Iceberg Detection In Satellite Images Using IBM Watson Studio

Introduction

a. Overview

The polar regions are unique, but fragile ecosystem that are increasingly threatened by changes to our climate. In this project, we develop a model for identifying and measuring icebergs in satellite images. As icebergs represent a significant hazard for shipping and offshore activities, any improvement in the ability to detect and track their progress along the Polar regions would be valuable. Collisions with icebergs pose a threat not just for lives and goods, but also for the environment, since they may result in oil spills or other ecological disasters. The reduction in sea ice across the Arctic has driven an increase in navigating these waters, especially through the North West and North East Passages which promise large reductions in travel time..

b.Purpose

The discovery of an iceberg depends on the alertness of a ship's watchkeepers, and a decaying iceberg poses additional hazards because of its trail of growlers and bergy bits. Although small in size, they have masses that are capable of damaging or sinking ships. As they drop into the sea, icebergs often roll over and lose their snow layers. In a heavy sea, the bergs' smooth wetted ice surfaces produce a low radar cross section. This makes them difficult to discriminate by eye against foam and whitecaps. So, we develop a machine learning model for identifying and measuring icebergs in satellite images.

Literature Survey

a. Existing Problem

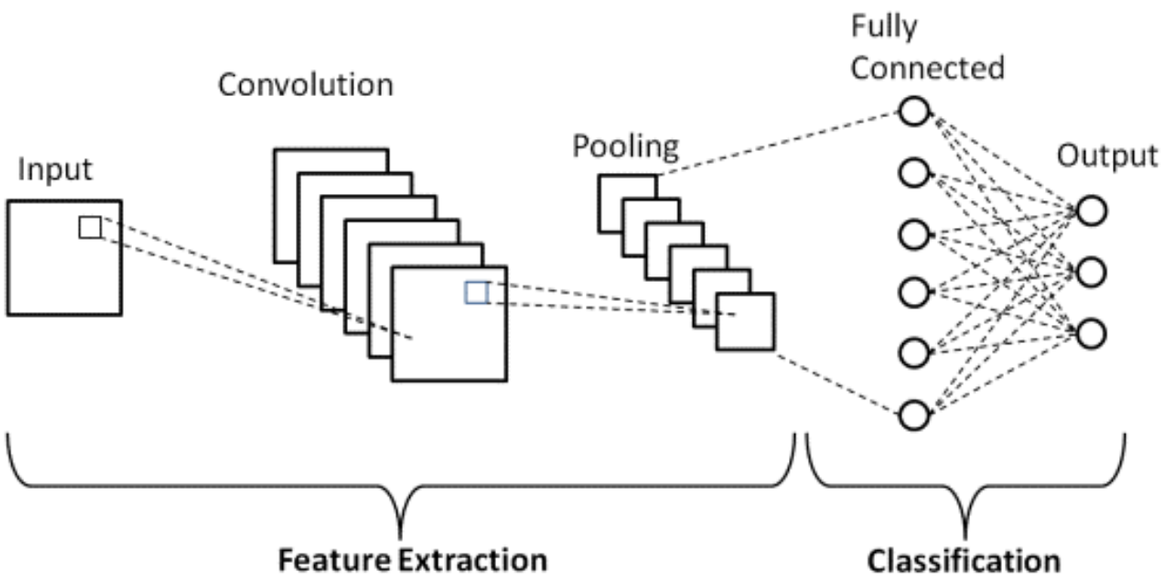
Icebergs present serious hazards for ship navigation and offshore installations. Consequently, there is a large interest to localize them in timely and over vast areas. Because of their independence of cloud cover and daylight, satellite Synthetic Aperture Radar (SAR) images are among the preferred data 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.

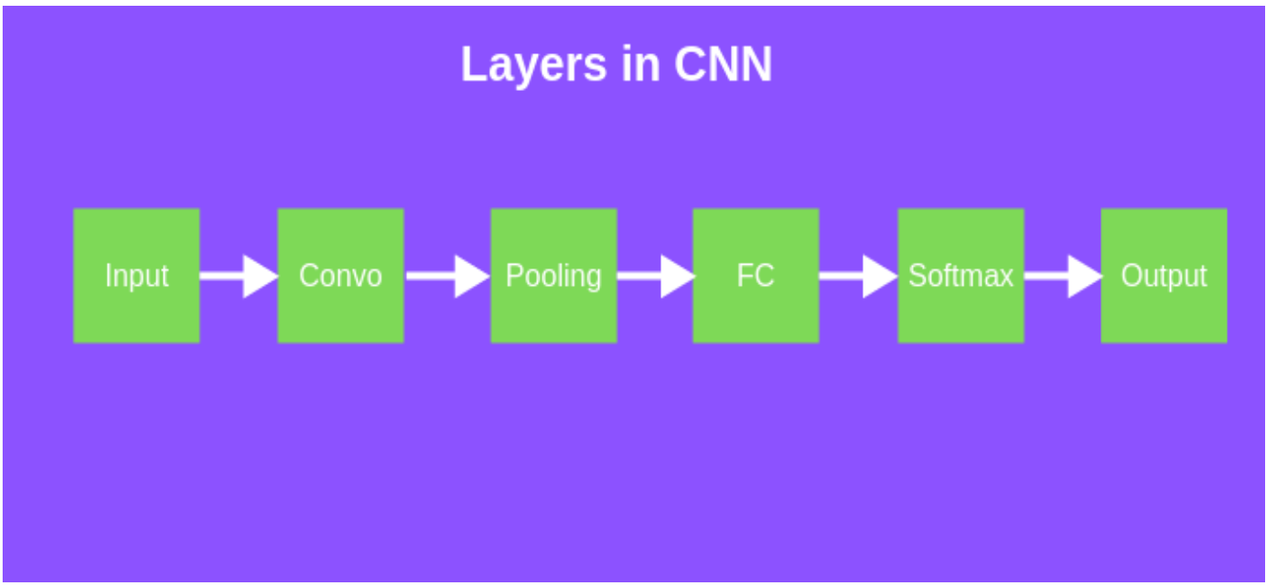
b. Proposed Solution

The main aim of this project is to build a model that automatically identifies whether a remotely sensed target is an iceberg or not. Processed SAR data are characterized by speckle noise, which causes a grainy appearance of the images making the identification of icebergs extremely difficult. 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.

Theoretical Analysis

a. Block Diagram





Convolutional Neural Network comes under a class of Deep Neural Network that can recognize and classify particular features from images and are widely used for analyzing visual images. Their applications range from image and video recognition, image classification, medical image analysis, computer vision and natural language processing. There are three types of layers that make up the CNN which are the convolutional layers, pooling layers, and fully-connected (FC) layers.

b. Software Designing

Convolution Layers

There are three types of layers that make up the CNN which are the convolutional layers, pooling layers, and fully-connected (FC) layers. In addition to these three layers, there are two more important parameters which are the dropout layer and the activation function.

1. Convolutional Layer

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.

2. Pooling Layer

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map.

In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer.

3. Fully Connected Layer

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture. In this, the

input image from the previous layers are flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place.

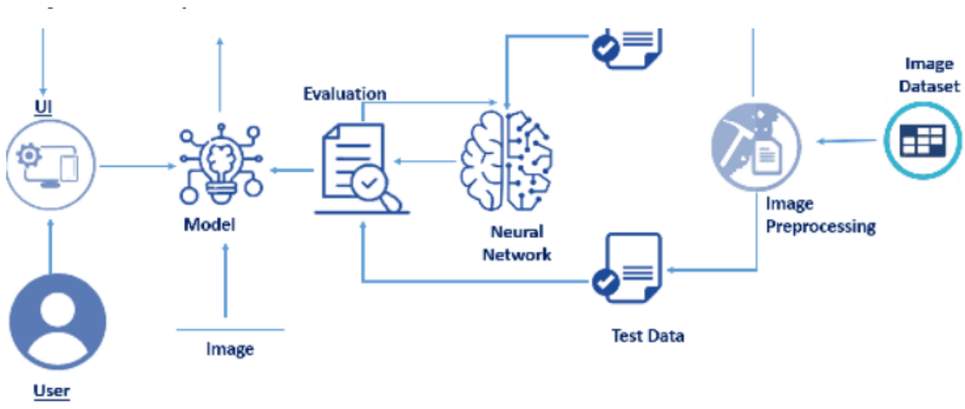
4. Dropout

Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model's performance when used on a new data. To overcome this problem, a dropout layer is utilised wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model. On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network.

5. Activation Functions

Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, Softmax, tanH and the Sigmoid functions. Each of these functions have a specific usage. For a binary classification CNN model, sigmoid and softmax functions are preferred an for a multi-class classification, generally softmax us used.

Flowchart



Each input image will pass it through a series of convolution layers with filters. In order to perceive the same as humans, CNNs have digital colour images that have red-blue-green (RGB) encoding. The Convolutional Layer, Activation Layer, Pooling Layer, and Fully Connected Layer are all interconnected so that CNNs can process and perceive data in order to classify images.

Result

In this CNN model, various convolutional layers were tried, different get back to capacities were applied for iceberg detection in high-goals satellite pictures. Our work can be additionally improved by investigating the pre-handling of our information. In our examination, we dismissed the rate point that may have tremendously impacted our outcome. At long last, we accomplish great precision from examination of chunk of ice discovery in satellite pictures utilizing profound learning procedures and automatically identifying whether a remotely sensed target is an iceberg or not.

Advantages

1. The main advantage of CNN is that, it is computationally efficient. It uses special convolution and pooling operations and performs parameter sharing. This enables CNN models to run on any device, making them universally attractive.CNN is now the go-to model on every image related problem.
2. The main strengths of CNNs are to provide an efficient dense network which performs the prediction or identification etc. efficiently.
3. This project builds a model that automatically identifies whether a remotely sensed target is an iceberg or not which saves lives and billions of dollars in energy infrastructure.

Disdvantages

1. One drawback of CNN model is that it needs large training data and doesn't encode the position and orientation of object.
2. There is always a possibility of wrong prediction as CNN makes predictions by looking at an image and then checking to see if certain components are present in that image or not.

Conclusion

In conclusion, iceberg roots a dangerous risk for transportation, marine oil and gas production plants. Data on Iceberg and Island is significant for atmosphere science and for different marine activities in the sea. The detection and monitoring of the icy objects in the polar zone, which is often dark and cloudy are done with the well-established tool know as synthetic aperture radar (SAR). In this project, the CNN method will be utilized for distinguishing the iceberg from high-resolution satellite images.

Future Scope

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. Inorder to achieve this, we utilize the Convolution Neural Network(CNN) for iceberg detection in high-goals satellite pictures.

Bibliography

1. Marino A, Rulli R, Wesche C, Hajnsek I. A new algorithm for iceberg detection with dual-polarimetric SAR data. In2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS) 2015 Jul 26 (pp. 3446-3449). IEEE.

2. Dierking W, Wesche C. C-band radar polarimetry—Useful for detection of icebergs in sea ice?. IEEE Transactions on Geoscience and Remote Sensing. 2013 Feb 26;52(1):25-37.

3. Howell C, Power D, Lynch M, Dodge K, Bobby P, Randell C, Vachon P, Staples G. Dual polarization detection of ships and icebergs-recent results with ENVISAT ASAR and data simulations of RADARSAT-2. InIGARSS 2008-2008 IEEE International Geoscience and Remote Sensing Symposium 2008 Jul 7 (Vol. 3, pp. III-206). IEEE.

Appendix

a. Source Code

In [1]:
```
from keras.preprocessing.image import ImageDataGenerator
```

In [2]:
```
train_datagen=ImageDataGenerator(rescale= 1./255,zoom_range=0.2,horizontal_flip=True,shear_range=0.2)
```

In [3]:
```
test_datagen=ImageDataGenerator(rescale= 1./255)
```

In [4]:
```
x_train=train_datagen.flow_from_directory(r"C:\Users\maddi\OneDrive\Desktop\SMARTBRIDGE\Iceberg Detection\dataset\train",
                                          target_size=(75,75),
                                          class_mode='binary',
                                          batch_size=32)
```

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

In [5]:
```
x_test=test_datagen.flow_from_directory(r"C:\Users\maddi\OneDrive\Desktop\SMARTBRIDGE\Iceberg Detection\dataset\test",
                                        target_size=(75,75),
                                        class_mode='binary',
                                        batch_size=32)
```

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

In [6]:
```
x_train.class_indices
```

Out[6]:
```
{'Iceberg': 0, 'Ship': 1}
```

In [7]:
```
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
```

In [8]:
```
model=Sequential()
```

In [8]:
```
model=Sequential()
```

In [9]:
```
model.add(Conv2D(64,(3,3),input_shape=(75,75,3),activation='relu'))

```

In [10]:
```
model.add(MaxPooling2D(pool_size=(3,3), strides=(2,2)))
```

In [11]:
```
model.add(Flatten())
```

In [12]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 73, 73, 64)        1792
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 36, 36, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 82944)             0
=================================================================
Total params: 1,792
Trainable params: 1,792
Non-trainable params: 0
_________________________________________________________________
```

In [13]:
```
model.add(Dense(512,activation="relu"))
model.add(Dense(256,activation="relu"))
model.add(Dense(1,activation="sigmoid"))
```

In [14]:
```
model.compile(loss="binary_crossentropy",optimizer='adam',metrics=['accuracy'])
```

In [14]:
```
model.compile(loss="binary_crossentropy",optimizer='adam',metrics=['accuracy'])
```

In [17]:
```
model.fit_generator(x_train,
                    steps_per_epoch=len(x_train),
                    epochs=10,
                    validation_data=x_test,
                    validation_steps=len(x_test))
```

```
Epoch 1/10
41/41 [==============================] - 13s 325ms/step - loss: 0.6015 - accuracy: 0.6659 - val_loss: 0.5430 - val_accuracy: 0.
7719
Epoch 2/10
41/41 [==============================] - 13s 323ms/step - loss: 0.5577 - accuracy: 0.7188 - val_loss: 0.6170 - val_accuracy: 0.
6250
Epoch 3/10
41/41 [==============================] - 13s 311ms/step - loss: 0.5173 - accuracy: 0.7461 - val_loss: 0.4590 - val_accuracy: 0.
7688
Epoch 4/10
41/41 [==============================] - 13s 316ms/step - loss: 0.4408 - accuracy: 0.7804 - val_loss: 0.4808 - val_accuracy: 0.
7906
Epoch 5/10
41/41 [==============================] - 14s 330ms/step - loss: 0.4273 - accuracy: 0.8014 - val_loss: 0.4386 - val_accuracy: 0.
8000
Epoch 6/10
41/41 [==============================] - 14s 336ms/step - loss: 0.4207 - accuracy: 0.7788 - val_loss: 0.4401 - val_accuracy: 0.
8031
Epoch 7/10
41/41 [==============================] - 14s 344ms/step - loss: 0.4417 - accuracy: 0.7819 - val_loss: 0.4102 - val_accuracy: 0.
8219
Epoch 8/10
41/41 [==============================] - 14s 342ms/step - loss: 0.4106 - accuracy: 0.8045 - val_loss: 0.3891 - val_accuracy: 0.
8281
Epoch 9/10
41/41 [==============================] - 14s 349ms/step - loss: 0.3891 - accuracy: 0.8232 - val_loss: 0.3770 - val_accuracy: 0.
8344
Epoch 10/10
41/41 [==============================] - 14s 340ms/step - loss: 0.4072 - accuracy: 0.8185 - val_loss: 0.4598 - val_accuracy: 0.
7781
```

Out[17]:
```
<tensorflow.python.keras.callbacks.History at 0x1c8a87e1f10>
```

In [9]:
```
model.save('iceberg.h5')
```

In [20]:
```
import numpy as np
import cv2
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

In [21]:
```
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)
```

In [22]:
```
frame=cv2.imread(r"C:\Users\maddi\OneDrive\Desktop\SMARTBRIDGE\Iceberg Detection\dataset\test\Ship\f263.png")
data=detect(frame)
```

```
[[0.74876046]]
[[1]]
```

In [ ]:
```

```

b.UI Output Screenshot