Binary Image Classification System to Identify Oil Spills in Ocean Imagery

Submitted to the Office of Graduate and Professional Studies of

UNIVERSITY OF HOUSTON

in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING DATA SCIENCE

UNIVERSITY OF HOUSTON

By:

Suchendra Reddy Yalamakuru - 2313042

Shreyas Mysore Narayanan - 2307777

Surya Vardhan Reddy Puchalapalli - 2311061

Chinta Bindhu sri - 2316871

Mappavarapu Madhu Sudhan Rao - 2281268

Sreeja Reddy Nannuri - 2205712

Under the Guidance of

GUAN QIN

2024

Table of Content

Abstract

This report presents a binary image classification system implemented using deep learning and machine learning models for classifying ocean imagery collected from various sources like drone images, news articles, areal imaging and satellite imaging etc. It consists of a detailed analysis of seven models built and evaluated on the basis of their prediction accuracy. Three out of seven models built are selected and their performance on test set is compared to calculate the model accuracy. The present classification system used advanced CNN models built using libraries like TensorFlow, Keras and OpenCV etc alongside with usual machine learning models like Support Vector Classifier, Random Forest Classifier and Boosting models etc. Our methodology employs OpenCv2 & PIL for image pre-processing and data augmentation techniques to enhance model performance. The results demonstrate the efficiency and accuracy of the system in effectively classifying the spill or non-spill images.

1 Introduction

Oil is a key driving force of national economies around the world as more than one-third of worlds total energy production is from oil. Although very few nations are blessed with natural reserves of oil & gas, improved transportation facilities alongside the canal systems helped reaching the ever increasing demand of oil all over the world. Pipeline distribution systems being restricted by lot many constraints like geopolitical situations, distance constraints, high maintenance costs and failure risks etc., is not a feasible option in most of the situations, leaving most of the supply chain dependent on physical transportation of resources. The transportation of resources happens through various modes like road, air and water. However transportation through water remains the cost effective and efficient option for a huge chunk of world (1.83 billion metric tons of crude oil was transported by sea in 2021).

Transportation of Oil through water not only has significant advantages in various aspects of distribution but also paves way for a lot other problems too. The unpredictable weather conditions of seas & oceans, lack of proper network means for efficient communication and navigation etc increases the risks of damage and losses during transportation. Oil spills results in a huge loss of resources owing to huge loses in distribution. (The total volume of oil lost to the environment from tanker spills in 2022 was approximately 15,000 tonnes)

On the other hand, Oil spills are also a major threat to marine life which in turn effects the human populations depend on them. According to statistics, an estimate of 250,000 birds, 2,800 sea otters, 302 harbor seals, 250 bald eagles, 22 whales, and billions of salmon and herring eggs were killed in an accident to oil tanker Exxon-Valdez in 1989. 11 million gallons of oil spilled in the accident spread over 470 miles has affected 1300 miles of shoreline.

Detecting and responding to oil spills in a timely and efficient manner is crucial for several reasons. First, timely detection of oil spills helps minimizing the environmental impact of the oil spill. It also helps in mitigating the loss of resources in turn reducing losses. Second, an efficient system of detection may be used in real time monitoring of the routes of transportation and helps in effective decision making to avoid accidents and spillages. It may also help in exploring new and safe routes to destinations. Third, the present system will help in developing a proper communication channel about accidents/damages to transports and timely dispatching help or rescue teams to the actual spot. Finally, effective detection of oil spillages can support environmental and safety regulations by ensuring safe transportation of resources and reduced oil spills.

1.1 Literature Review

Ocean oil spills seriously impact the environment and its inhabitants. Automated technologies for spotting and categorizing oil spills in satellite and aerial photos have been developed to lessen their effects. One of the most popular methods for detecting oil spills is the Binary Image Classification System, which involves analysing images to determine the presence or absence of oil.

For detecting oil spills in satellite images a Bayesian classification and morphological filtering has been developed using support vector machine (SVM) classifier to classify image pixels as oil or non-oil. Morphological filters were then applied to the classified images to remove false positives [1]. Texture features and a support vector machine (SVM) classifier has been used to classify image pixels as oil or non-oil for detecting oil spills in synthetic aperture radar (SAR) images [2]. A hybrid approach has been developed for detecting oil spills in aerial images by using a combination of color thresholding, morphological operations, and blob analysis to detect potential oil spills followed by a SVM classifier to classify the detected spills as oil or non-oil[3].

A deep learning approach for detecting oil spills in satellite images. Convolutional neural network (CNN) has been used to classify images as oil or non-oil. Three convolutional layers, each followed by a max-pooling layer, and two fully connected layers has been implemented and the output layer consisted of a single node with a sigmoid activation function, which outputted the probability of the input image being an oil spill. [4].

The performance of different deep learning techniques has been compared for detecting oil spills in satellite images. The performance of convolutional neural networks (CNN), residual neural networks (ResNet), and dense neural networks (DenseNet) has been compared on a dataset of 2000 satellite images [5]. A method for detecting marine oil spills using a VGG16 model has been proposed and a comparison has been made with traditional image processing techniques and other deep learning models, such as AlexNet and ResNet, and showed that the VGG16 model outperformed them in terms of accuracy and F1 score[6].This indicates the potential of using deep learning, particularly the VGG16 model, for automated oil spill detection in satellite images, which can help in early detection and response to oil spills, leading to more effective mitigation of their environmental impact.

A deep learning approach for detecting oil spills in satellite images using a combination of convolutional neural network (CNN) and long short-term memory (LSTM) network has been proposed [7]. The features extracted by the CNN were fed into an LSTM network, which is a type of recurrent neural network (RNN) that can capture temporal dependencies in sequential data. The LSTM network was used to classify the images as oil or non-oil. A deep learningbased approach for oil spill detection using unmanned aerial vehicle (UAV) images has been proposed where a contrast stretching technique has been used to enhance the contrast of the images and reduce the effects of shadows and clouds[8]. Here CNN architecture called Inception-v3 has been used along with VGG16, ResNet50. Features have been extracted from the last fully connected layer of each model and concatenated them to form a feature vector.

In a study by Abdullah.R & Marsono M.N [9], an efficient deep learning method for oil spill detection has been proposed by using Sentinel-2 satellite imagery. The proposed method used a CNN architecture called U-Net. In another study a deep learning-based approach for detecting oil spills in satellite images using multiscale features has been proposed[10]. This method utilized multiscale convolutional neural networks (MCNNs) to extract features from different scales of the input image. The extracted features were then fed into a fully connected network for classification.

In conclusion, the detection of oil spills in ocean imagery has been a critical issue to reduce environmental impact. Binary Image Classification System has been the most popular method for detecting oil spills, which involves analyzing images to determine the presence or absence of oil. Several studies have proposed deep learning techniques for detecting oil spills, which outperformed traditional image processing techniques and other deep learning models, such as AlexNet and ResNet. Among the deep learning models, VGG16 has shown better accuracy and F1 score for automated oil spill detection in satellite images. Moreover, the combination of CNN and LSTM networks has also been proposed to capture temporal dependencies in sequential data for oil spill detection. These automated technologies for oil spill detection can help in early detection and response to oil spills, leading to more effective mitigation of their environmental impact.

1.2 Objective

The objective of this project is to develop a binary image classification system that can accurately detect and identify oil spills in ocean imagery. This system will use machine learning techniques to automatically classify images as either containing or not containing an oil spill. The project aims in providing a useful tool for the oil and gas industry to detect and respond to oil spills in a timely and efficient manner, reducing the environmental impact and financial cost of oil spills.

Detailed Note

Oil spills are a major environmental and economic problem for the oil and gas industry. They can cause significant damage to the marine ecosystem, harm wildlife, and have long-term impacts on the local economy. Rapid detection and response to oil spills is critical to minimizing the environmental and financial impact of these events. However, traditional methods of detecting oil spills can be slow and resource intensive.

In recent years, advances in machine learning and computer vision have led to the development of automated image classification systems that can detect and identify objects in images with a high degree of accuracy. These systems have been applied to a range of fields, including medical imaging, robotics, and self-driving cars. In the oil and gas industry, image classification systems have the potential to revolutionize the way oil spills are detected and responded to.

The proposed binary image classification system will use the VGG-19 convolutional neural network model to classify ocean images as containing or not containing an oil spill. The system will be trained on a dataset of labeled ocean imagery, with images containing oil spills labeled as positive examples and images without oil spills labeled as negative examples. Data augmentation techniques will be used to increase the diversity of the training data and prevent overfitting.

The trained model will then be applied to new, unlabeled ocean imagery to classify the images as containing or not containing an oil spill. The system will output a confidence score for each classification, indicating the degree of certainty that an image contains an oil spill.

The business value of this project is significant. The ability to quickly and accurately detect oil spills in ocean imagery can help oil and gas companies to respond to spills in a timely and efficient manner, reducing the environmental and financial impact of these events. The system could also be used to monitor offshore oil drilling operations and identify potential spills before they become significant problems. Overall, the binary image classification system has the potential to improve the safety, sustainability, and profitability of the oil and gas industry.

2 Data

2.1 Dataset Background and Quality

The dataset is obtained from the Kaggle repository. The Dataset consists of RGB images collected from various sources like Google Earth, Ocean Surveys, News articles and blogs etc. It contains ocean imagery of spill and non-spill sites captures through drone, satellite pictures and real time photography. The images are separated into train, valid, and test sets. The train set contains 2800 images, out of which 1400 are oil spills and 1400 are non-spills (balanced dataset). The valid and test sets each contain 300 images. The images are in RGB format with varied sizes and resolutions. The dataset was obtained from the following link:

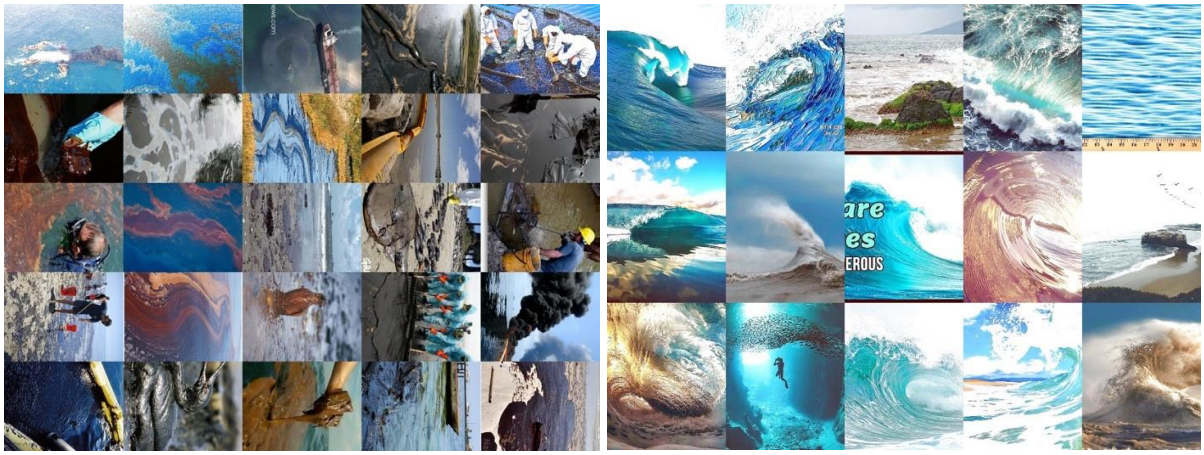


Figure 1. Spill and No Spill image samples from Train set

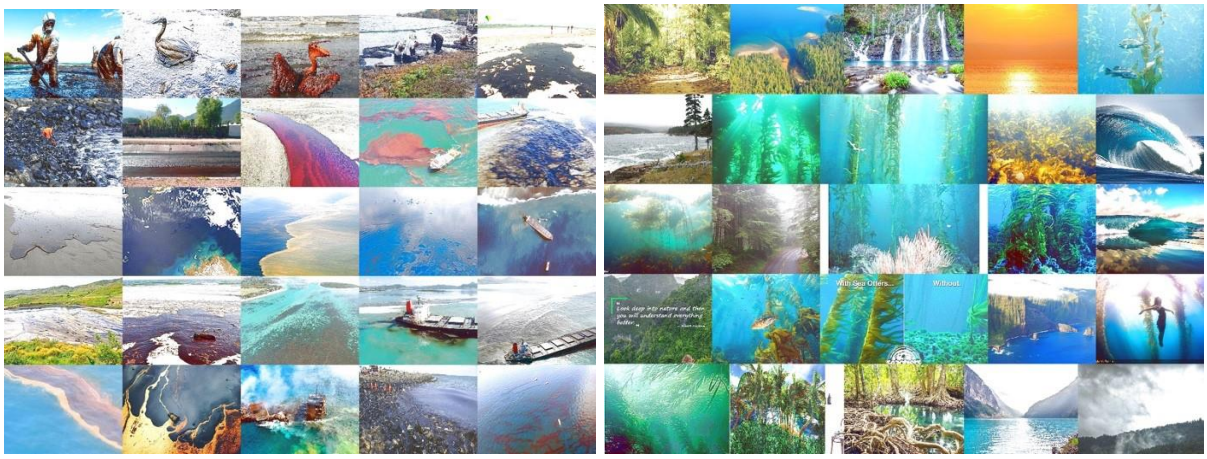


Figure 2. Spill and No spill Imagery in Validation set

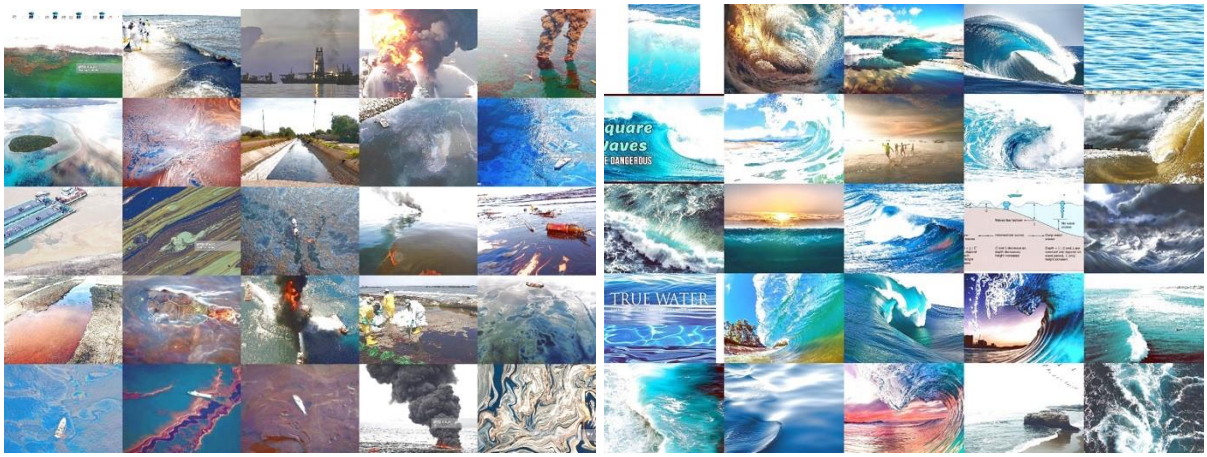


Figure 3. Spill and No Spill Imagery in Test Set

2.2 Data Processing, Wrangling, and EDA

In order to effectively train our classification models, we performed a series of data processing, wrangling, and exploratory data analysis (EDA) steps.

As the dataset is divided into multiple folders and subfolders, First we have collected all the directories and the paths of our data sets and extracted the address of each image into a readable format. Then we have visualized few sample images to look into the format of images.

Figure 4. Non spill training image before resizing

Figure 5. Spill training image before resizing

From the visualization we found that the images are of different sizes and resolutions. Next, we analysed the width and height distributions of the figure, which gave us a better understanding of the size variations of images in the dataset.

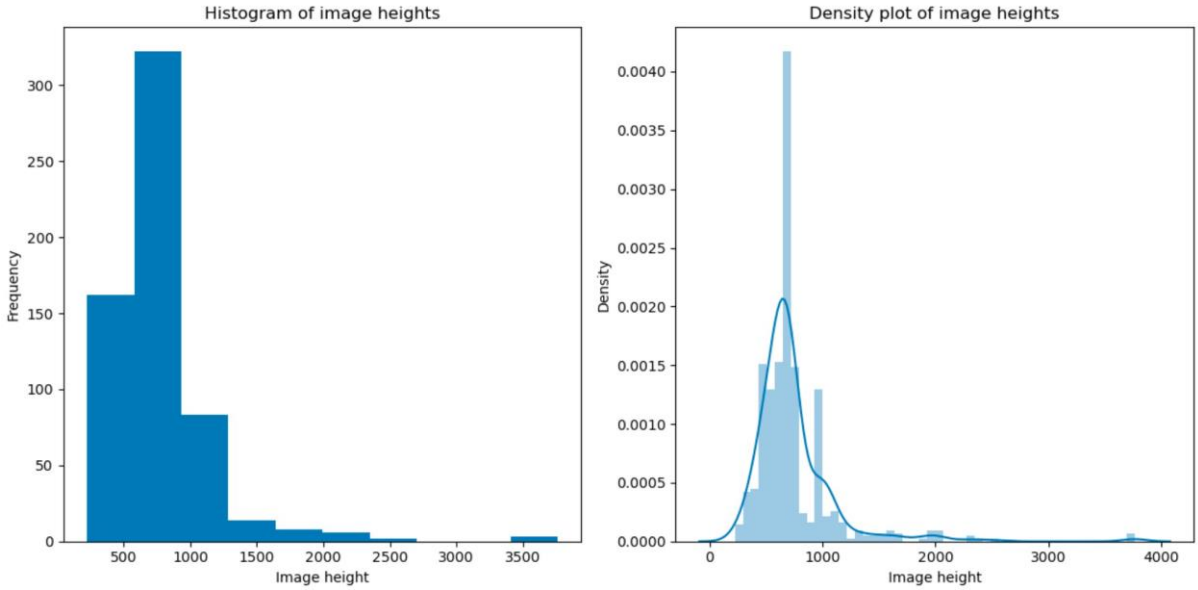


Figure 6. Distribution of image Heights for Training Data

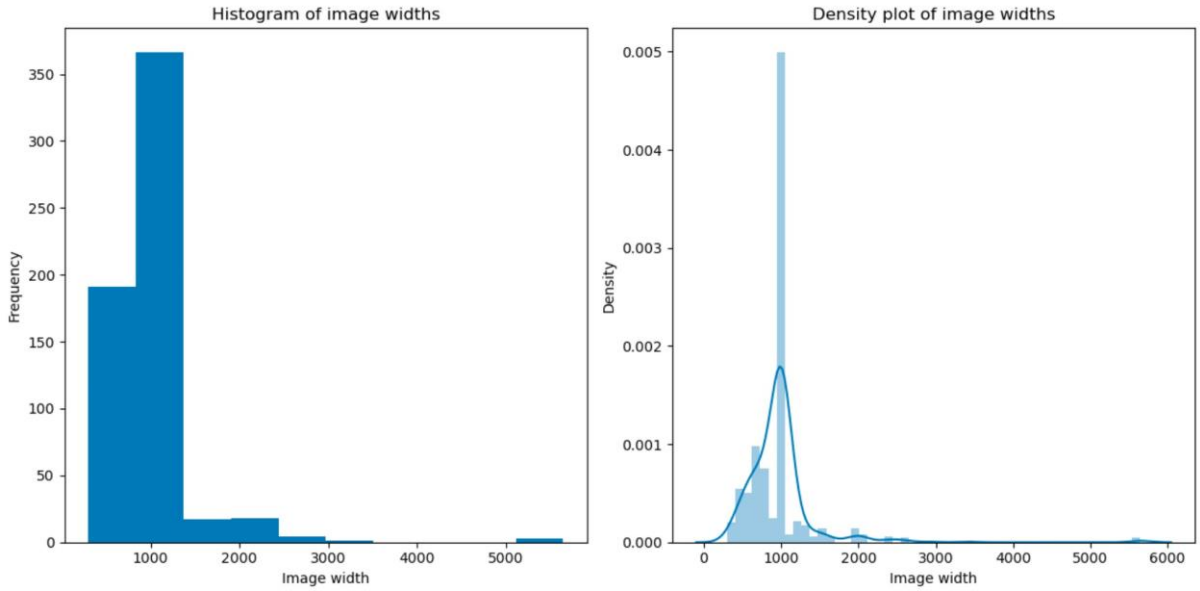


Figure 7. Distribution of image Widths for Training Data

From the height and width distributions we observed that majority of the images have resolution of ~ 1000 x 1000. So, we have decided to crop all the images in to a common size such that the image processing becomes easy going forward into further analysis.

We have visualized the images after resizing to make sure all the images in train, validation and test set are brought into similar dimenesions.

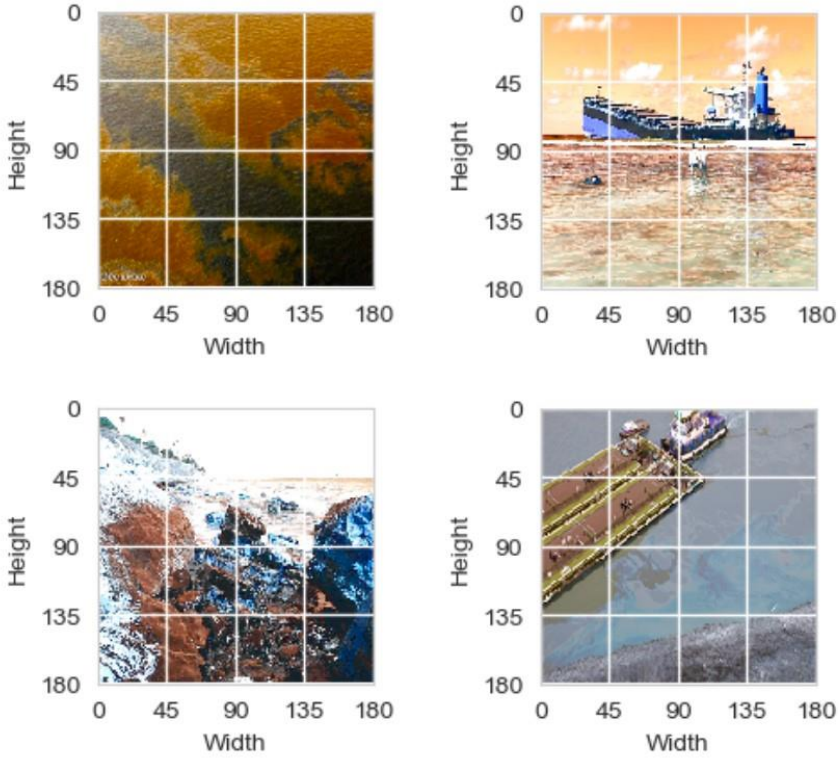


Figure 8. Resized images from training data

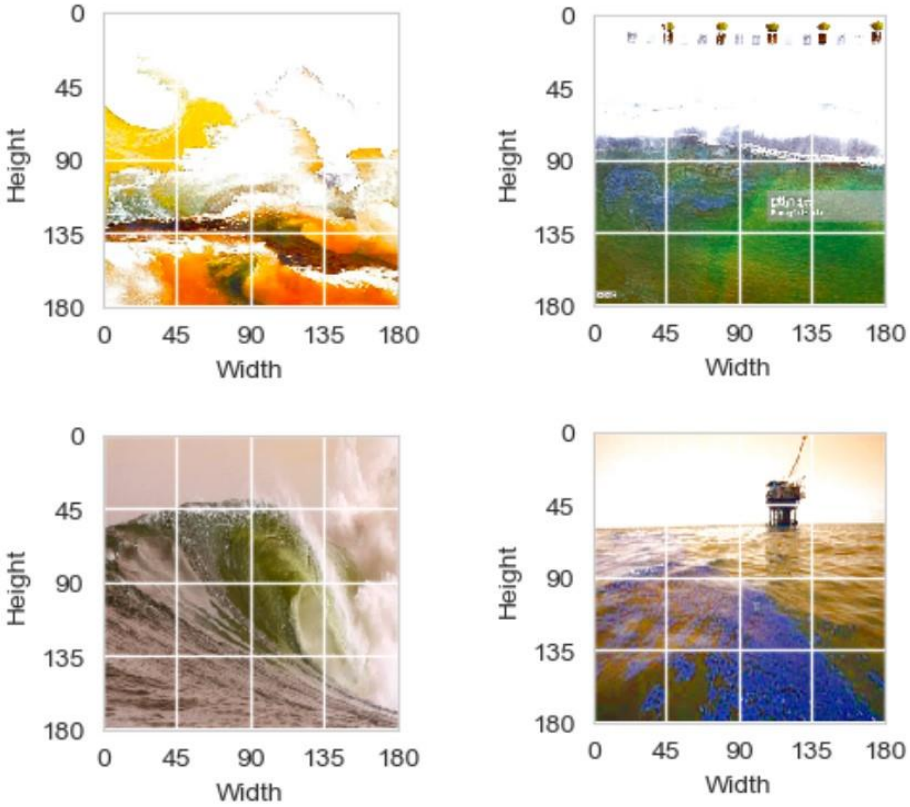


Figure 9. Resized images from validation and testing data

Overall, the EDA process provided valuable insights into the dataset's characteristics, enabling us to make informed decisions when developing and optimizing our model for classifying the images into spill and non-spill classes.

3 Methodology

3.1 Methods Description, Workflow, Model Building, Training, Testing, and Prediction

The methodology for this project involves a series of steps to develop, train, test, and evaluate the classifiers. The overall workflow consists of the following steps:

- Data Pre-processing
- Data Augmentation
- Model Building
- Model Selection

3.2 Data Pre-processing

Before training the models, we performed data pre-processing to ensure compatibility with the model's input requirements. This process involved resizing of images based on the insights form the heights and width distributions demonstrated above.

```
# Scaling  the data
x_train = np.array(X_train) / 255
x_val = np.array(X_valid) / 255
x_test=np.array(X_test)/255

x_train.reshape(-1, 180,180,3)
y_train = np.array(y_train)

x_val.reshape(-1, 180,180,3)
y_val = np.array(y_valid)

x_test.reshape(-1, 180,180,3)
y_test = np.array(y_test)
```

Figure 10. Resizing images into a fixed size

3.3 Data Augmentation

To improve the generalization of our model and increase the diversity of our dataset, we performed data augmentation. It introduces variations to the existing dataset to increase its diversity, prevent overfitting and improves the generalization of the model. We have applied various transformations like horizontal & vertical flipping, image rotation to a random degree and width & height shift range variations. These transformations were applied randomly to the training images to create additional training samples.

```
#Data Augmentation
datagen = ImageDataGenerator(
       featurewise_center=False,  # set input mean to 0 over the dataset
      samplewise_center=False,  # set each sample mean to 0
       featurewise_std_normalization=False,  # divide inputs by std of the dataset
       samplewise_std_normalization=False,  # divide each input by its std
       zca_whitening=False,  # apply ZCA whitening
      rotation_range = 30,  # randomly rotate images in the range (degrees, 0 to 180)
      width_shift_range=0.1,  # randomly shift images horizontally (fraction of total width)
    height_shift_range=0.1,  # randomly shift images vertically (fraction of total height)
       horizontal_flip = True,  # randomly flip images vertical_flip=False)  # randomly flip images
)
datagen.fit(x_train)
datagen.fit(x_val)
datagen.fit(x_test)
```

Figure 11. Data Augmentation

3.4 Model Building

We have implemented seven Deep Learning and Machine Learning Models. Out of which the best models were selected in model selection based on their performance on validation set.

3.4.1 VGG19 Model (CNN based)

```
vgg19 = VGG19(weights='imagenet',include_top=False,input_shape=(180,180,3))
for layer in vgg19.layers[:19]:
    layer.trainable = False

model = Sequential()
model.add(vgg19)
model.add(MaxPool2D((2,2),strides=2))
model.add(Flatten())
model.add(Dense(2,activation='sigmoid'))
model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg19 (Functional)          (None, 5, 5, 512)         20024384

 max_pooling2d_7 (MaxPooling  (None, 2, 2, 512)        0
 2D)

 flatten_7 (Flatten)         (None, 2048)              0

 dense_7 (Dense)             (None, 2)                 4098

=================================================================
Total params: 20,028,482
Trainable params: 4,723,714
Non-trainable params: 15,304,768
_________________________________________________________________
```

Figure 12. VGG19 Model building

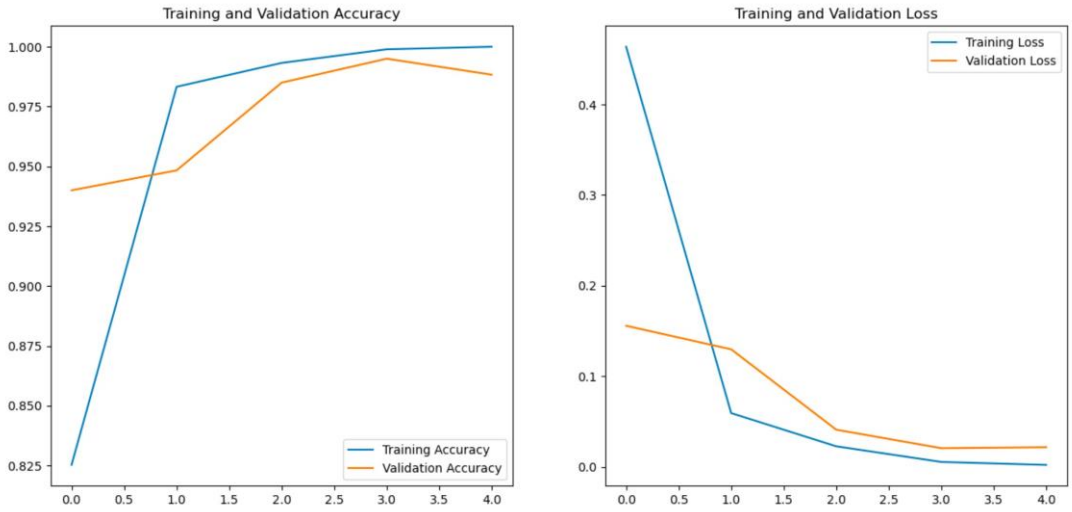


Figure 13. Performance evaluation of VGG19

```
#Model Evaluation using confusion matrix
from sklearn.metrics import classification_report
pred = np.argmax(y_pred_valid, axis=1)
pred = pred.reshape(1, -1)[0]
report = classification_report(y_valid, pred, target_names=['Non Oil Spill(Class 0)', 'Oil Spill (Class 1)'])
print("validation data")
print(report)
```

```
validation data
                      precision    recall  f1-score   support

Non Oil Spill(Class 0)       0.99      1.00      1.00       300
   Oil Spill (Class 1)       1.00      0.99      0.99       300

              accuracy                           0.99       600
             macro avg       1.00      0.99      0.99       600
          weighted avg       1.00      0.99      0.99       600
```

Figure 14. Model Evaluation on validation set of VGG19 model

3.4.2 Non Linear Kernel SVC Model

Model Building

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
svm_model = SVC(kernel='rbf', C=1.0, gamma='scale')
```

```
# Train the SVM model
svm_model.fit(X_train_flat, y_train)
```

```
SVC()
```

```
# Predict the labels for validation data
y_valid_pred = svm_model.predict(X_valid_flat)
```

Figure 15. Non Linear Kernel SVC model building

```
#Model Evaluation with classification report
print("Svm_Non-Linear-Validation Data:")
print(classification_report(y_valid, y_valid_pred, target_names=['Non Spil Oil(Class 0)', 'Oil Spill(Class 1)']))
```

```
Svm_Non-Linear-Validation Data:
                      precision    recall  f1-score   support

Non Spil Oil(Class 0)       0.88      0.98      0.93       300
   Oil Spill(Class 1)       0.98      0.86      0.92       300

            accuracy                           0.92       600
           macro avg       0.93      0.92      0.92       600
        weighted avg       0.93      0.92      0.92       600
```

Figure 16. Non Linear Kernel SVC model evaluation

3.4.3 Linear Kernel SVC model

```
svm_model_linear = SVC(kernel='linear')
svm_model_linear.fit(X_train_flat, y_train)
```

```
SVC(kernel='linear')
```

```
# Predict the labels for validation data
y_valid_pred = svm_model_linear.predict(X_valid_flat)
print("Svm-Linear-Validation Data:")
print(classification_report(y_valid, y_valid_pred, target_names=['Non Spil Oil(Class 0)', 'Oil Spill(Class 1)']))
```

```
Svm-Linear-Validation Data:
                      precision    recall  f1-score   support

Non Spil Oil(Class 0)       1.00      1.00      1.00       300
   Oil Spill(Class 1)       1.00      1.00      1.00       300

            accuracy                           1.00       600
           macro avg       1.00      1.00      1.00       600
        weighted avg       1.00      1.00      1.00       600
```

Figure 17. Linear Kernel SVC model Building and Evaluation

3.4.4 Logistic Regression Model

```
# Create a Logistic Regression model
logreg_model = LogisticRegression()
logreg_model.fit(X_train_flat, y_train)
```

```
LogisticRegression()
```

```
# Predict the labels for validation data
y_valid_pred = logreg_model.predict(X_valid_flat)
print("Logistic_Regression-Validation Data:")
print(classification_report(y_valid, y_valid_pred, target_names=['Non Spill Oil(Class 0)', 'Spill Oil(Class 1)']))
```

```
Logistic_Regression-Validation Data:
                       precision    recall  f1-score   support

Non Spill Oil(Class 0)       1.00      1.00      1.00       300
    Spill Oil(Class 1)       1.00      1.00      1.00       300

              accuracy                           1.00       600
             macro avg       1.00      1.00      1.00       600
          weighted avg       1.00      1.00      1.00       600
```

Figure 18. Logistic regression model Building and Evaluation

3.4.5 Random Forest Classifier Model

```
rf_model = RandomForestClassifier()
rf_model.fit(X_train_flat, y_train)
```

```
RandomForestClassifier()
```

```
# Predict the labels for validation data
y_valid_pred = rf_model.predict(X_valid_flat)
print("Random_Forest-Validation Data:")
print(classification_report(y_valid, y_valid_pred, target_names=['Non Spill Oil(Class 0)', 'Spill Oil(Class 1)']))
```

```
Random_Forest-Validation Data:
                       precision    recall  f1-score   support

Non Spill Oil(Class 0)       0.80      0.97      0.87       300
    Spill Oil(Class 1)       0.96      0.75      0.84       300

              accuracy                           0.86       600
             macro avg       0.88      0.86      0.86       600
          weighted avg       0.88      0.86      0.86       600
```

Figure 19. Random Forest Classifier model Building and Evaluation

3.4.6 Gradient Boosting Classifier Model

```
gb_model = GradientBoostingClassifier()
gb_model.fit(X_train_flat, y_train)
```

```
GradientBoostingClassifier()
```

```
# Make predictions and evaluate
y_valid_pred = gb_model.predict(X_valid_flat)
print(" Gradient_boosting Classifier_Validation Data:")
print(classification_report(y_valid, y_valid_pred, target_names=['Non Spill Oil(Class 0)', 'Spill Oil(Class 1)']))
```

```
 Gradient_boosting Classifier_Validation Data:
                       precision    recall  f1-score   support

Non Spill Oil(Class 0)       0.79      0.97      0.87       300
    Spill Oil(Class 1)       0.97      0.75      0.84       300

              accuracy                           0.86       600
             macro avg       0.88      0.86      0.86       600
          weighted avg       0.88      0.86      0.86       600
```

Figure 20. Gradient boosting Classifier Model building and evaluation

3.4.7 XG-Boosting Classifier Model

```
xgb_model = XGBClassifier()
xgb_model.fit(X_train_flat, y_train)
```

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              predictor=None, random_state=None, ...)
```

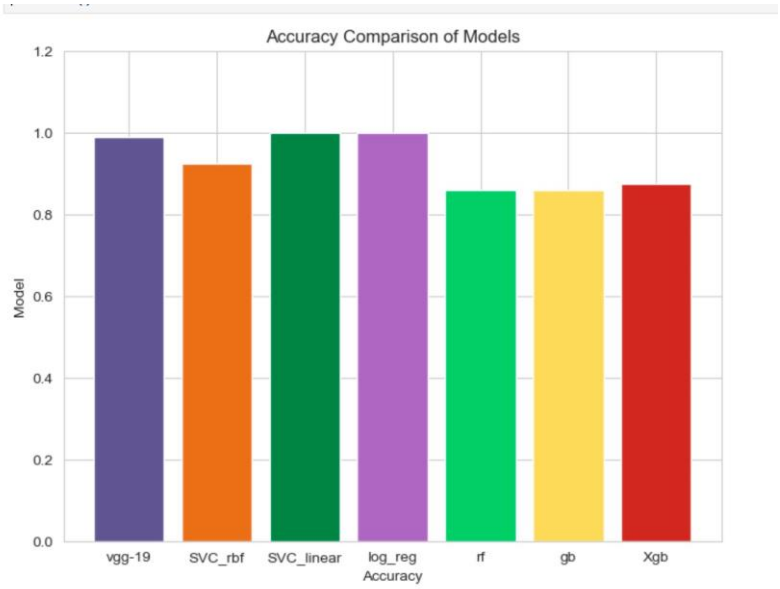
```
# Make predictions and evaluate
y_valid_pred = xgb_model.predict(X_valid_flat)
print("Extreme_gradient_Validation Data:")
print(classification_report(y_valid, y_valid_pred, target_names=['Non Spill Oil(Class 0)', 'Spill Oil(Class 1)']))
```

```
Extreme_gradient_Validation Data:
                      precision    recall  f1-score   support

Non Spill Oil(Class 0)       0.82      0.97      0.89       300
    Spill Oil(Class 1)       0.96      0.78      0.86       300

              accuracy                           0.88       600
             macro avg       0.89      0.88      0.87       600
          weighted avg       0.89      0.88      0.87       600
```

Figure 21. *XG- boosting Classifier Model building and evaluation*

3.5 Model Selection

After analysing the performance evaluation of all the seven models from above, we observed that all the models performed pretty well on validation data. However, few models have done exceptionally well in almost all the categories of metrics like accuracy, precision, recall and f-score. So we have selected three best performing models out of seven models built. As all three models are performing exceptionally well there is every chance that they may perform differently on test set. So, we have performed their performance on test set to check if they over fit.



df

	Accuracy	Model
0	0.99	vgg-19
1	0.92	SVC_rbf
2	1.00	SVC_linear
3	1.00	log_reg
4	0.86	rf
5	0.86	gb
6	0.88	Xgb

Figure 22. Comparing performance of all seven models

```
# Predict the labels for Test data for SVM_linear
accuracy_test = []
y_test_pred = svm_model_linear.predict(X_test_flat)
print("SVM_LINEAR-Test Data:")
print(classification_report(y_test, y_test_pred, target_names=['Non Spil Oil(Class 0)', 'Oil Spill(Class 1)']))
```

```
SVM_LINEAR-Test Data:
                      precision    recall  f1-score   support

Non Spil Oil(Class 0)       1.00      1.00      1.00       300
   Oil Spill(Class 1)       1.00      1.00      1.00       300

             accuracy                           1.00       600
            macro avg       1.00      1.00      1.00       600
         weighted avg       1.00      1.00      1.00       600
```

Figure 23. Performance evaluation of Linear SVM model on test data

```
# Predict the labels for Test data for Logistic regression
y_test_pred = logreg_model.predict(X_test_flat)
print("LOGISTIC REGRESSION-Test Data:")
print(classification_report(y_test, y_test_pred, target_names=['Non Spil Oil(Class 0)', 'Oil Spill(Class 1)']))
```

```
LOGISTIC REGRESSION-Test Data:
                      precision    recall  f1-score   support

Non Spil Oil(Class 0)       1.00      1.00      1.00       300
   Oil Spill(Class 1)       1.00      1.00      1.00       300

             accuracy                           1.00       600
            macro avg       1.00      1.00      1.00       600
         weighted avg       1.00      1.00      1.00       600
```

Figure 24. Performance evaluation of Logistic Regression model on test data

```
#Prediction for vgg model
predictions = model.predict(x_test)
```

```
19/19 [==============================] - 60s 3s/step
```

```
pred = np.argmax(predictions, axis=1)
pred = pred.reshape(1, -1)[0]
report = classification_report(y_test, pred, target_names=['Non Oil Spill(Class 0)', 'Oil Spill (Class 1)'])
print("VGG_19-Test data")
print(report)
```

```
VGG_19-Test data
                        precision    recall  f1-score   support

Non Oil Spill(Class 0)       0.99      1.00      1.00       300
   Oil Spill (Class 1)       1.00      0.99      0.99       300

              accuracy                           0.99       600
             macro avg       1.00      0.99      0.99       600
          weighted avg       1.00      0.99      0.99       600
```

Figure 25. Performance evaluation of VGG19-model on test data

4 Results and Discussion

```
df
```

	Accuracy	Model
0	0.99	vgg-19
1	0.92	SVC_rbf
2	1.00	SVC_linear
3	1.00	log_reg
4	0.86	rf
5	0.86	gb
6	0.88	Xgb

Figure 26. Accuracies of all the models

From the results of we can observe that all three linear models have performed very well whereas non-linear models have shown variations in accuracy. From this we can conclude that the dataset is linearly separable.

	Accuracy	Model
0	1.0	SVC_linear
1	1.0	Log_reg
2	1.0	vgg-19

Figure 27. Accuracies of selected models

All three selected models are linear and hence they performed exceptionally well. Further extension of the project may use complex satellite imagery which is not linearly separable may produce slight variation in accuracies.

5 Conclusion and Recommendations

In this project, we have successfully developed seven classification models capable of classifying ocean imagery into spill and no spill categories. The evaluation metrics demonstrated that the model achieved maximum possible performance in terms of Accuracy and F-score. Although the model demonstrated promising results, there is still room for improvement. Future work could explore the following recommendations:

Investigate additional data augmentation techniques to enhance the model's generalization capabilities. Experiment with different data sets that are not highly linear separable etc. Explore the use of transfer learning with other pre-trained models to leverage knowledge from related tasks and potentially improve performance.

By addressing these recommendations, we believe that the selected models for ocean imagery classification could be further refined and contribute to more efficient and effective detection of oil spills.

6 References

[1] Guo, L., Li, X., Li, Z., & Li, X. (2013). Automatic detection of oil spills in satellite images using Bayesian classification and morphological filtering. Marine Pollution Bulletin, 77(1-2), 218-224.

[2] Liu, Y., Huang, X., Xie, Y., & Gao, S. (2014). Oil spill detection in synthetic aperture radar images using texture features and support vector machines. Journal of Applied Remote Sensing, 8(1), 083575.

[3] Prasad, P., Mohan, M., Chakravarthy, V., & Naidu, K. (2017). Oil spill detection using image processing techniques. Journal of Marine Science and Engineering, 5(3), 31.

[4] Suresh, M., Sundaram, S., & Jaya, D. S. (2018). Oil spill detection from satellite images using convolutional neural networks. Marine Pollution Bulletin, 129(1), 262-268.

[5] Alharbi, H., Al-Muhtadi, J., & Al-Muhtadi, Y. (2019). Oil spill detection using deep learning techniques: A comparative study. In Proceedings of the 6th International Conference on Control, Decision and Information Technologies (CoDIT) (pp. 722-727). IEEE.

[6] Gao, F., Liu, Y., Sun, Q., & Gao, X. (2021). Detection of marine oil spills based on a VGG16 model. Marine Pollution Bulletin, 163, 111941.

[7] Jena, S. K., Nayak, J., & Panda, S. (2020). Deep learning approach for oil spill detection using satellite images. Journal of Ocean Engineering and Science, 5(1), 46-54.

[8] Ghavami, M., Samiei-Esfahany, S., & Zare, M. (2020). A deep learning-based oil spill detection approach using UAV images. Marine Pollution Bulletin, 151, 110833.

[9] Abdullah, R., & Marsono, M. N. (2021). An efficient deep learning method for oil spill detection using Sentinel-2 satellite imagery. Remote Sensing, 13(6), 1089.

[10] He, X., Wang, Z., Zhang, Y., Zhou, Y., Wang, X., & Guan, L. (2020). Oil spill detection in satellite images with multiscale features and deep learning. IEEE Transactions on Geoscience and Remote Sensing, 59(11), 9683-9693.