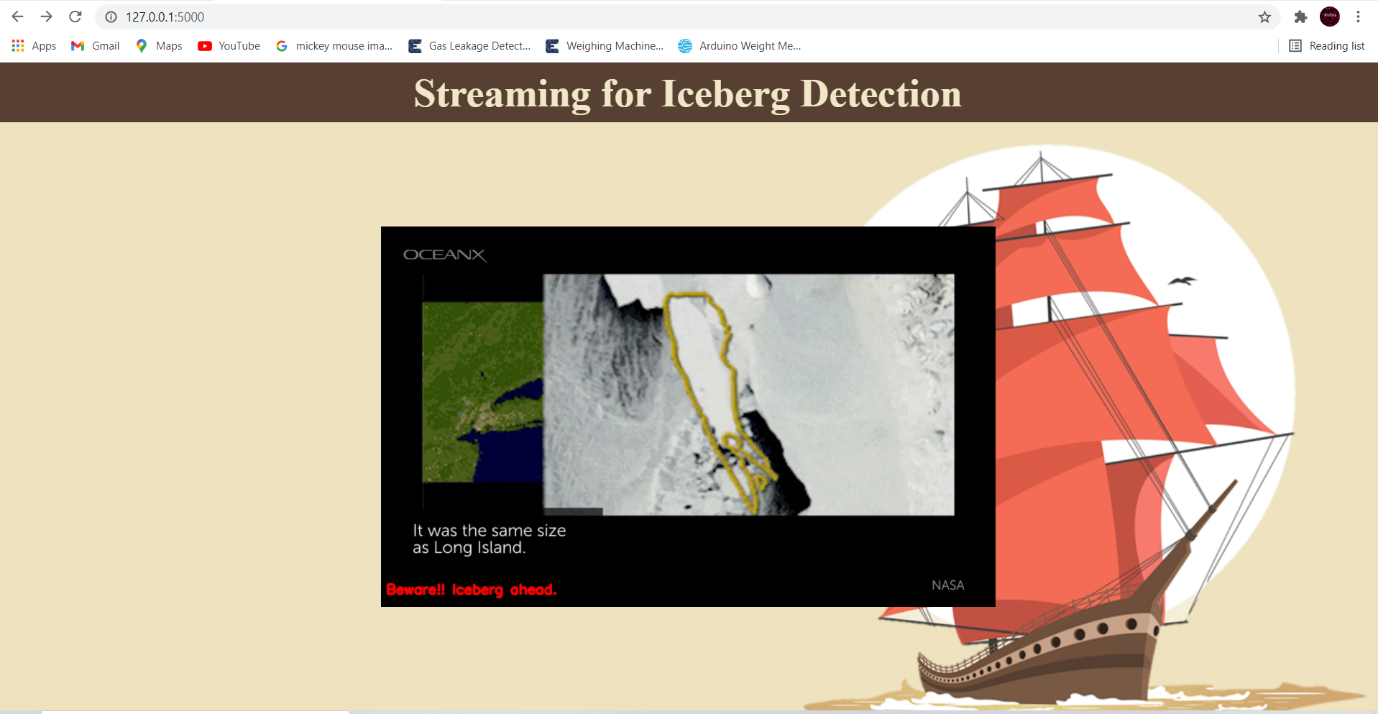
****

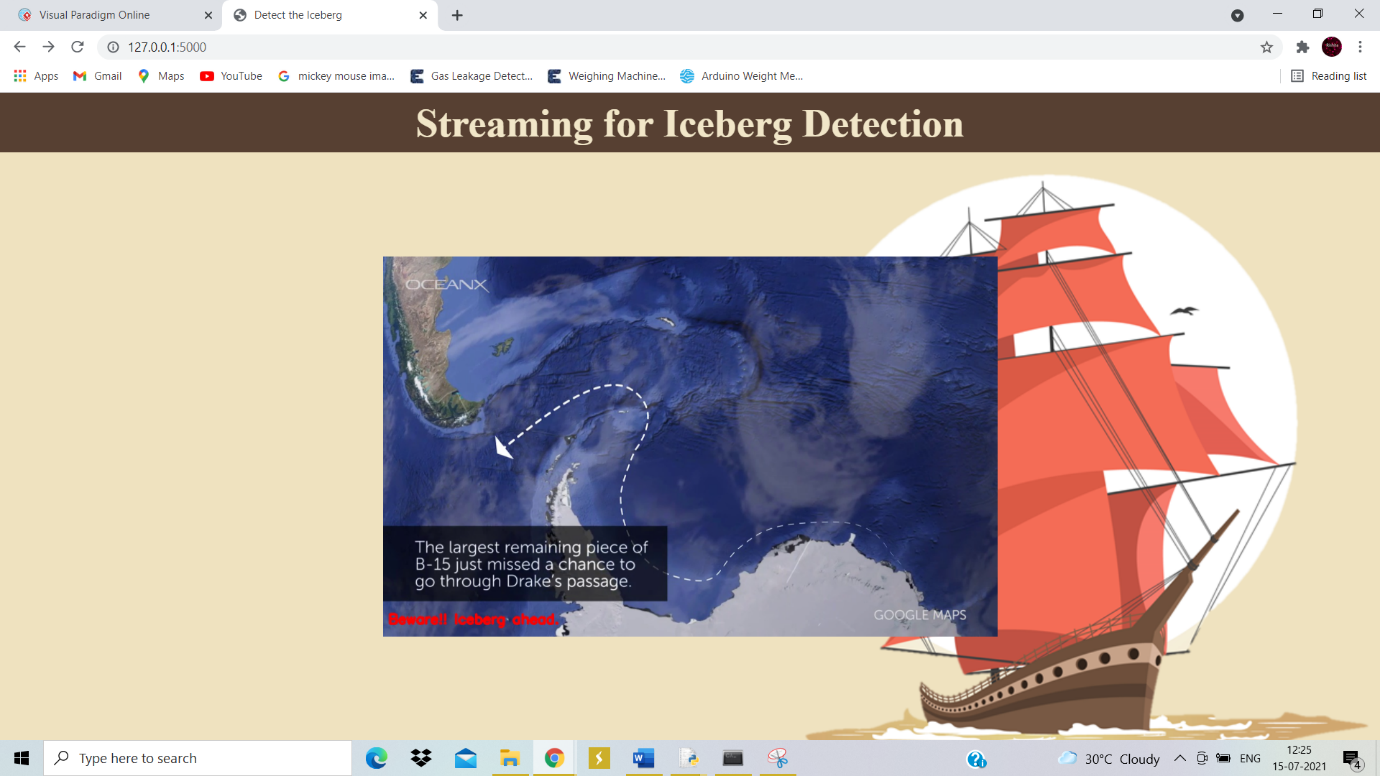
 Navigate to the localhost where you can view your web page and Upload an image and see the predicted output on UI your page and output looks like:

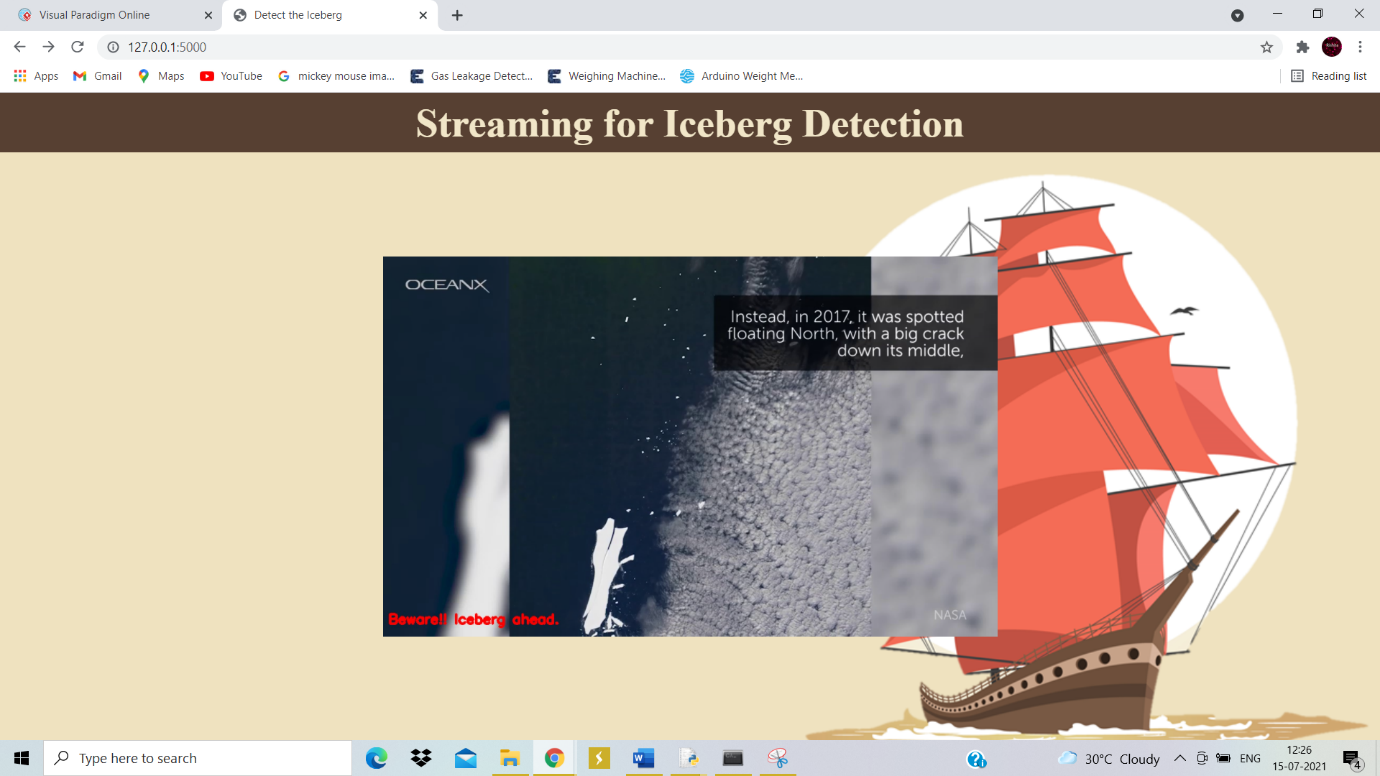
****

****

****

****

****

****

****