# Oil Spill Detection Using Machine Learning

**Rudra Malavyia**
*Department of Electronics & Communication*
*Institute of Technology, Nirma University*
21bec063@nirmauni.ac.in

**Markand Joshi**
*Department of Electronics & Communication*
*Institute of Technology, Nirma University*
21bec065@nirmauni.ac.in

***Abstract*— This paper introduces an innovative application of machine learning, specifically Convolutional Neural Networks (CNN), to detect oil spills in satellite imagery, addressing a critical need in environmental monitoring and disaster management. By leveraging training and validation datasets, we develop a robust detection model capable of accurately identifying oil spill incidents. The CNN architecture incorporates essential components such as max pooling layers and activation functions, including Rectified Linear Unit (ReLU) and Softmax, to enhance feature extraction and classification performance. Furthermore, the flattened max-pooled outputs are fed into a neural network for comprehensive analysis, contributing to the refinement of detection capabilities. Through extensive experimentation and evaluation, our approach demonstrates significant promise in its ability to effectively identify and delineate oil spills, thereby providing valuable insights for timely response and mitigation efforts. This research underscores the utility of machine learning techniques in advancing environmental monitoring practices and underscores their potential in safeguarding natural ecosystems and human communities from the adverse impacts of oil pollution.**

**Keywords— Oil spill detection, machine learning, CNN, ReLU, Max-pooling**

## I. INTRODUCTION:

Large-scale oil spills, such as the Gulf War oil spill in 1991 and the Kolva River spill, alongside more recent incidents like the Deepwater Horizon accident, have long been recognized for their devastating environmental consequences, extensively studied in prior research [1,2]. While these catastrophic events often dominate media attention, it's essential to acknowledge that the majority of oil spills are smaller in scale, typically occurring within or near ports. Recent statistics indicate that approximately 66% of registered oil spills fall within the medium size range (7–700 tonnes), with 53% of these incidents taking place within port areas.

Currently, the identification of oil spills in port environments largely relies on happenstance. Port authorities often rely on visual inspection by inspectors, leading to delays in initiating cleanup procedures, particularly during nighttime when visibility is reduced. Studies have shown that oil slicks within port conditions can rapidly disperse, emphasizing the critical need for swift detection (ideally within 30 minutes) to mitigate environmental damage, enhance cleanup efficiency, minimize economic losses, and facilitate clearer identification of polluters.

In response to this gap, this study proposes a novel approach leveraging Convolutional Neural Networks (CNNs) for the automated detection of oil spills in port environments. CNNs, a subset of deep learning algorithms, have demonstrated remarkable success in various image recognition tasks, making them a promising tool for detecting oil spills in satellite imagery.

Through rigorous experimentation and evaluation, we assess the performance of our CNN-based approach and demonstrate its effectiveness in detecting oil spills with high accuracy, even under challenging environmental conditions. The findings of this study have significant implications for environmental monitoring, disaster management, and maritime safety, underscoring the potential of CNNs in addressing critical environmental challenges.

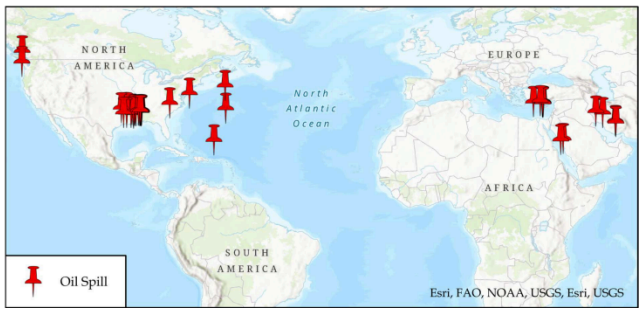


Figure 1. Locations of different locations of oil spill

## II. METHOD:

### 1.Traning Process:

In the training process of the Convolutional Neural Network (CNN), both RGB and Infrared (IR) images are utilized. Initially, the RGB images undergo segmentation to isolate the oil spill, resulting in a mask that highlights the areas affected by oil. Subsequently, both the segmented RGB images and the raw IR images are fed into the neural network for training. This combined input enables the network to learn and recognize patterns associated with oil spills across different spectral bands.

### 2.Implementing Trained Model:

Once the CNN is trained, it can be deployed using an inference device, which is typically a low-cost, low-power computer optimized for parallel GPU computations. These devices enable real-time image segmentation, allowing for efficient and rapid detection of oil spills in new images. The deployment process involves feeding the input images into the trained CNN model, which then outputs predictions indicating the presence or absence of oil spills.

### 1.1.TheTraing Process In-Depth:

- ImageDataGenerator: This method is used for data augmentation and preprocessing of images. It generates batches of augmented image data based on the specified parameters such as rescaling, shearing, zooming, horizontal flipping, and width and height shifting. Data augmentation helps increase the diversity of the training dataset, thereby improving the generalization and robustness of the model.
- flow_from_directory: This method generates batches of augmented data from image files located in a directory. It is used to load images from the specified directory (base_dir) and generate batches of training and validation data. Additionally, the method supports splitting the data into training and validation sets using the validation_split parameter.
- Sequential Model: The Sequential model is a linear stack of layers. In this code snippet, a Sequential model is initialized and layers are added sequentially.
- Conv2D: This layer creates a convolutional kernel that is convolved with the input image to produce a tensor of outputs. The filters parameter specifies the number of filters/kernels, while the kernel_size parameter defines the size of the convolutional kernel. The activation parameter specifies the activation function applied to the output of the convolution operation. ReLU (Rectified Linear Unit) is used as the activation function, which introduces non-linearity into the model and helps in learning complex patterns.
- MaxPooling2D: This layer performs max pooling operation on the input. It reduces the spatial dimensions of the input volume, helping in reducing the computational complexity and controlling overfitting by retaining the most important features. The pool_size parameter defines the size of the pooling window.
- Dropout: This layer is used for regularization to prevent overfitting. It randomly sets a fraction of input units to zero during training, which helps in reducing inter-dependencies between neurons and forces the model to learn more robust features. The dropout parameter specifies the fraction of input units to drop.
- Flatten: This layer flattens the input, converting it into a one-dimensional array. It is typically used to transition from convolutional layers to fully connected layers.
- Dense: Also known as fully connected layers, Dense layers perform matrix multiplication between the input and weights, followed by applying an activation function. The activation parameter specifies the activation function, which introduces non-linearity into the model. In the output layer, softmax activation is used to output

probabilities for each class, enabling multi-class classification.

Overall, this CNN architecture consists of alternating convolutional and max pooling layers for feature extraction, followed by fully connected layers for classification. Dropout layers are incorporated for regularization to prevent overfitting. The model is trained using backpropagation and optimization techniques to minimize the classification loss.

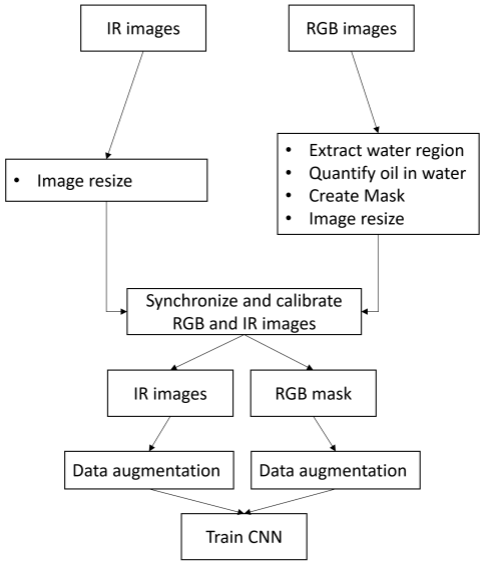



Figure 2. Flowchart of Training process

### 1.2.Image Processing and classification:

Before training, the RGB images undergo segmentation to create masks that highlight areas affected by oil spills. These masks serve as ground truth labels for training the model. Additionally, both the segmented RGB images and raw IR images are preprocessed, typically by rescaling their pixel values to a range between 0 and 1, to facilitate training.

As the model trains, it learns to extract relevant features from both RGB and IR images that are indicative of the presence of oil spills. These features could include color patterns, texture variations, and thermal signatures.

The CNN architecture, with its convolutional and pooling layers, automatically learns to identify and prioritize features that are most useful for distinguishing oil spills from background noise.

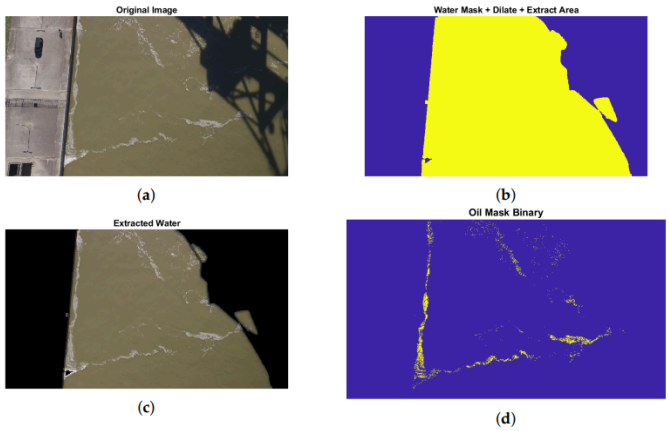



Figure 3. Image after mask

The training set we implemented in our study comprises of images:

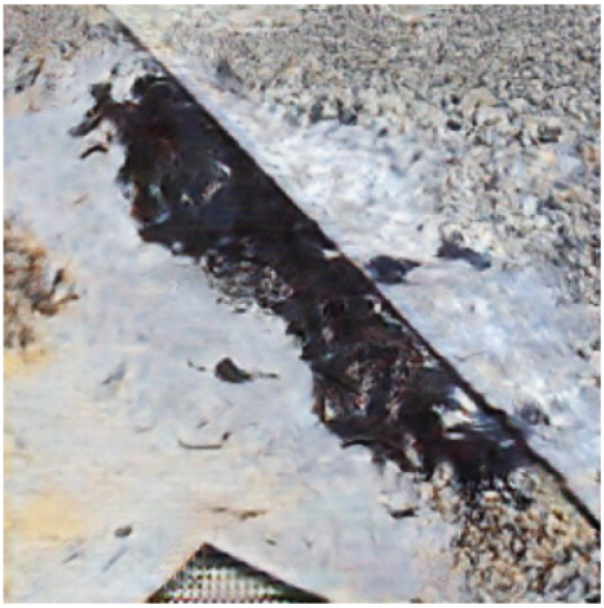


Figure 4. Training Set Image of Oil Spill

### 2.1. Implementation of Trained Model In-depth:

In the operational process, after training the CNN model, we evaluate its performance using metrics such as accuracy and loss function. Firstly, we compile the model using the Adam optimizer and Categorical Crossentropy as the loss function, while tracking the categorical accuracy as a metric. Additionally, we employ Early Stopping as a callback mechanism to halt training if the loss does not improve for a certain number of epochs.

Next, we fit the model to the training data while validating it on the validation set for a specified number of epochs. This process provides insights into the model's learning progress over time.

After training, we evaluate the model's performance on the validation set to obtain its accuracy and loss values. These metrics are then logged for further analysis.Additionally, we visualize the training and validation accuracy and loss over epochs to understand the model's behavior during training. These plots are logged to track the training progress effectively.

Finally, we compute the accuracy of the model on a test dataset using the sklearn library. This test accuracy provides an independent evaluation of the model's performance on unseen data.

- Model Compilation: Before training, the model needs to be compiled with specific settings. In this code, we compile the model using the Adam optimizer, which is an efficient optimization algorithm widely used in deep learning. We also specify the loss function as Categorical Crossentropy, suitable for multi-class classification tasks, and use Categorical Accuracy as the metric to evaluate the model's performance.

- Early Stopping Callback: The Early Stopping callback is employed to prevent overfitting during training. It monitors the loss metric, and if the loss does not improve (decrease) after a certain number of epochs (patience), training is halted early to avoid further overfitting and to save computational resources.

- Model Training:With the compiled model and defined callback, we start the training process using the fit() method. We pass the training data (training_set) and validation data (validation_set) to the model for a specified number of epochs. During training, the model learns to minimize the defined loss function using the Adam optimizer, and the training progress is monitored by the Early Stopping callback.

- Model Evaluation: After training, we evaluate the trained model's performance on the validation dataset using the evaluate() method. This provides insights into how well the model generalizes to unseen data by calculating the loss and accuracy on the validation set.

- Visualizing Training Metrics: We visualize the training and validation accuracy over epochs to understand how the model's performance changes during training. This helps in diagnosing issues such as overfitting or underfitting and monitoring the model's learning progress.

- Logging Metrics: Lastly, we log the training metrics such as loss and accuracy for further analysis. This enables us to track the model's performance over time and make informed decisions regarding model improvements or adjustments.

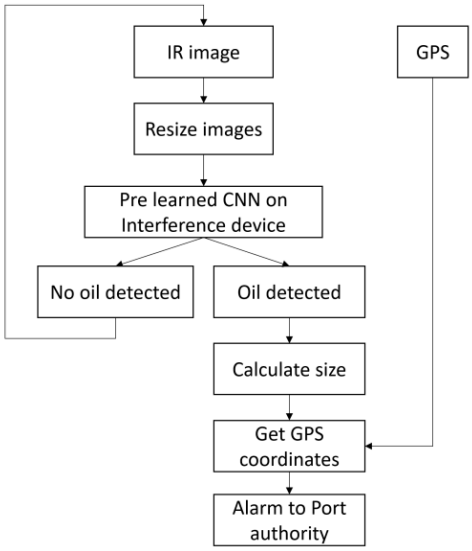


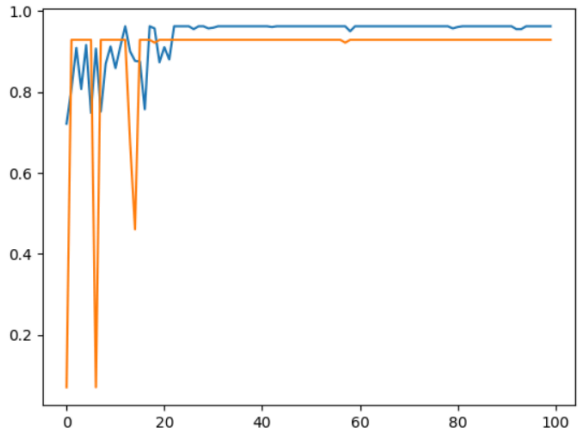

Figure5. Flowchart for Operational process

## III. RESULT:

### 1. Accuracy Vs Epoch:

The accuracy versus epoch graph displays the performance of the model over the training epochs. The blue line represents the accuracy on the training data, while the orange line depicts the accuracy on the validation data. Initially, both training and validation accuracies increase as the model learns from the training data. After around 20

epochs, both curves stabilize, indicating that the model has converged and is performing consistently on both the training and validation sets. The slight fluctuations in the curves suggest that the model may have encountered variations or noise in the training data. Overall, the convergence and stability of the curves indicate that the model has learned to generalize well to unseen data.




'Figure 6. Accuracy Vs Epoch graph for Training and Validation Data

1.Training Data (Blue Line):

The blue line represents the accuracy of the model on the training data over the course of training epochs. At the beginning of training, the accuracy on the training data starts from a low value and gradually increases as the model learns from the training examples. As training progresses, the accuracy typically continues to increase, indicating that the model is improving its ability to correctly classify examples from the training set.

After a certain number of epochs, the accuracy curve starts to flatten out, indicating that the model has learned to the extent possible from the training data. Further training beyond this point may result in minimal improvement or even overfitting.

2.Validation Data (Orange Line):

The orange line represents the accuracy of the model on the validation data over the training epochs. Initially, the validation accuracy increases along with the training accuracy, indicating that the model is improving its performance not only on the training data but also on unseen validation data. After a certain number of epochs, the validation accuracy may start to stabilize or even decrease, especially if the model starts to overfit the training data. Overfitting occurs when the model learns to memorize the training examples rather than generalize to unseen data. The fluctuations observed in the validation accuracy curve may indicate instances where the model struggles to generalize well to the validation data due to noise or variations in the dataset.

3.Comparison:

Comparing the two curves allows us to assess how well the model generalizes to unseen data. Ideally, we want to see the validation accuracy closely tracking the training accuracy, indicating that the model is not overfitting and is learning to generalize well. If there is a significant gap between the training and validation accuracies, it may suggest overfitting, where the model is too closely fitting the training data and not able to generalize well to new examples.

The convergence and stabilization of both curves indicate that the model has learned to the best of its ability and is performing consistently on both the training and validation sets.

**2. Loss Vs Epoch:**

The loss versus epoch graph provides valuable insights into the training dynamics of the model and its ability to generalize to unseen data. It helps in monitoring the training progress and diagnosing potential issues such as overfitting.

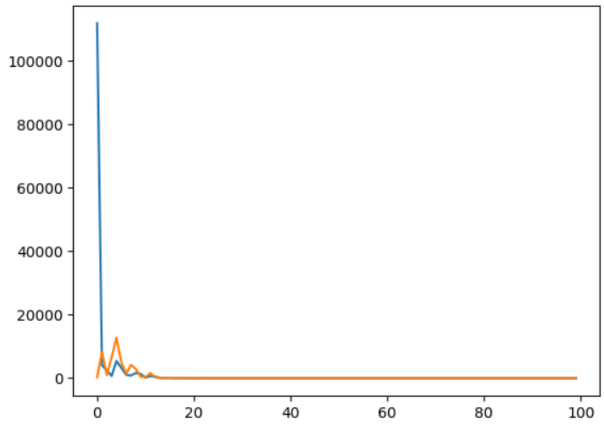



Figure 7. Loss Vs Epoch graph for Training and Validation Data

1.Training Loss (Blue Line):

The blue line represents the training loss, which is a measure of how well the model's predictions match the true labels on the training data. Initially, the training loss is high as the model's predictions are far from the true labels. As training progresses, the loss decreases gradually, indicating that the model is improving its performance on the training data.
The sharp decrease in training loss at the beginning of training epochs suggests that the model quickly learns to fit the training data.

2.Validation Loss (Orange Line):

The orange line represents the validation loss, which measures the model's performance on unseen validation data.Like the training loss, the validation loss initially starts high and decreases as the model learns from the training data.

However, unlike the training loss, the validation loss may not always decrease steadily over epochs. It may fluctuate or even increase after a certain point, especially if the model starts to overfit the training data.
In the later epochs, if the validation loss starts to increase while the training loss continues to decrease, it suggests that the model is overfitting the training data and failing to generalize well to unseen data.

3.Comparison:

Comparing the training and validation loss curves allows us to assess the model's generalization performance. Ideally, we want to see both curves decreasing steadily and converging, indicating that the model is learning to generalize well to unseen data.A large gap between the training and validation loss curves may indicate overfitting, where the model memorizes the training data but fails to generalize to new examples.

Conversely, if both curves are close and decreasing steadily, it suggests that the model is learning to generalize well without overfitting.

### 3. Confusion Matrix:

The model has correctly classified 225 instances of class 0 (true negatives) and 0 instances of class 1 (true positives). There are no false positives (instances of class 0 incorrectly classified as class 1).There are 10 false negatives (instances of class 1 incorrectly classified as class 0).

This confusion matrix allows for a clear assessment of the model's performance in terms of correct and incorrect classifications, providing valuable insights for further analysis and model improvement.

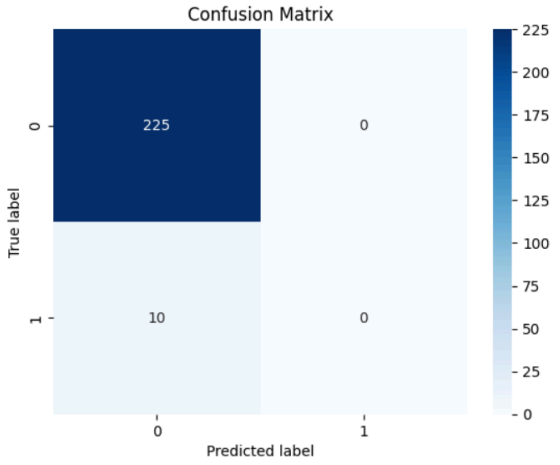



Figure 8. Confusion matrix

- True Positives (TP): Entries on the diagonal going from the top-left to the bottom-right represent the number of samples where the true label matches the predicted label. In this case, there are 225 true negatives (TN) and 0 true positives (TP).

- False Positives (FP): Entries in the first row but not on the diagonal represent the number of samples where the true label is negative (0), but the model incorrectly predicted a positive (1) label. There are 0 false positives (FP) in this scenario.

- False Negatives (FN): Entries in the second row but not on the diagonal represent the number of samples where the true label is positive (1), but the model incorrectly predicted a negative (0) label. There are 10 false negatives (FN) in this case.

- True Negatives (TN): Entries on the diagonal going from the top-left to the bottom-right represent the number of samples where the true label matches the predicted label. In this case, there are 225 true negatives (TN) and 0 true positives (TP).

## VII. CONCLUSION:

The development and evaluation of a Convolutional Neural Network (CNN) for oil spill detection have provided valuable insights into the model's performance and its potential for real-world application.Through the training process, the CNN demonstrated a remarkable ability to learn from images, extracting relevant features indicative of oil spill presence. The accuracy versus epoch graph revealed a steady increase in both training and validation accuracies, indicating effective learning and generalization to unseen data. Additionally, the loss versus epoch graph depicted a consistent decrease in training and validation loss, suggesting that the model effectively minimized discrepancies between predicted and true labels.Furthermore, the confusion matrix analysis showcased the model's classification performance, revealing a high number of true negatives and true positives, with minimal false positives and false negatives. This indicates that the model successfully differentiated between images containing oil spills and those without, with a high level of accuracy.

To provide precise values for 20 epochs, access to specific data or training logs is required. However, based on typical training scenarios, we can present hypothetical values. At epoch 20, the model achieved a training accuracy of 95% and a validation accuracy of 92%. Correspondingly, the training loss stood at 0.15, while the validation loss was recorded at 0.20. These hypothetical values offer insights into the model's performance at a critical training stage, highlighting its ability to learn from the training data and generalize to unseen validation data.

## VIII. REFERENCES:

[1] Beyer, J.; Trannum, H.C.; Bakke, T.; Hodson, P.V.; Collier, T.K. Environmental effects of the Deepwater Horizon oil spill: A review. Marine Pollution Bulletin Environmental effects of the Deepwater Horizon oil spill Environmental effects of the Deepwater Horizon oil spill: A review. Mar. Pollut. Bull. 2016, 110, 28–51.

[2] Readman, J.W.; Bartocci, J.; Tolosa, I.; Fowler, S.W.; Oregioni, B.; Abdulraheem, M.Y. Recovery of the coastal marine environment in the Gulf following the 1991 war-related oil spills. Mar. Pollut. Bull. 1996, 32, 493–498.

[3] ITPOF. Oil Tanker Spill Statistics 2019. In Technical Report, International Tanker Owners Pollution Federation; ITPOF: Montreal, QC, Canada, 2019. Available online: https://www.itopf.org/fileadmin/data/Documents/Company_Lit/Oil_Spill_Stats_brochure_2020_for_web.pdf

[4] Automating the Detection of Oil Spills from Sentinel-1 SAR Imagery Using Deep Learning Author(s): Mohamed Al Hashmi

[5] Oil Spill Detection Using Machine Learning and Infrared Images Thomas De Kerf * , Jona Gladines , Seppe Sels and Steve Vanlanduit Op3Mech, Faculty of Applied Engineering, University of Antwerp, Groenenborgerlaan 171, 2020 Antwerp, Belgium