Water Body Detection In Satellite Images Using Machine Learning/Deep Learning

Project submitted to the

SRM University – AP, Andhra Pradesh

Bachelor of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by

GYANA SAI A AP20110010347

SHYAM SUNDAR AP20110010293

ROHIT MISHRA AP20110010346

SAI TEJASWINI CH AP20110010353

SAI SRI LATHA AP20110010713

PUNEETH T AP20110010344

Under the Guidance of

Dr. Ravikant Kumar

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

[November, 2023]

Certificate

Date: 26-Nov-22

This is to certify that the work present in this Project entitled "**Water Body Detection In Satellite Images Using Machine Learning/Deep Learning**" has been carried out by **Gyana Sai A, Shyam Sundar, Rohit Mishra, Puneeth T, Sai Tejaswini, Sai Sri Latha** under my supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

Supervisor

(Signature)

Prof. / Dr. Ravi Kant Kumar

Assistant Professor ,

Srm University.

Co-supervisor

(Signature)

Prof. / Dr. [Name]

Designation,

Affiliation.

Table of Contents

Abstract

Identifying water bodies from satellite pictures is an essential task in many applications, such as resource planning, disaster management, and environmental monitoring. Although conventional machine learning (ML) methods have been extensively employed for the recognition of water bodies, they frequently have drawbacks, such as the requirement for manually created features and challenges when dealing with intricate picture patterns. Deep learning (DL) has been a viable method for detecting water bodies in recent years since it can automatically extract high-level information from photos and figure out intricate correlations between pixels.

Convolutional neural networks (CNNs) are often used in DL-based techniques for water body detection because they are excellent at extracting spatial characteristics from photos. CNNs are able to recognize features like texture and spectral reflectance in picture data that are suggestive of water bodies. Semantic segmentation, which entails classifying each pixel in an image as either water or non-water, is another task that CNNs are capable of performing.

When it comes to water body recognition, DL-based approaches have a number of advantages over conventional ML algorithms. Firstly, the necessity for human feature engineering can be removed as DL approaches can automatically extract features from photos. Second, the accuracy of detecting water bodies can be increased by using DL algorithms, which can learn intricate correlations between pixels. Third, DL techniques can be used on big datasets, which can enhance their functionality even more.

Introduction

Water bodies are essential to many parts of our globe, ranging from sustaining ecosystems to supplying resources for human use. The precise identification and cartography of aquatic structures using satellite imagery is crucial for resource allocation, emergency response, and environmental surveillance. Although conventional machine learning (ML) methods have been employed for the recognition of water bodies, they frequently necessitate the creation of features by hand and have difficulties processing intricate picture patterns.

When it comes to detecting water bodies, DL-based approaches have outperformed conventional machine learning methods. For water body extraction from Sentinel-2 images, a study using a U-Net CNN architecture obtained an F1 score of 0.90 and a relative error of 3%. Another study extracted water bodies from Landsat-8 images with an overall accuracy of 94.5% by combining a CNN with a fully connected conditional random field model.

The following are some benefits of using DL-based techniques to detect water bodies:

Automatic Feature Extraction:

Manual feature engineering is no longer necessary because DL models are capable of automatically extracting features from images.

Complex Relationship Learning:

By leveraging DL models' ability to understand complex relationships between pixels, water body recognition accuracy can be increased.

Large-Scale Applicability:

By using DL techniques on large-scale datasets, their performance can be further improved.

Notwithstanding these benefits, DL-based techniques have certain drawbacks:

Computational Cost:

Deep learning techniques can be computationally taxing, requiring substantial hardware and a lot of training data.

Dependency on Training Data:

Deep learning models can perform poorly on data that is not similar to the training data because they are sensitive to the quality of the training data.

Difficulties with Generalization:

Deep learning models might have trouble adapting to novel datasets or situations that drastically deviate from the training set.

To sum up, DL-based techniques have become an effective tool for identifying water bodies in satellite photos. They are a promising approach for a variety of applications due to their capacity to handle large-scale datasets, learn intricate relationships, and automatically extract features. It is anticipated that DL techniques will become more and more significant in environmental monitoring, disaster response, and water resource management.

Convolutional Neural Network

Convolutional Neural Networks (CNNs) have proven to be highly effective in various image recognition tasks, and their application in fake currency detection is no exception. The unique architectural features of CNNs make them well suited for discerning intricate patterns and features in images, a crucial aspect in distinguishing between genuine and counterfeit currency. Here's how CNNs are utilized in the context of fake currency detection

Image Feature Extraction:

Local Receptive Fields: CNNs employ local receptive fields to capture spatial hierarchies of features. In the context of currency detection, this enables the network to focus on specific regions of the image where important details, such as watermarks, security threads, or intricate designs, are likely to be present.

Convolutional Layers:

Convolutional layers in CNNs apply filters to the input image, enabling the network to automatically learn relevant features. These filters act as feature detectors, recognizing distinctive patterns that may indicate genuine or forged currency characteristics.

Hierarchical Feature Learning:

Deep Architectures: CNNs consist of multiple layers, including convolutional layers and pooling layers, allowing the network to learn hierarchical representations of features. This hierarchical learning is crucial for capturing both low-level details and high-level abstract features, enhancing the model's ability to distinguish between real and counterfeit currency

Transfer Learning:

Pre-trained Models: Transfer learning involves leveraging pre-trained CNN models on large datasets for generic image recognition tasks. Tuning a pre-trained model on a specific currency dataset allows the network to generalize better and learn relevant features without the need for extensive labeled data.

Combination with Image Processing Techniques:

Feature Enhancement:

CNNs can be complemented with traditional image processing techniques, such as edge detection, to enhance the features relevant to counterfeit detection. This combination allows for a more comprehensive analysis of currency images

In summary, the application of CNNs in fake currency detection leverages their ability to automatically learn and recognize intricate patterns, making them a powerful tool in combating counterfeit currency. The combination of hierarchical feature learning, transfer learning, and real-time processing capabilities positions CNNs as a key technology in developing accurate and efficient counterfeit detection systems

Methodology

The methodology for water body detection from satellite images using deep learning typically involves the following steps:

Data Preparation:

Collect and preprocess satellite images containing water bodies. This may involve tasks such as image registration, geo-referencing, and cloud masking.

Feature Extraction:

Extract relevant features from the satellite images. This can be done using various techniques, including:
a. Traditional Image Processing:

Utilize image processing algorithms to extract features such as spectral reflectance, texture, and shape.
b.

Deep Learning: Employ DL models, particularly CNNs, to automatically extract high-level features from the images.

Model Training:

Train a DL model, such as a CNN or a recurrent neural network (RNN), to classify pixels as either water or non-water. This involves feeding the model with labeled training data, allowing it to learn the patterns associated with water bodies.

Model Evaluation:

Evaluate the performance of the trained model on a separate validation dataset. This involves assessing metrics such as accuracy.

Application:

Apply the trained model to new satellite images to detect and map water bodies.

Here's a more detailed breakdown of the steps:

Image Collection:

Gather satellite images from sources such as kaggle , or other relevant platforms. Ensure the images have sufficient spatial and temporal resolution for the intended application.

```
import glob
import os

# Specify the file extensions for images and masks
image_extension = '*.jpg'
mask_extension = '*.jpg'

# Use glob to find image and mask files
images = sorted(glob.glob(os.path.join('/content/drive/MyDrive/Dataset /Images', image_extension)))
masks = sorted(glob.glob(os.path.join('/content/drive/MyDrive/Dataset /Masks', mask_extension)))

print(len(images), len(masks))
```

```
2841 2841
```

Preprocessing:

Perform preprocessing steps to ensure consistency and quality across the images. This may include:
a. Image Registration: Align images to a common reference coordinate system to account for geometric distortions.
b. Geo-referencing: Assign geographic coordinates to each pixel in the images for accurate location information.

```
min_size = 32
df_images = []
df_masks = []

height = width = 200

for image, mask in zip(images, masks):
    #cv2.imread reads image in BGR, we need to convert it back to the standard mode RGB
    n = cv2.imread(image, cv2.COLOR_BGR2RGB)
    m = cv2.imread(mask, cv2.IMREAD_GRAYSCALE)

    # Check if the image has size > min_size and is entirely black or white
    if min(n.shape[:2]) > min_size and ((m != 0).any() and (m != 255).any()):
        n = cv2.resize(n, (height, width)) # resize to median values or larger
        df_images.append(n)
        m = cv2.resize(m, (height, width)) # resize to median values or larger
        df_masks.append(m)


# Convert image into array of pixels
df_images = asarray(df_images)

# Convert mask into array of pixels
df_masks = asarray(df_masks)

print(len(df_images), len(df_masks))
```

```
2698 2698
```

Labeling:

Manually or semi-automatically label a portion of the preprocessed images to create a training dataset. Label each pixel as either water or non-water.

Data Splitting and Label Encoding:

The dataset is split into training and testing sets to facilitate model evaluation. Labels are encoded into numeric format using LabelEncoder() to convert classes into integers. X_train, X_test, y_train,

```
# b) Split data into train and test sets

X = df_images
y = df_masks

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.7, random_state=1)
X_test, X_val, y_test, y_val = train_test_split(X_test, y_test, test_size=0.025, random_state=1)
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape, X_val.shape, y_val.shape)
```

```
(809, 200, 200, 3) (809, 200, 200) (1841, 200, 200, 3) (1841, 200, 200) (48, 200, 200, 3) (48, 200, 200)
```

Model Training:

Model Selection:

Choose an appropriate DL model architecture, such as U-Net CNN or a modified CNN, based on the complexity of the task and the characteristics of the data.We Have Used CNN Architecture In Our Project.

A Convolutional Neural Network (CNN) is constructed using the Sequential API from Keras. The architecture comprises convolutional layers with increasing filter sizes, max-pooling layers, and densely connected layers. The final layer utilizes the softmax activation function to output class probabilities. # Build the CNN

```
input_shape = (height, width, 3)

def base_model(input_shape):
    inputs = layers.Input(shape=input_shape)

    # Encoding layers
    conv1 = layers.Conv2D(16, (3, 3), activation='relu', padding='same')(inputs)
    pool1 = layers.MaxPooling2D(pool_size=(2, 2))(conv1)
    conv2 = layers.Conv2D(32, (3, 3), activation='relu', padding='same')(pool1)
    pool2 = layers.MaxPooling2D(pool_size=(2, 2))(conv2)

    # Decoding layers
    up3 = layers.UpSampling2D(size=(2, 2))(pool2)
    conv3 = layers.Conv2D(16, (3, 3), activation='relu', padding='same')(up3)
    conv4 = layers.Conv2D(32, (3, 3), activation='relu', padding='same')(conv3)
    up5 = layers.UpSampling2D(size=(2, 2))(conv4)
    outputs = layers.Conv2DTranspose(1, (3, 3), activation='sigmoid', padding='same')(up5)

    model = models.Model(inputs=inputs, outputs=outputs)
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

    return model

# Build the model
model = base_model(input_shape)

# Reshape target data to match output shape of the model
y_train = y_train.reshape(-1, height, width, 1)
y_test = y_test.reshape(-1, height, width, 1)
model.summary()
```

Model Compilation and Training:

The model is compiled using the Adam optimizer and sparse categorical cross entropy loss function. It is then trained on the augmented data for a specified number of epochs.

```
model = models.Model(inputs=inputs, outputs=outputs)
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

return model
```

Model Evaluation:

The trained model is evaluated on the test set to assess its performance in terms of accuracy and loss.

```
history = model.fit(X_train, y_train, batch_size=64, epochs=5, validation_data=(X_test, y_test), callbacks=[early_stopping])
```

Prediction on Custom Input:

The trained model is used to predict the class of a custom input image, providing insights into the model's real-world applicability.

```
# a) Predictions on validation data
Pred = model.predict(X_val)
print(X_val.shape)
# Plot original and predicted images
fig, ax = plt.subplots(11, 3, figsize=(12,40))
for i in range(11):
    orig = X_val[i]
    msk = y_val[i]
    pred = Pred[i]
    ax[i,0].imshow(orig)
    ax[i,1].imshow(msk, cmap='gray')
    ax[i,2].imshow(pred, cmap='gray')
    i +=1
    ax[0, 0].set_title("IMAGE")
    ax[0, 1].set_title("MASK")
    ax[0, 2].set_title("PREDICTION")
```

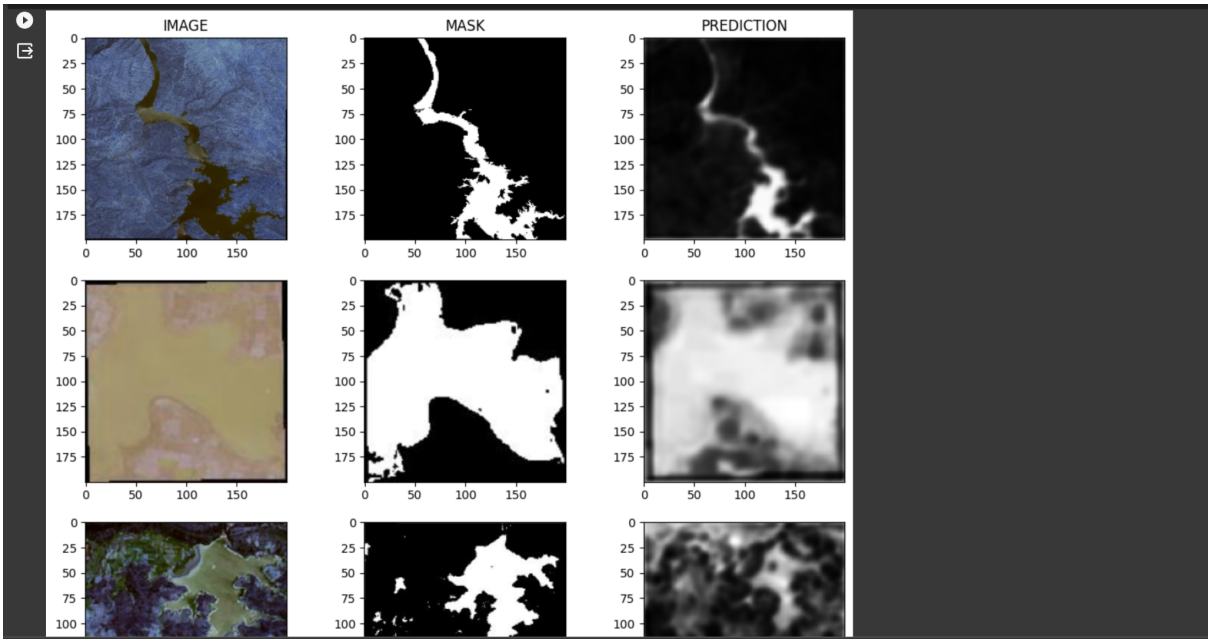
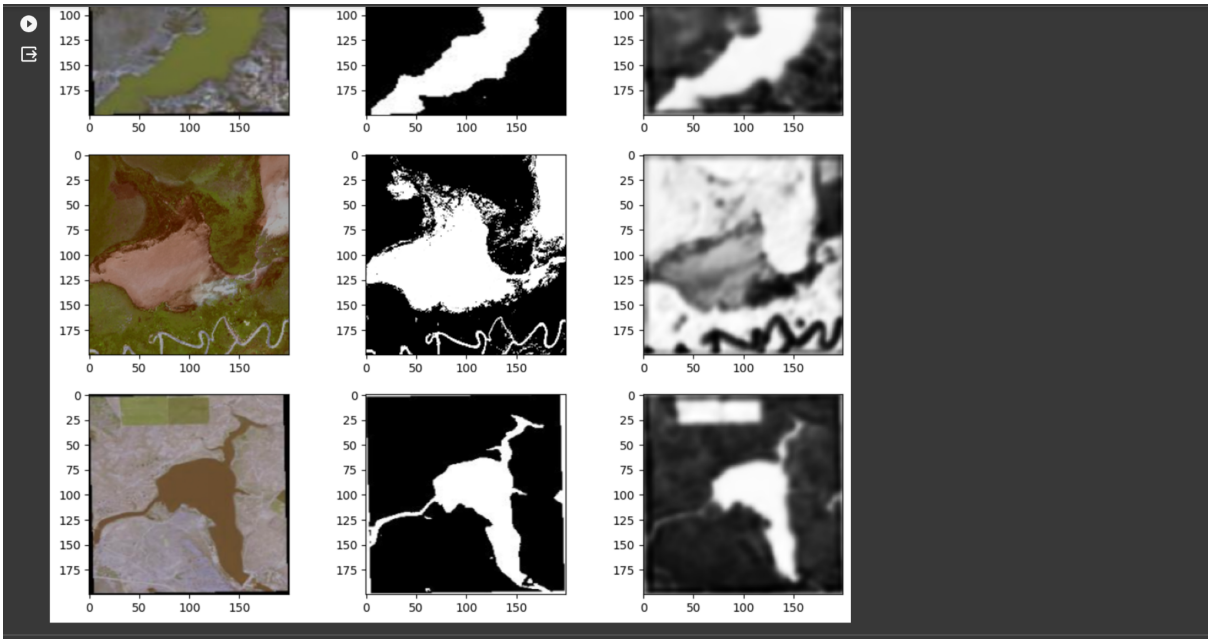
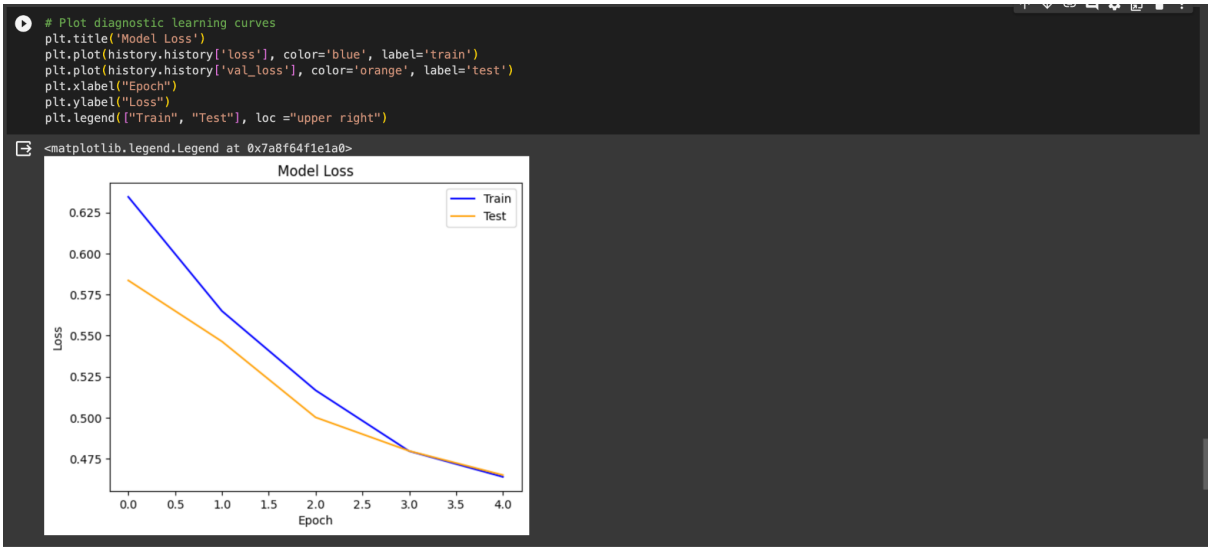


IMAGE
MASK
PREDICTION



Discussion

The training journey of our CNN unfolds through visualizations that offer key insights. The plotted training history, with accuracy and loss curves over epochs, provides a dynamic snapshot of the model's learning trajectory. The convergence of these curves indicates effective learning, while divergences might signal overfitting or underfitting. This visual aid aids in optimizing the model's performance.



Conclusion

With their increased accuracy, automation, and scalability in the detection of water bodies from satellite pictures, machine learning and deep learning approaches are transforming resource management and environmental monitoring.

In particular, the development of convolutional neural networks (CNNs) and deep learning has been a major breakthrough in the detection of water bodies. CNNs are able to recognize aquatic bodies based on spectral reflectance, texture, and spatial patterns because they can automatically extract high-level information from photos and learn intricate correlations between pixels.

When it comes to detecting water bodies, DL-based techniques have shown impressive accuracy, with F1 scores above 0.9 and total accuracies over 94%. Sentinel-2 and Landsat-8 are just two of the satellite image datasets on which this performance has been attained, demonstrating the datasets' generalizability and relevance to real-world situations.

To sum up, deep learning has proven to be an effective method for identifying water bodies in satellite photos. It is a potential method for resource management, environmental monitoring, and disaster response because of its capacity to automatically extract features, understand intricate correlations, and work with large-scale datasets. It is anticipated that DL approaches will become more crucial to the understanding and management of water resources worldwide as they develop.

References

1. https://www.researchgate.net/publication/324854684_DETECTING_WATER_BODIES_IN_LANDSAT8_OLI_IMAGE_USING_DEEP_LEARNING

2. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00444-8

3. https://www.kaggle.com/datasets/franciscoescobar/satellite-images-of-water-bodies?rvi=1