iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** I **Month of publication:** January 2025

DOI: https://doi.org/10.22214/ijraset.2025.66121

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue I Jan 2025- Available at www.ijraset.com

Automated Salt Segmentation in Seismic Images Using U-Net Architecture for Improved Geological Analysis

Kushagra Shukla[1], Harsh[2], Aryan Gautam[3], Raghav Soni[4], Prof. Vidya R[5]
Dept. of CS & E , Bangalore Institute of Technology, Bangalore, India

Abstract: ***Salt segmentation is a critical advancement in geospatial and geological analysis, enabling precise identification of salt deposits in seismic imagery. This project utilizes the U-Net architecture—a deep learning model known for its robust performance in biomedical image segmentation—to automate and enhance the accuracy of salt detection. Leveraging an encoder-decoder structure with skip connections, the system achieves high-resolution segmentation of complex geological boundaries. Data augmentation and efficient preprocessing ensure model generalization across diverse geological datasets. This solution offers significant improvements in accuracy, efficiency, and scalability, reducing reliance on manual interpretation. Future directions include exploring hybrid architectures and real-time processing for broader applications in energy exploration and environmental monitoring.***

Keywords: U-Net, seismic images, salt segmentation, deep learning, geospatial analysis

I. INTRODUCTION

Traditional methods for salt segmentation in seismic imagery often face challenges related to inefficiency, inconsistency, and susceptibility to errors. These methods hinder resource exploration and geological analysis, resulting in increased costs and operational delays. By leveraging advancements in deep learning, this project aims to address these challenges through an automated and robust salt segmentation system.

The U-Net architecture—originally designed for biomedical image segmentation—is adapted here for seismic data analysis. Its encoder-decoder structure and skip connections ensure precise feature extraction and detailed localization of salt boundaries. Complementary techniques, such as data augmentation and advanced loss functions, enhance the model's ability to generalize across diverse geological datasets.

The objectives of this project include achieving accurate salt deposit segmentation, automating geological workflows, and integrating seamlessly with existing geospatial analysis tools. This paper highlights the challenges of traditional approaches and introduces a scalable, efficient solution using cutting-edge machine learning methodologies.

II. RELATED WORK

A. Litearture Review

The transition from traditional to automated deep learning-based segmentation systems has been a focal point in recent academic and industry research. Various studies emphasize the potential of deep learning models, particularly U-Net, in addressing the limitations of classical salt segmentation techniques. For instance, one study highlights how U-Net's encoder-decoder architecture, combined with skip connections, significantly enhances segmentation accuracy in seismic data by preserving spatial information across layers [1]. Research also demonstrates that modifications to U-Net, such as adding residual connections or boundary decoders, further optimize performance for complex geological datasets [2,3].

Data augmentation has been a key area of focus in improving model generalization. Techniques such as rotation, flipping, and scaling have been shown to increase dataset diversity, as detailed in recent studies, enabling models to perform reliably across varying geological conditions [4]. However, challenges such as overfitting and sensitivity to noisy data remain areas of active investigation.

The application of advanced loss functions, including Dice loss and focal loss, has also been extensively explored. These functions address the class imbalance commonly encountered in salt segmentation tasks, ensuring that smaller salt deposits are accurately identified [5]. Additionally, studies emphasize the importance of integrating pre-trained models and transfer learning to accelerate convergence and improve segmentation results [6].

Lastly, research on hybrid models combining convolutional neural networks (CNNs) with transformers highlights their potential in handling large-scale seismic datasets. These models leverage the strengths of both architectures, offering scalability and enhanced contextual understanding for complex segmentation tasks [9].
This project draws inspiration from these advancements to create a robust and efficient salt segmentation pipeline

B. Existing Solutions and Limitations

Numerous automated salt segmentation solutions have been introduced to address the challenges inherent in traditional methodologies.
Notable among these are models based on the U-Net architecture, which have shown significant promise in segmenting salt deposits from seismic images. Several studies have implemented U-Net variants with advanced features, such as residual blocks and attention mechanisms, to improve performance metrics like accuracy and precision [3,5]. Despite these advancements, reliance on large, annotated datasets remains a common limitation.
Other classical machine learning approaches, such as support vector machines (SVMs) and random forests, have been utilized for segmentation tasks. These models generally require extensive feature engineering and often struggle with complex geological boundaries. Moreover, traditional algorithms are less adaptable to high-resolution seismic imagery and require considerable computational resources for comparable accuracy.
Deep learning models, while superior in many respects, face challenges related to generalization. For instance, models trained on datasets from one geological region may not perform well in another due to variations in seismic data characteristics. This issue often necessitates retraining or fine-tuning models, which can be resource-intensive. Furthermore, high-resolution image segmentation demands substantial computational power and memory, posing scalability challenges for real-world applications.
Scalability challenges for real-world applications.
This project specifically addresses these limitations by optimizing U-Net architecture and leveraging data augmentation techniques to enhance model generalization. The integration of advanced loss functions and attention mechanisms further improves the model's ability to delineate complex geological boundaries accurately. These improvements pave the way for more scalable and robust segmentation solutions tailored to diverse geological scenarios.

III. SYSTEM DESIGN

A. Sequence Diagram

The sequence for Uploading and Processing Seismic Data is as follows:

1) The user uploads a seismic image or dataset in the application.
2) The application sends the uploaded data to the backend server.
3) The backend server preprocesses the data (resizing, normalization, and augmentation).
4) The backend server sends the preprocessed data to the U-Net segmentation model.
5) The U-Net model processes the data and generates a segmentation mask for salt deposits.
6) The backend retrieves the segmentation mask and performs post-processing (e.g., thresholding and overlaying the mask).
7) The backend sends the processed results back to the application.
8) The application displays the segmentation mask overlayed on the original image to the user.

Listing and Managing Processed Results:

1) The user requests to list all previously processed datasets in the application.
2) The application sends this request to the backend server.
3) The backend queries the database for processed datasets and retrieves the relevant information.
4) The backend server sends the retrieved data to the application.
5) The application displays the datasets in a structured format (e.g., table or gallery).
6) The user selects a specific dataset to view detailed segmentation results.
7) The application sends a detailed request to the backend for the selected dataset.
8) The backend retrieves the segmentation results and metrics and sends them back to the application.
9) The application displays the results and metrics to the user.

Training the Model and Viewing Metrics:

1) The user initiates model training using a new dataset in the application.
2) The application sends the dataset to the backend server.
3) The backend preprocesses the dataset and sends it to the U-Net model training module.
4) The training module trains the model, logs performance metrics, and saves the updated model.
5) The backend server retrieves the training logs and metrics and sends them to the application.
6) The application displays the training metrics (e.g., accuracy, IoU, loss values) to the user.

B. Activity Diagram

This activity diagram illustrates the process of uploading, processing, and managing seismic data for salt segmentation:
Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, sc, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

1) The user starts by logging into the application.
2) The user uploads a seismic image or dataset for segmentation.
3) The backend preprocesses the uploaded data and processes it using the U-Net model.
4) The system generates and displays segmentation results.
5) The user can:
    a. View detailed results and metrics.
    b. List previously processed datasets and view historical data.
    c. Train the model with new datasets.
6) The user can log out to end the session.

C. User Interface Diagram

1) Proposed UI/UX for Uploading and Processing Data:
    a. Upload Screen: Allows users to upload seismic images or datasets with file validation.
    b. Processing Screen: Displays progress during preprocessing and segmentation.
    c. Result Display Screen: Shows segmentation results overlayed on the original image, with options for zooming and downloading.
2) Proposed UI/UX for Managing Processed Results:
    a. Result List Screen: Displays a list of previously processed datasets in a table or gallery view.
    b. Detail View Screen: Shows detailed segmentation results, metrics, and visualizations.

IV. IMPLEMENTATION

This section proposes the design, technologies, and architecture for implementing the salt segmentation system using U-Net architecture. The following subsections detail the system's modules, data structures, algorithms, and implementation plan.

A. Feature Technology

1) PyTorch Framework: Used to develop and train the U-Net model for high-accuracy segmentation of seismic images.
2) Flask Web Framework: Employed to build a user-friendly web interface for image upload, segmentation visualization, and result management.
3) MongoDB Atlas: Used as a cloud-based NoSQL database for storing segmentation results and metadata.
4) Matplotlib/Plotly: Utilized for generating detailed visualizations of segmentation outputs and training metrics.
5) For papers with more than six authors: Add author names horizontally, moving to a third row if needed for more than 8 authors.
6) For papers with less than six authors: To change the default, adjust the template as follows.
a) *Selection:* Highlight all author and affiliation lines.
b) *Change number of columns:* Select the Columns icon from the MS Word Standard toolbar and then select the correct number of columns from the selection palette.
c) *Deletion:* Delete the author and affiliation lines for the extra authors.

B. Major Modules

The proposed salt segmentation system consists of eight interconnected modules, each designed to facilitate efficient seismic image analysis and salt segmentation. These modules are described below:

1) Image Upload and Preprocessing Module Enables users to upload seismic images for segmentation. This module preprocesses images by resizing, normalization, and data augmentation to prepare them for U-Net model inference.

2) Segmentation Execution Module Processes seismic images through the U-Net architecture to segment salt deposits. This module ensures accurate identification of salt regions and generates segmentation masks.

3) Result Visualization Module Provides users with side-by-side comparisons of the original image, ground truth (if available), and predicted segmentation masks. This module also displays segmentation metrics such as Intersection over Union (IoU) and Dice Coefficient.

4) Historical Data Management Module Enables users to access previously processed images and segmentation results. This module supports querying results by date, accuracy, or dataset name, fostering easier data exploration and analysis.

5) User Authentication and Authorization Module Manages secure user authentication and role-based access control. This module ensures that only authorized users can access segmentation results and perform operations such as uploads or deletions.

6) Model Training and Evaluation Module Allows developers to train the U-Net model on new datasets and evaluate its performance using metrics such as IoU and loss values. This module includes functionalities for data augmentation and model optimization.

7) Storage and Retrieval Module Uses MongoDB for securely storing segmentation results, associated metadata (e.g., timestamps, user IDs), and training logs. This module ensures efficient data retrieval for subsequent operations.

8) Reporting and Analytics Module Generates detailed reports for model performance, segmentation accuracy, and salt deposit coverage. This module visualizes metrics using libraries like Matplotlib and Plotly to aid in decision-making and model refinement.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced. Styles named “Heading 1”, “Heading 2”, “Heading 3”, and “Heading 4” are prescribed.

C. Data Structures

MongoDB Data Schema Stores information about uploaded images, segmentation masks, and associated metadata.

```
{
        "imageID": "string",
        "userID": "string",
        "segmentationMask": "binary_data",
        "timestamp": "datetime",
        "metrics": { "iou": "float",
                      "diceCoefficient": "float"
                     }
}
```

D. Salt Segmentation Algorithm

1) Step 1: Data Preprocessing

The algorithm first resizes the raw seismic image to a fixed size of 128x128 pixels to ensure uniformity and compatibility with the model. It then converts the image to RGB format and normalizes the pixel values to a range of [0, 1], ensuring consistent input and improved model performance. Finally, the image is converted into a tensor format, making it suitable for efficient computation and ready for segmentation model processing.

Example:

```
# Preprocessing pipeline
transform = transforms.Compose([
   transforms.Resize((128, 128)),  # Resize image
   transforms.ToTensor(),          # Convert to Tensor
```

```
    transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])  # Normalize to range [0, 1]
])

# Apply preprocessing to the image

image = Image.open('seismic_image.jpg')
processed_image = transform(image)
```

2) Step 2: Algorithm for U-Net Model Architecture

The U-Net model starts by defining encoder layers for feature extraction and downsampling, followed by a bottleneck layer for capturing abstract features. Decoder layers are then defined to upsample and reconstruct spatial features, with skip connections ensuring fine-grained details are preserved. Finally, an output layer with a sigmoid activation is added to generate the segmentation mask, and the complete model is returned for training.

Example:

```
class UNet(nn.Module):
    def __init__(self):
        super(UNet, self).__init__()
        self.encoder = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2)
        )
            self.bottleneck = nn.Conv2d(64, 128, kernel_size=3, padding=1)
            self.decoder = nn.ConvTranspose2d(128, 64, kernel_size=3, padding=1)
            self.output_layer = nn.Conv2d(64, 1, kernel_size=1, activation='sigmoid')  # Segmentation mask

    def forward(self, x):
        x1 = self.encoder(x)
        x2 = self.bottleneck(x1)
        x3 = self.decoder(x2)
        return self.output_layer(x3)

# Initialize the model
model = UNet()
```

3) Step3:Algorithm For Model Training

The training process for the U-Net model begins by setting key parameters such as the number of epochs and batch size, which control the duration and efficiency of learning. During each epoch, the model processes a batch of training images and their corresponding labels, performing a forward pass to predict segmentation masks. The loss between the predictions and true labels is calculated, and backpropagation is used to update the model's weights to improve performance. Regular logging of the loss throughout training enables progress tracking and early detection of potential issues.

Example:

```
# Set up the optimizer and loss function
optimizer = optim.Adam(model.parameters(), lr=0.001)
criterion = nn.BCEWithLogitsLoss()

# Training loop
for epoch in range(num_epochs):
```

```
    for inputs, labels in train_loader:
        optimizer.zero_grad()
        outputs = model(inputs)  # Forward pass
        loss = criterion(outputs, labels)  # Calculate loss
        loss.backward()  # Backpropagation
        optimizer.step()  # Update weights
    print(f"Epoch {epoch+1}/{num_epochs}, Loss: {loss.item()}")
```

4) Step4: Algorithm For Model Evaluation

The model is set to evaluation mode, and test data is loaded to generate predictions by feeding the images through the trained model. The predictions are compared to the ground truth, and evaluation metrics such as accuracy or the Dice coefficient are calculated, with the results being logged for further analysis.

Example:

```
# Set model to evaluation mode
model.eval()

# Evaluation loop
with torch.no_grad():
    for inputs, labels in test_loader:
        outputs = model(inputs)  # Make predictions
        dice = dice_coefficient(outputs, labels)  # Calculate Dice coefficient
        print(f"Dice Coefficient: {dice.item()}")

# Dice coefficient function

def dice_coefficient(prediction, ground_truth):
    smooth = 1e-6
    intersection = torch.sum(prediction * ground_truth)
    return (2. * intersection + smooth) / (torch.sum(prediction) + torch.sum(ground_truth) + smooth)
```

V. TESTING AND EVALUATION

A. *Unit Testing*

1) Objective: Validate the functionality of individual modules, such as image processing, plant species identification, and model prediction.

2) Approach: Use test scripts to independently test each function, including image resizing, RGB conversion, normalization, and model predictions. Ensure that the image preprocessing pipeline works correctly, and the model produces accurate species predictions based on test images.

3) Expected Outcome: Each module works as expected, producing accurate and consistent results for various plant seismic images, with proper handling of edge cases (e.g., invalid or corrupted images).

B. *Integration Testing*

1) Objective: Ensure smooth interactions between interconnected modules, such as the frontend interface, backend server, and plant identification model.

2) Approach: Test scenarios like: Uploading images through the web interface and ensuring they are correctly preprocessed by the backend before being fed to the model. Verifying that the model returns correct species predictions, and the result is displayed in the frontend. Checking the integration of the user interface with the server, ensuring accurate communication between the two.

3) Expected Outcome: The components work together seamlessly, with correct species identification displayed on the frontend, and no data loss or errors in communication between the backend, model, and frontend.

C. System testing

1) Objective: Validate the complete workflow, ensuring the system performs as expected under realistic conditions.

2) Approach: Perform end-to-end tests, including preprocessing images, running them through the UNet model, and evaluating the segmentation results. Test the model under different image quality scenarios to ensure robustness in segmentation.

3) Expected Outcome: The system performs the full pipeline (image preprocessing, segmentation, evaluation) correctly and efficiently, providing accurate segmentation masks for salt deposits in seismic images.

D. Performance Metrics

1) Response Time

- Metric: Time taken to preprocess images, run them through the model, and generate segmentation masks.
- Expected Standard: Achieve a response time of under 5 seconds for processing and segmentation to ensure smooth operation for real-time or large-scale data processing.

2) Scalability

- Metric: The system's ability to handle high volumes of seismic image data for salt segmentation.
- Approach: Load testing with large datasets to evaluate the performance of the image processing pipeline and model inference under high load.
- Expected Standard: The system remains responsive and produces accurate segmentation outputs under peak loads, ensuring scalability for processing large sets of seismic images.

VI. RESULTS

While the system is still under development, the expected results from testing and evaluation once the salt segmentation model is operational are:

1) Functionality Validation: Each individual module (image preprocessing, UNet model architecture, training, and evaluation) is expected to perform accurately. System testing should confirm the end-to-end functionality, ensuring that images are correctly preprocessed, the model segments salt deposits accurately, and evaluation metrics (e.g., Dice coefficient, accuracy) are computed correctly.

2) User Feedback Insights: User feedback from domain experts (e.g., geologists, data scientists) is anticipated to highlight the model's accuracy in detecting salt deposits and its ease of use. Suggestions for model improvements, such as handling different types of seismic images or enhancing segmentation precision, may be provided to refine both the model and the user interface.

3) Performance and Scalability Metrics: Response times for preprocessing, model inference, and segmentation are expected to meet or exceed the benchmark of under 5 seconds per image. Load tests should confirm that the system can efficiently process large datasets of seismic images without significant delays or performance degradation, ensuring its scalability for high-volume applications.

VII. CONCLUSION

The SaltSegNet platform offers an innovative and effective solution for salt deposit detection using advanced image processing and machine learning techniques, specifically employing the UNet architecture for accurate segmentation. One of the key strengths of this system is its ability to preprocess seismic images and utilize deep learning models to accurately identify and segment salt deposits, providing geologists and researchers with valuable insights into subsurface structures.

Future work for SaltSegNet will focus on further optimizing the model for faster inference times and improving its robustness against diverse seismic data. Additionally, the integration of AI-driven analytics to predict potential salt-rich regions and automate data analysis is planned. The system will also expand to handle more complex datasets and real-time segmentation, ensuring scalability and enhancing its usability in large-scale geological studies.

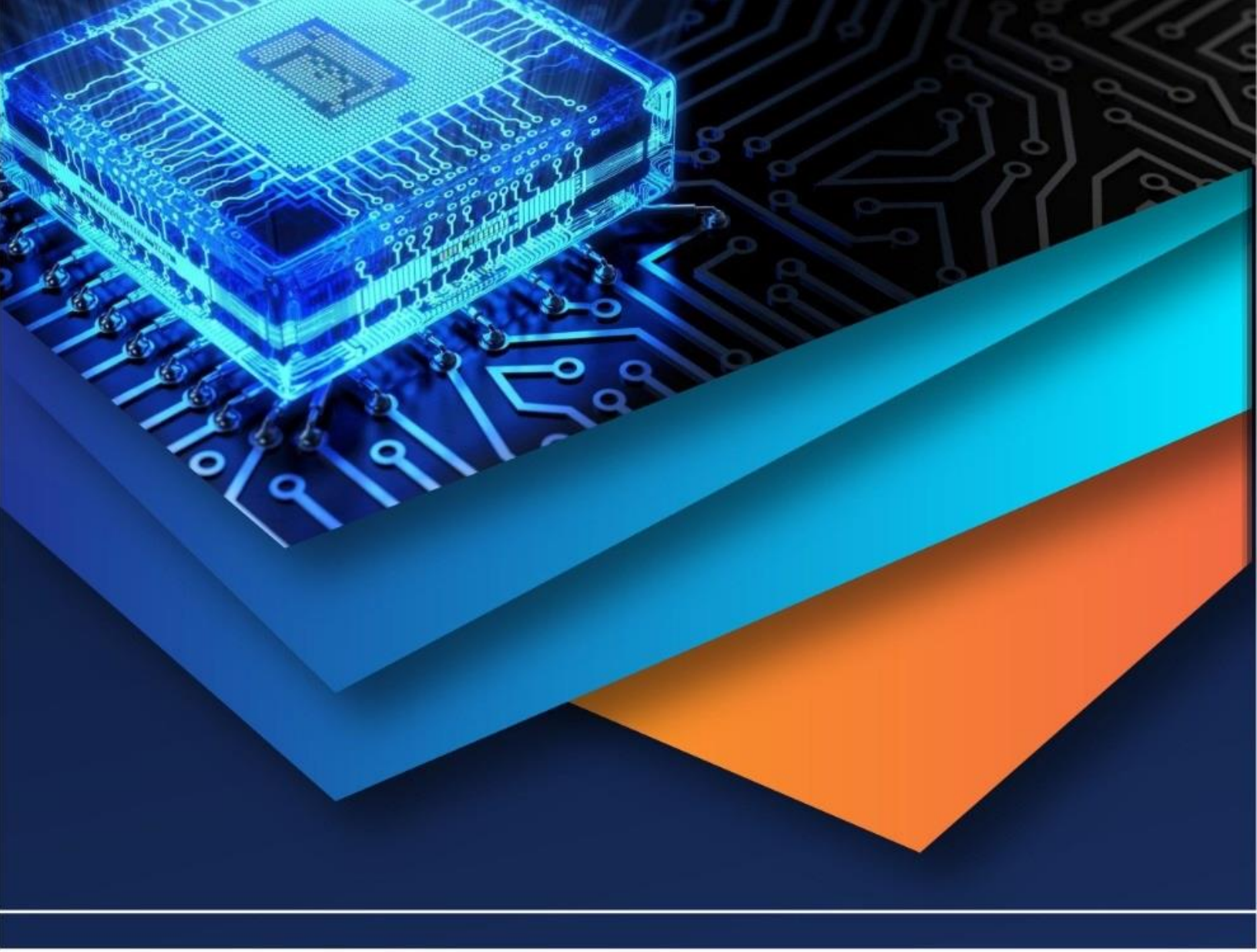
VIII. ACKNOWLEDGMENT

We would like to express our sincere gratitude to Vidya R, Assistant Professor, Department of Computer Science and Engineering, Bangalore Institute of Technology, for their invaluable guidance, support, and mentorship throughout the course of this research. Their insights and suggestions were instrumental in shaping this project and enhancing its overall quality.

We would also like to thank Bangalore Institute of Technology for providing the necessary resources and infrastructure that facilitated the successful completion of this research.
Finally, we extend our appreciation to family and friends for their continuous support and motivation during this research journey.

REFERENCES

[1] Gregorius Airlangga, “Advanced Seismic Data Analysis: Comparative study of Machine Learning and Deep Learning for Data Prediction and Understanding”, Research of Artificial Intelligence, Volume 3, Number 2, November 2023.
[2] Muhammad Saif ul Islam, “Using deep learning based methods to classify salt bodies in seismic images”, Journal of Applied Geophysics, May 2020.
[3] Jyostna Devi Bodapati, RamaKrishna Sajja, Veeranjaneyulu Naralasetti, “An efficient Approach for Semantic Segmentation of Salt Domes in Seismic Images Using Improved UNET Architecture”, Journal of The Institute of Engineers(India), Volume 104, Pages 569-578, April 2023.
[4] Yang Xue, Xiwu Luan, “Implications of Salt Tectonics on Hydrocarbon Ascent in the Eastern Persian Gulf: Insights into the Formation Mechanism of Salt Diapirs, Gas Chimneys, and Their Sedimentary Interactions”, Journal of Ocean University of China, July 2024.
[5] Mirnomoy Sarkar, “Salt Detection Using Segmentation of Seismic Image”, Journal of A&T State University, March 2022.
[6] Zhifeng Xu, Kewen Li, “3D Salt-net: a method for salt body segmentation in seismic images based on sparse label”, Volume 53, pages 29005–29023, 2023.
[7] Yelong Zhao, Bo Liu, “Boundary U-Net: A Segmentation Method to Improve Salt Bodies Identification Accuracy”, Conference Paper, January 2022.
[8] Mikhail Karchevskiy, Insaf Ashrapov, Leonid Kozinkin, “Automatic salt deposits segmentation: A deep learning approach”, 2018.
[9] Haolin Wang, Lihui Dong, Song Wei, “Improved U-Net-Based Novel Segmentation Algorithm for Underwater Mineral Image”, Minzu University of China, Vol.32, No.3, 2022.
[10] Dmitry Koroteev, Zeljko Tekic, “Artificial intelligence in oil and gas upstream: Trends, challenges, and scenarios for the future”, Journal on Energy and AI, March 2021.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY