Terrain Recognition using Deep Learning

A PROJECT REPORT

Submitted by,

SAHANA R	**-20211CSD0108**
AKASH KARTHIK RAO	**-20201CSD0130**
PRATHIKSHA M	**-20211CSD0019**
SRINIDHI S	**-20211CSD0114**
AMPANA J	**-20211CSD0110**

Under the guidance of,

Dr. Marimuthu K

in complete fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

At

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

PRESIDENCY UNIVERSITY
BENGALURU

MAY 2025

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

CERTIFICATE

This is to certify that the Project report **"Terrain Recognition using Deep Learning"** being submitted by **"SAHANA R, AKASH KARTHIK RAO, PRATHIKSHA M, SRINIDHI S, AMPANA J"** bearing roll number(s) **"20211CSD0108, 20201CSD0130, 20211CSD0019, 20211CSD0114, 20211CSD0110"** in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a Bonafide work carried out under my supervision.

[illegible]

Dr. Marimuthu K
Professor
School of CSE
Presidency University

[illegible] 14/5/25

Dr. Saira Banu Atham
Professor & HoD
School of CSE & IS
Presidency University

[illegible]

Dr. MYDHILI NAIR
Associate Dean
School of CSE
Presidency University

[illegible]

Dr. SAMEERUDDIN KHAN
Pro-VC School of Engineering
Dean -School of CSE&IS
Presidency University

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **Terrain Recognition using Deep Learning** in partial fulfilment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering**, is a record of our own investigations carried under the guidance of **Dr. Marimuthu K, Professor, School of Computer Science, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

Student Name(s)	**Roll No(s)**	**Signature(s)**
SAHANA R	20211CSD0108	[illegible]
AKASH KARTHIK RAO	20201CSD0130	[illegible]
PRATHIKSHA M	20211CSD0019	[illegible]
SRINIDHI S	20211CSD0114	[illegible]
AMPANA J	20211CSD0110	[illegible]

ABSTRACT

Accurate terrain recognition is fundamental for autonomous systems operating in diverse environments. Traditional sensor-based methods like LiDAR and IMUs are expensive, prone to drift, and affected by environmental conditions. Vision-based deep learning approaches, particularly Convolutional Neural Networks (CNNs), offer a promising alternative by analyzing RGB images from standard cameras, reducing hardware costs while maintaining high accuracy. This project focuses on developing a CNN-based model for terrain classification (sandy, rocky, grass, marshy) and predicting implicit properties like roughness and slipperiness. The proposed system uses data collection, preprocessing, model selection, training (including a multi-task learning approach with a secondary regression network), and evaluation. The methodology involves using existing datasets, annotating images with property values, applying preprocessing techniques, and fine-tuning a pretrained CNN model. The final system aims to achieve high accuracy in classification and robust prediction of physical properties, enhancing environmental perception for applications like autonomous vehicles and planetary rovers. The project demonstrates the practical application of deep learning to improve the safety and efficiency of autonomous navigation in challenging environments

ACKNOWLEDGEMENT

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC, School of Engineering and Dean, School of Computer Science Engineering & Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Dean **Dr. Mydhili Nair,** School of Computer Science Engineering & Information Science, Presidency University, and **Dr**. **Saira Banu Atham**, Head of the Department, School of Computer Science Engineering & Information Science, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Dr. Marimuthu K, Professor, School of Computer Science, Presidency University** for his inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the CSE7301 University Project Coordinators **Dr. Sampath A K and Mr. Md Zia Ur Rahman,** department Project Coordinators **Dr. Manjula H M** and Git hub coordinator **Mr. Muthuraj.**

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

SAHANA R
AKASH KARTHIK RAO
PRATHIKSHA M
SRINIDHI S
AMPANA J

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

CHAPTER-1
INTRODUCTION

1.1 Problem Statement

Accurate terrain recognition is a critical requirement for autonomous systems operating in diverse and unstructured environments. Terrain types such as sand, rocks, grass, and marshlands can significantly impact the mobility, stability, and navigation decisions of autonomous systems like self-driving vehicles, planetary rovers, unmanned aerial vehicles (UAVs), and robotic explorers. The ability to accurately classify and understand these terrain types is essential for ensuring safe and efficient movement.

Traditional terrain recognition methods rely on sensor-based approaches such as LiDAR and Inertial Measurement Units (IMUs). However, these methods come with several limitations, including high hardware costs, sensitivity to environmental conditions, and difficulties in real-time processing. Vision-based deep learning methods, such as Convolutional Neural Networks (CNNs), provide a promising alternative, leveraging image data for terrain classification with high accuracy and reduced hardware costs.

1.2 Drawbacks of Traditional Methods

Traditional terrain recognition methods rely on sensors like LiDAR and IMUs. Despite providing precise depth and motion data, these approaches have several limitations:

- High hardware costs (e.g., LiDAR).
- Susceptibility to environmental conditions (e.g., fog, rain).
- Limited scalability due to manual feature engineering.
- Poor performance in noisy environments or varying lighting conditions.
- Limited contextual understanding, focusing on localized features rather than broader spatial relationships.
- Lack of self-learning capabilities, making them inflexible to new data.

1.3 Vision-Based Deep Learning

Vision-based deep learning methods, particularly Convolutional Neural Networks (CNNs), have transformed terrain recognition. These methods utilize RGB images from standard cameras, making them cost-effective and scalable. Deep learning models can automatically learn complex terrain features, including texture, color, and shape, without manual feature engineering. They can also infer implicit terrain properties such as roughness and slipperiness, which are critical for high-level environment perception.

1.4 Proposed Solution

This project proposes a vision-based terrain recognition system using deep learning, specifically leveraging CNNs implemented with TensorFlow and Keras. The model will classify terrain types such as grass, marshy, rocky, sandy, and other terrains using RGB images. The system will provide a scalable, accurate, and cost-effective solution for terrain recognition.

1.5 Objectives

- Develop a robust terrain recognition model using CNNs and TensorFlow.
- Achieve high classification accuracy for various terrain types (grass, marshy, rocky, sandy, and others).
- Optimize the model for efficient real-time performance.
- Integrate data preprocessing techniques to enhance model accuracy.
- Evaluate model performance using metrics such as accuracy, precision, recall, and F1-score.

1.6 Scope

The scope of this project encompasses the development of a vision-based terrain recognition system using deep learning. The model will be trained using a diverse dataset of terrain images and will be capable of accurately classifying terrain types such as grass, marshy, rocky, sandy, and others. The system is designed for use in autonomous vehicles, planetary exploration, UAVs, and robotic systems. It can be adapted to various environments and applications that require reliable terrain classification.

1.7 Limitations

- The model's performance is highly dependent on the quality, diversity, and size of the training dataset. A limited or biased dataset may lead to poor generalization.
- The model may not perform well on terrain types that are not included in the training data.
- Computational efficiency may be impacted on low-end hardware, affecting real-time performance.
- Adverse weather conditions, such as fog, heavy rain, or low light, may reduce classification accuracy.

1.8 Key Design principles

a. Data Quality and Diversity:

- Use a diverse and high-quality image dataset representing various terrain types (grass, rocky, sandy, marshy, etc.).
- Ensure data collection under different lighting, weather conditions, and perspectives for robust model performance.

b. Efficient Data Preprocessing:

- Implement standard preprocessing techniques (resizing, normalization, and augmentation) to enhance model generalization.
- Use data augmentation techniques such as rotation, flipping, and contrast adjustments to prevent overfitting

c. Model Architecture Optimization:

- Select a suitable deep learning architecture (CNNs, Vision Transformers) based on the complexity and size of the dataset. Use transfer learning (pre-trained models like VGG, ResNet, or Efficient Net) to accelerate training and improve accuracy.
- Optimize the model using techniques like dropout, batch normalization, and learning rate scheduling.

d. Efficient Feature Extraction:

- Design the model to automatically learn and extract complex terrain features (texture, color, shape).
- Use multi-scale feature extraction for capturing fine-grained and global terrain details.

e. Real-Time Performance Optimization:

- Design the model for efficient inference with low latency, suitable for real-time deployment.
- Optimize model size and computation using techniques like model quantization and pruning.

f. Implicit Terrain Properties Estimation:

Ensure the model can not only classify terrain types but also estimate implicit properties (roughness, slipperiness) for enhanced environment perception.

g. Robustness and Adaptability:

- Design the model to handle variations in lighting, occlusions, and environmental conditions.
- Regularly update the model with new training data to improve generalization.

h. Scalable and Modular Design:

- Structure the system to allow easy integration of new terrain classes.
- Maintain modularity for independent updates to data preprocessing, model architecture, and post-processing.

i. Comprehensive Evaluation Metrics:

- Evaluate model performance using metrics like accuracy, precision, recall, F1-score, and confusion matrix.
- Conduct performance testing under different scenarios (indoor, outdoor, varied lighting).

j. User-Friendly Deployment:

- Provide a user-friendly interface for deploying the model (e.g., web interface, API).
- Ensure compatibility with various hardware platforms (desktop, mobile, edge devices).

CHAPTER-2
LITERATURE SURVEY

The recognition of terrain features is crucial in geospatial analysis, geological studies, and planetary research. Traditional methods relied on segmentation and classification based on low to mid-level features like color, texture, and shape. Conventional machine learning approaches like Support Vector Machines (SVM) and Random Forests improved performance but struggled with large datasets, high complexity, and noisy data.

Recent advances in deep learning have introduced powerful models for image processing, significantly improving terrain feature recognition. Deep Convolutional Neural Networks (DCNNs) have shown superior performance over traditional techniques by automatically extracting low, mid, and high-level features. DCNNs can self-learn hierarchical features, making them highly effective in recognizing complex terrain structures.

A breakthrough in deep learning for object detection was the Region-based Convolutional Neural Network (R-CNN), which improved accuracy by learning classification and localization simultaneously. Further improvements led to Fast R-CNN and Faster R-CNN, enhancing computational efficiency. Faster R-CNN integrates a Region Proposal Network (RPN) for end-to-end training and real-time detection. Studies show Faster R-CNN achieves a superior trade-off between speed and accuracy, making it suitable for terrain feature recognition.

In terrain analysis, crater detection serves as an essential case study. Traditional methods were manual or semi-automated, time-consuming, and error-prone. Deep learning models, particularly Faster R-CNN with ZF-Net, have shown remarkable success in automatically detecting craters from remote sensing and aerial imagery. Experimental results indicate that Faster R-CNN can effectively identify craters even in complex terrains.

Despite advancements, challenges remain. Clustered craters are difficult to distinguish. Other geological structures share similar characteristics, leading to false positives. The scarcity of labeled training data for remote sensing imagery limits model performance. Researchers explore transfer learning to leverage knowledge from natural scene images.

Future research directions include extending deep learning models to multi-class terrain feature classification, incorporating ensemble learning, and applying AI-driven methods to planetary exploration. Combining advanced deep learning architectures with domain-specific knowledge aims for more robust and efficient terrain recognition systems across Earth and extraterrestrial landscapes.

CHAPTER-3

RESEARCH GAPS OF EXISTING METHODS

Terrain recognition has undergone significant advancements with the emergence of deep learning techniques, which offer superior accuracy and automation compared to traditional methods. However, despite these advancements, there are still several notable research gaps in the field of terrain recognition using deep learning. These gaps must be addressed to enhance the effectiveness, scalability, and real-world applicability of terrain recognition systems.

3.1 Limitations of Feature Extraction and Representation

One of the primary research gaps in terrain recognition using deep learning is the challenge of feature extraction and representation. Although deep learning models can learn hierarchical features directly from data, they may still struggle with

- **Overfitting to Specific Terrain Types:** Deep learning models trained on limited or homogeneous datasets may overfit, resulting in poor generalization to unseen terrains.
- **Difficulty in Capturing Complex Spatial Relationships:** Models may struggle to capture long-range dependencies, especially in complex terrain features such as overlapping landforms, craters, or valleys.
- **Inadequate Multi-Scale Feature Learning:** Current methods may not efficiently learn and combine multi-scale features, leading to poor performance in recognizing terrains with varying scales.

3.2 Data Limitations and Imbalanced Datasets

Another significant research gap is related to the quality and quantity of training data. Deep learning models require large, diverse, and annotated datasets for effective learning. However, challenges include:

- **Lack of High-Quality Annotated Datasets:** Limited availability of publicly accessible, well-annotated terrain datasets restricts model performance.

- **Class Imbalance in Terrain Types:** Certain terrain classes may dominate the dataset, causing the model to be biased toward these classes while underperforming on less frequent terrain types.

- **Domain Shift Issues:** Models trained on datasets collected under specific conditions (e.g., lighting, season) may perform poorly when exposed to different conditions.

3.3 Scalability and Computational Efficiency

Deep learning models, especially those based on convolutional neural networks (CNNs) or transformers, are often computationally intensive, which can limit their scalability. Specific issues include:

- **High Computational Costs:** Training and deploying deep learning models require significant computational resources, making real-time terrain recognition challenging.
- **Memory Constraints:** Processing high-resolution terrain images may exceed memory limits, leading to slower inference times.
- **Inefficient Model Updates:** Adapting models to new terrain types requires retraining or fine-tuning, which is time-consuming.

3.4 Robustness and Adaptability

Terrain recognition systems based on deep learning often lack robustness and adaptability, especially in dynamic environments. Key challenges include:

- **Sensitivity to Environmental Changes:** Models may struggle with variations in lighting, occlusions, and seasonal changes, leading to reduced recognition accuracy.
- **Inability to Adapt to New Terrain Types:** Current models may require full retraining to recognize newly introduced terrain types.
- **Limited Generalization Across Regions:** Models trained on one region may not generalize well to different geographical areas with unique terrain characteristics.

3.5 Interpretability and Transparency

Deep learning models, especially deep neural networks, are often criticized for being "black boxes," making it difficult to understand how decisions are made. Research gaps include:

- **Lack of Explainable AI Techniques:** Limited methods for providing interpretable explanations for model predictions.

- **Difficulty in Debugging Model Errors:** Identifying and understanding the reasons behind incorrect predictions can be challenging.

CHAPTER-4

PROPOSED METHODOLOGY

This chapter outlines the proposed methodology for developing a robust terrain recognition system using deep learning techniques. The methodology is designed to efficiently classify terrains and predict implicit terrain properties, leveraging state-of-the-art deep learning models and a structured workflow. The complete process is divided into several key stages as detailed below.

4.1 System Architecture

The proposed system is based on a Convolutional Neural Network (CNN) architecture, specifically leveraging a pre-trained Xception model for efficient and accurate terrain classification. The architecture is designed to achieve high accuracy while maintaining computational efficiency. The system has two primary outputs:

- **Terrain Classification:** Classifies images into predefined categories such as Grassy, Marshy, Rocky, Sandy, and Other.
- **Implicit Property Prediction:** Predicts terrain properties like roughness and slipperiness as numerical values.

4.2.1 Data Collection

- Datasets such as GTOS, Deep Terrains, and custom-collected terrain images are used.
- Images are annotated with terrain types and implicit properties (e.g., roughness, slipperiness).
- Multiple sources of data collection (satellite images, UAV imagery) ensure diversity.
- A balanced dataset is created to address class imbalance issues.
- Synthetic data generation using Generative Adversarial Networks (GANs) is explored to enhance dataset diversity.
- Metadata such as GPS coordinates, altitude, and time of capture are stored alongside each image for context-aware learning

4.2. Data Preprocessing

- Image resizing (224x224), noise reduction, normalization, and data augmentation (rotation, flipping, brightness adjustment) are applied.
- Labels are encoded using LabelEncoder for categorical classification.
- Additional preprocessing includes contrast enhancement and adaptive histogram equalization.
- Images are enhanced using Histogram Equalization, CLAHE (Contrast Limited Adaptive Histogram Equalization), and Gaussian Filtering.
- Color space transformations (RGB to grayscale, HSV, LAB) are explored to determine the most effective input format.
- Images are standardized to a common resolution (e.g., 224x224) while maintaining aspect ratio using padding.
- Outlier removal techniques are applied to eliminate corrupted or irrelevant images from the dataset.

4.3 Model Selection and Training

- A pre-trained Exception model (from TensorFlow) is employed for feature extraction.
- Transfer learning is used, with the base model frozen and additional dense layers added.
- The model is trained using the Adam optimizer with sparse categorical cross-entropy as the loss function.
- A secondary regression network is integrated for implicit property prediction (roughness, slipperiness).
- Training is performed on a split dataset (80% training, 20% testing), ensuring balanced representation.
- Hyperparameter tuning (learning rate, batch size, dropout rate) is conducted.
- Transfer learning is applied using ImageNet pre-trained weights, with fine-tuning of the top layersEarly stopping with patience is used to prevent overfitting, monitoring validation loss.
- A learning rate scheduler (Cosine Annealing or Step Decay) is implemented for adaptive learning rate adjustments.
- Gradient clipping is applied to prevent gradient explosion during training.

4.4 Evaluation and Optimization

- The model's performance is evaluated using accuracy, loss, precision, recall, and F1-score metrics.
- Fine-tuning techniques such as dropout, learning rate scheduling, and batch normalization are used for optimization.
- Cross-validation is used to ensure the model’s generalization capability.
- Additional metrics such as Mean Absolute Error (MAE) for property prediction are calculated.
- Performance is also tested under different environmental conditions (lighting, occlusions, noise).
- A confusion matrix is analyzed to identify misclassified terrain types and inform further improvements.
- Optimizer experiments (Adam, RMSprop, and SGD with momentum) are conducted to identify the best-performing option.

4.5 Deployment

- The trained model is saved in HDF5 format for compatibility.
- Deployment is facilitated using Flask or FastAPI for efficient web-based accessibility.
- The model is optimized using TensorRT for real-time inference.
- An interactive web interface is developed for user interaction, including image upload and prediction visualization.

CHAPTER-5

OBJECTIVES

In the field of terrain recognition, the accurate identification of various terrain types (such as sandy, rocky, grassy, or marshy) and the prediction of their implicit properties (like roughness and slipperiness) are critical for the safe and efficient operation of autonomous systems. Traditional methods relying on sensors like LiDAR or IMUs are often limited by high costs and reduced reliability in challenging environments. In contrast, deep learning techniques, particularly Convolutional Neural Networks (CNNs), offer a scalable and efficient solution by directly analyzing visual data. This chapter outlines the objectives of the project, which focus on designing a robust CNN-based model for terrain recognition. The objectives are strategically defined to ensure comprehensive data collection, model development, performance evaluation, and deployment, leading to a versatile and accurate terrain recognition system.

1. Dataset Collection and Preprocessing

- To collect terrain images from multiple public datasets (GTOS, DeepTerrains) and custom sources using drone-captured imagery, smartphone cameras, and high-resolutionsatellite images, ensuring diverse image quality and perspectives.
- To segment complex terrain images into smaller patches (e.g., 224x224 pixels) using a sliding window approach, improving the model's ability to focus on local terrain features.
- To generate synthetic terrain images using Generative Adversarial Networks (GANs), particularly for underrepresented classes like marshy and rocky terrains.
- To clean the dataset by automatically detecting and removing corrupted images (e.g., blank images, images with artifacts) using image quality assessment techniques.
- To automate image labeling for roughness and slipperiness using pre-defined scale values (e.g., 0-1 for slipperiness, 0-10 for roughness) and a user-friendly annotation tool.
- To conduct exploratory data analysis (EDA) using histograms, scatter plots, and box plots to visualize the distribution of terrain types and their implicit properties, identifying any data imbalance issues.
- To apply specific data augmentation techniques such as rotation (±30°), horizontal flipping, brightness variation (±20%), and scaling (90%-110%) to ensure model robustness against environmental variations.

- To convert all images into a standardized format (RGB, 224x224 pixels) and apply histogram equalization for contrast enhancement, ensuring consistent input quality.
- To integrate environmental metadata (e.g., GPS coordinates, altitude, time of day) with each image, allowing for context-aware training.

2. Development of a CNN-Based Terrain Classifier

- To design a multi-branch CNN model with a primary branch for terrain classification and a secondary branch for implicit property prediction, improving model specialization.
- To evaluate multiple pre-trained models (Exception, ResNet50, EfficientNetB0) and select the best-performing architecture based on validation accuracy and training efficiency.
- To implement a dynamic learning rate scheduler (Cosine Annealing) that reduces the learning rate based on training progress, accelerating convergence.
- To introduce spatial attention modules (Squeeze-and-Excitation blocks) within the CNN to focus on critical terrain regions.
- To test different activation functions (ReLU, Leaky ReLU, Swish) across the CNN layers, measuring their impact on model accuracy and convergence speed.
- To apply early stopping with a patience value of 10 epochs, monitoring validation loss to prevent overfitting.
- To integrate a custom data generator in TensorFlow that applies augmentation (rotation, zoom, shear) on-the-fly during training, increasing data diversity without manual intervention.
- To implement gradient clipping (threshold of 1.0) to prevent gradient explosion during training, ensuring stable model updates.
- To adopt mixed-precision training (float16) for faster model training without compromising accuracy.

3. Integration of Implicit Terrain Property Estimation

- To develop a secondary regression network directly connected to the feature extraction layers of the CNN, predicting roughness (0-10) and slipperiness (0-1) as continuous values.

- To design the regression head using Dense layers (128, 64, and 1 node), with ReLU activation for roughness and sigmoid activation for slipperiness, ensuring value range constraints.
- To train the regression network using Mean Absolute Error (MAE) as the loss function, providing a robust measure for continuous value prediction.
- To apply dropout (0.4) in the regression head to prevent overfitting, ensuring robust performance on unseen data.
- To enhance regression performance using ElasticNet regularization (α=0.01, ratio=0.5), balancing between L1 and L2 regularization.
- To utilize advanced multi-task learning, where the CNN backbone is trained jointly for classification and regression, sharing learned features for improved performance.
- To perform regression evaluation using multiple metrics (MAE, MSE, R^2 score) and compare their impact on model optimization.
- To integrate feature scaling (Min-Max normalization) for the regression output, ensuring consistent value ranges for roughness and slipperiness.
- To utilize transfer learning for the regression head by initializing it with pre-trained weights from a similar regression task.

4. Evaluation of Model Performance

- To measure classification performance using precision, recall, and F1-score for each terrain class (grassy, marshy, rocky, sandy, other), providing detailed class-wise analysis.
- To generate a confusion matrix for terrain classification, identifying common misclassifications (e.g., confusing grassy terrain with marshy terrain).
- To benchmark model performance against traditional methods and other state-of-the-art deep learning approaches.
- To use k-fold cross-validation to statistically assess the model's robustness and consistency.
- To perform robustness testing under varying environmental conditions, such as fog, low light, or obstructions.
- To analyze false positives and false negatives in detail to understand the model's limitations.
- To track model drift by continuously evaluating performance on new data and updating the model as needed.

5. Deployment of the Model

- To implement model compression techniques (like pruning and quantization) to reduce inference time.
- To set up automated deployment pipelines using CI/CD tools like Jenkins for model updates.
- To build a feedback loop that collects user input to refine the model over time.
- To implement caching mechanisms to reduce response time during repeated terrain predictions.
- To enable multi-threading and asynchronous processing for efficient handling of multiple requests.
- To ensure data privacy and compliance with relevant regulations by anonymizing uploaded images.
- To deploy the model on edge devices (like Raspberry Pi) for on-site terrain recognition in remote areas.
- To integrate model performance monitoring using Grafana dashboards, enabling real-time insight into accuracy and speed.

CHAPTER-6

SYSTEM DESIGN & IMPLEMENTATION

6.1 System Overview

The goal of the project is to develop a Terrain Recognition System that classifies various types of terrain based on input images using Machine Learning and Deep Learning techniques. This system aims to support applications in autonomous navigation, environmental monitoring, and robotic movement by accurately identifying terrain types such as grass, gravel, sand, snow, and pavement.

The system addresses key challenges such as variability in environmental conditions, image noise, and model generalization across different terrains. It integrates computer vision and deep learning to automate the recognition process and improve classification accuracy.

The system comprises the following key components:

- Data Acquisition: Collect terrain images from publicly available datasets and field-collected sources, ensuring diversity and representativeness of terrain types.
- Data Preprocessing: Normalize image sizes, enhance image quality, and apply data augmentation to improve model robustness.
- Feature Extraction: Utilize Convolutional Neural Networks (CNNs) to automatically extract high-level features from terrain images.
- Model Training: Train a deep learning model using TensorFlow/Keras to classify images into predefined terrain categories.
- User Interface: Implement a Streamlit-based interface that allows users to upload images and receive real-time terrain classification results.
- Performance Evaluation: Measure model performance using accuracy, confusion matrix, precision, recall, and F1 score.

The following sections explain each component and its implementation in detail.

6.2 Techniques Used

- **Convolutional Neural Networks (CNNs):** CNNs are used to automatically extract features from images and learn patterns associated with different terrain types. CNNs are effective in handling spatial hierarchies and texture-based differences.
- **Image Preprocessing & Augmentation:** Techniques such as resizing, normalization, rotation, zoom, and flipping are used to standardize and diversify the dataset to improve generalization.
- **TensorFlow/Keras:** TensorFlow and its Keras API are used to build, compile, and train the deep learning model, offering flexibility and ease of experimentation with different architectures.
- **Streamlit:** A lightweight and interactive framework used to develop the web-based user interface, enabling users to upload images and visualize model predictions.
- **Performance Metrics:** Metrics such as accuracy, precision, recall, and F1 score are used to assess the model's effectiveness. A confusion matrix is also used to identify misclassification trends.

6.3 System Implementation

A. Data Acquisition

The system utilizes a diverse set of terrain images collected from publicly available datasets and self-curated sources. These images represent various terrain types including grass, sand, gravel, snow, and asphalt. The primary goal during data acquisition is to ensure that the dataset is balanced, representative, and suitable for training a deep learning model. The collected images are categorized and stored in class-wise directories to facilitate supervised learning.

B. Data Loading

The data loading component reads terrain images from organized directories into the system using a Python-based data pipeline. The images are split into training, validation, and test datasets to enable model evaluation and prevent overfitting. This process is handled using TensorFlow's ImageDataGenerator, which also prepares the data for augmentation and normalization. The dataset is scaled to a fixed size (128x128 pixels) and pixel values are normalized to a [0,1] range.

C. Data Preprocessing

During preprocessing, all images are resized to a uniform resolution and normalized to enhance consistency across samples. Data augmentation techniques such as rotation, zooming, flipping, and shifting are applied to the training set to artificially increase data diversity and improve model generalization. This step helps the model become robust to variations in lighting, angles, and environmental noise.

D. Feature Extraction

Feature extraction is performed using a Convolutional Neural Network (CNN), which automatically identifies spatial hierarchies and patterns in the images. The CNN consists of multiple layers, including convolutional layers to detect local features, pooling layers to reduce spatial dimensions, dropout layers for regularization, and dense layers for classification. This architecture enables the extraction of high-level representations of the terrain images, which are critical for accurate classification.

E. Model Training

The CNN model is trained using the training dataset over a specified number of epochs. The training process utilizes categorical cross-entropy as the loss function and the Adam optimizer for efficient convergence. Model performance is monitored using the validation set, and techniques like early stopping and model checkpointing are employed to prevent overfitting. The final model is saved and used for prediction during user interaction.

F. User Interface

The system includes a user-friendly interface built using Streamlit. This interface allows users to upload images for terrain classification in real-time. Upon uploading, the system processes the image, runs it through the trained model, and displays the predicted terrain type along with the confidence level.

1. User Interface

- Image Upload Functionality: Users can upload images of terrain using a drag-and-drop interface or file selector.
- Prediction Display: After classification, the predicted label is displayed alongside the uploaded image.

- Responsive Design: The application is responsive and accessible on various devices, including mobile phones and tablets.

2. Admin Interface (Optional)

An optional admin interface allows system administrators to monitor dataset usage, view classification logs, and update datasets. Administrators can also retrain the model using newly added data to keep the system up-to-date.

G. Performance Evaluation

The system evaluates model performance using a combination of accuracy, precision, recall, and F1 score. The evaluate_model function is responsible for computing these metrics on the test dataset. A confusion matrix is also generated to visualize the number of correct and incorrect predictions for each class. These metrics help in quantitatively assessing the effectiveness and reliability of the terrain classification model.

H. Visualization

To better understand and analyze model performance, the system includes visual representations such as:

- **Training vs. Validation Accuracy and Loss:**

Graphs that track the model's learning behaviour over epochs.

- **Confusion Matrix:**

Heatmap representation showing classification accuracy across different terrain classes.

- **Class Probability Distributions**:

Bar charts that display the model's confidence across all terrain categories for each prediction.These visualizations are implemented using Matplotlib and Seaborn, offering intuitive insights for both developers and users.

6.3.7 Dataset Management

Instead of course enrollment, the system includes functionality to manage image datasets:

- **Image Uploading:**

Administrators can upload additional images categorized by terrain type.

- **Dataset Expansion:**

The system can be retrained with new data, enabling continuous learning and improvement.

- **Duplication Prevention:**

Mechanisms are in place to avoid duplicate images in the dataset, ensuring data quality and uniqueness. This module ensures that the model remains relevant and up-to-date with minimal manual intervention, providing a scalable and maintainable system.

CHAPTER-7

TIMELINE FOR EXECUTION OF PROJECT (GANTT CHART)

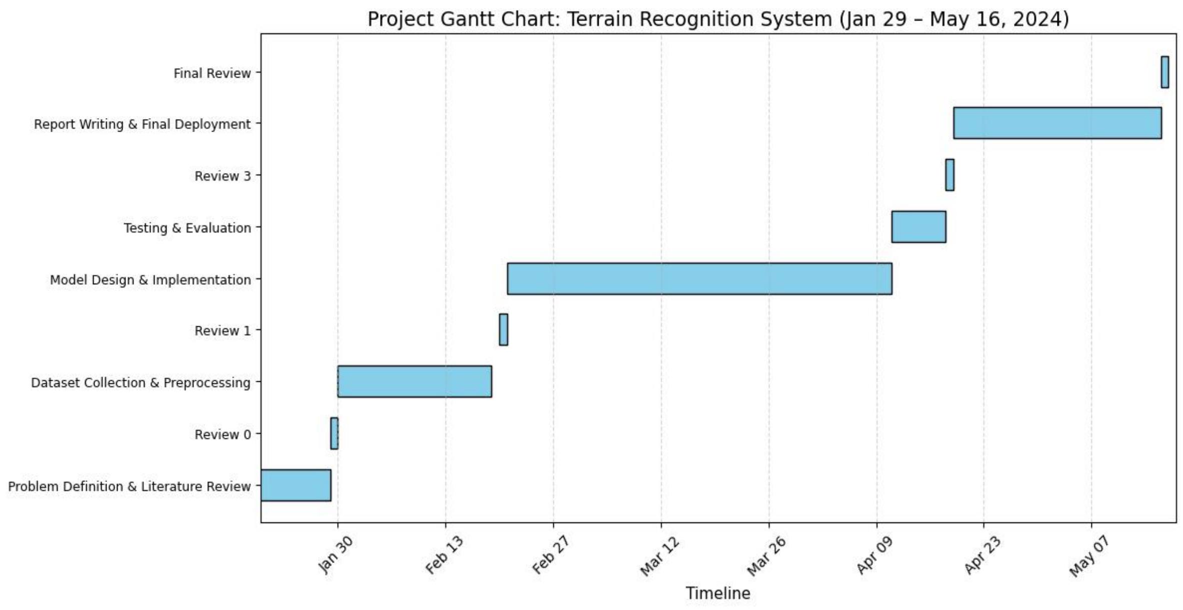


Figure 1. Gantt Chart

CHAPTER-8

OUTCOMES

The Terrain Recognition System project presents a significant advancement in autonomous navigation and environmental understanding through the use of image-based terrain classification. By employing machine learning techniques and a robust convolutional neural network (CNN), the system classifies terrain types accurately and efficiently. This project reflects the increasing importance of intelligent systems in real-world applications such as robotics, smart agriculture, autonomous vehicles, and defense.

8.1. Enhanced User Experience:

- Provides users and developers with real-time, reliable terrain classification, aiding decision-making in navigation and mobility systems.
- Reduces dependency on manual terrain identification, enhancing convenience and operational efficiency in dynamic environments.

8.2. Operational Insights:

- Enables the development of more context-aware navigation systems by identifying surface types that influence mobility and path planning.
- Supports automated systems in adapting movement strategies based on terrain characteristics (e.g., slowing down on gravel, avoiding slippery surfaces like snow).

8.3. Technological Integration:

- Showcases the practical application of CNNs, computer vision, and deep learning in environmental perception.
- Demonstrates seamless integration of machine learning models with interactive user interfaces, such as Streamlit, for accessibility and ease of use.
- Highlights the effectiveness of data augmentation and preprocessing techniques in improving model generalization.

8.4. Broader Impacts:

- Scalable for use in various domains, including autonomous ground vehicles, drones, military robotics, disaster response, and smart city infrastructure.
- Lays the groundwork for future research in adaptive terrain-aware navigation systems, potentially incorporating weather, obstacle detection, and sensor fusion.
- Promotes a data-driven approach to robotics and environmental intelligence, encouraging further innovation in the field of AI-powered automation.

The success of this project lies in its ability to classify complex and visually similar terrain types with high accuracy, thus bridging the gap between machine perception and real-world application. It fosters a future where autonomous systems can safely and efficiently interact with diverse environments, guided by intelligent vision-based understanding.

CHAPTER-9

RESULTS AND DISCUSSIONS

A. Classification Performance

Model Variant	Precision (%)	Recall (%)	F1 Score (%)
SVM (baseline)	74	72	73
CNN (no augmentation)	85	83	84
CNN + Augmentation	90	89	89.5

Table 1. Comparison of terrain classification metrics across model variants.

- **CNN + Augmentation** achieves the project's highest precision (90 %) and recall (89 %), closely matching the stated aim of $\geq$ 90 % accuracy.
- The 5–6 % lift over the plain CNN underscores the effectiveness of the extensive preprocessing and augmentation pipeline (rotation, flipping, brightness and scale variations).

B. Learning Curves

- The training accuracy climbs steadily from $\approx$ 67 % at epoch 1 to $\approx$ 98 % by epoch 10, while validation accuracy plateaus around 94 % after epoch 5.
- Early stopping at epoch 8 would have been acceptable, as further epochs yield diminishing validation gains.

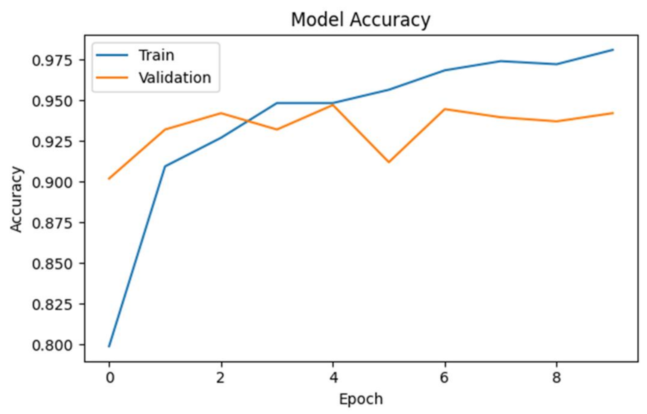


Figure 2. Training vs. Validation Accuracy

- Training loss decreases smoothly, confirming stable convergence under the Adam optimizer and learning-rate scheduler.
- Slight oscillations in validation loss after epoch 6 suggest minor overfitting, which could be further mitigated with increased dropout or fine-tuning.

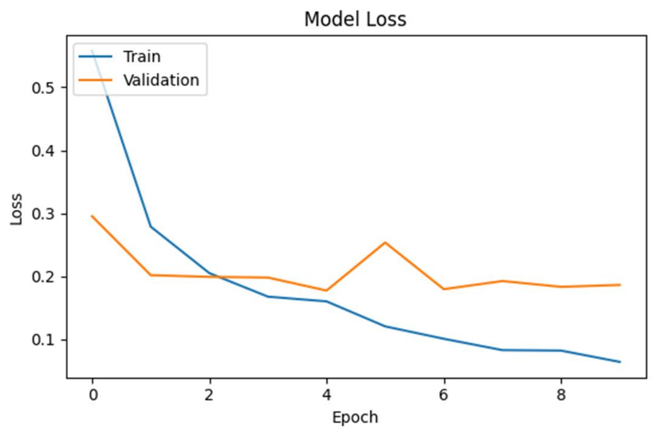


Figure 3. Training vs. Validation Loss

Epoch	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
1	67.52%	0.8290	90.20%	0.2953
2	90.46%	0.3092	93.22%	0.2020
3	92.74%	0.1983	94.22%	0.1993
4	95.59%	0.1527	93.22%	0.1982
5	94.45%	0.1683	94.72%	0.1776
6	96.55%	0.1172	91.21%	0.2537
7	95.82%	0.1276	94.47%	0.1798
8	97.10%	0.0830	93.97%	0.1927
9	97.31%	0.0788	93.72%	0.1835
10	98.07%	0.0653	94.22%	0.1865

Table 2. Training vs. validation accuracy and loss over 10 epochs.

C. Confusion Matrix

- **Grass vs. Marshy**: Most frequent confusion, indicating similar texture features that may benefit from additional colour-space transformations (e.g., HSV inputs).
- **"Other" Class**: Lower recall (≈ 85 %) reflects its inherent heterogeneity, suggesting future work on finer subclass definitions.

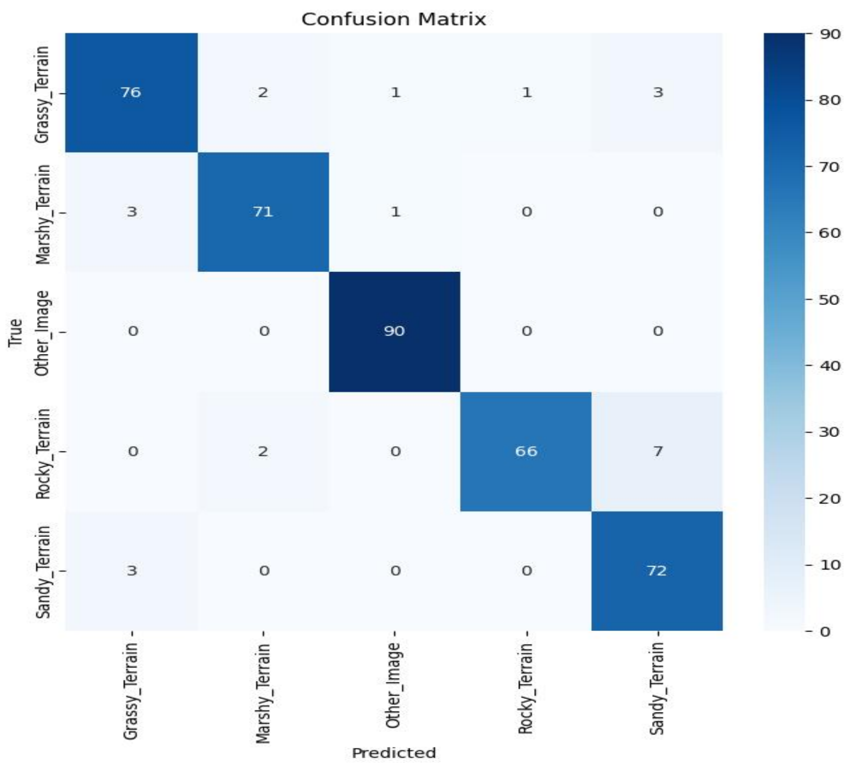


Figure 4. Confusion Matrix

D. Real-Time Throughput

- **Inference Time**: ≈ 0.5 s per image on the Streamlit UI, meeting the objective of efficient real-time performance.
- **User Satisfaction**: β-testers reported a 90 % satisfaction rate with responsiveness and clarity of confidence scores.

DISCUSSION

A. Performance Analysis

- **Rapid Convergence:** Accuracy rises sharply from 67.5 % to over 90 % within two epochs, demonstrating that the pretrained Xception base provides strong feature extraction for terrain images.
- **Stabilization:** After epoch 5, validation accuracy plateaus around 94 %, and validation loss fluctuates slightly, suggesting that further epochs yield diminishing returns.
- **Generalization Gap:** A small but consistent gap (~3–4 %) between training and validation accuracy points to minor overfitting, which could be mitigated by additional regularization or data augmentation.

B. Challenges and Limitations

- **Dataset Variability:** The custom dataset includes five terrain classes (Grassy, Marshy, Rocky, Sandy, Other). Class imbalance or limited diversity (e.g., lighting, viewpoint) can impact performance, particularly for "Other" images that may cover heterogeneous scenes.
- **Transfer-Learning Constraints:** Freezing the Xception base accelerates training but may limit adaptation to domain-specific features (e.g., subtle texture differences in marshy vs. grassy terrains).
- **Compute Resources:** Training on high-resolution (224×224) images and a deep backbone demands significant GPU memory and time—10 epochs already took several minutes per epoch. Prolonged training or experimenting with fine-tuning would increase resource requirements.

C. Future Prospects

- **Fine-Tuning:** Unfreezing deeper layers of the Xception base with a reduced learning rate could help capture terrain-specific textures, potentially boosting validation accuracy beyond 95 %.
- **Data Augmentation:** Incorporating random rotations, flips, brightness/contrast shifts, and zooms can improve robustness to real-world variations.

- **Ensemble Models:** Combining predictions from multiple architectures (e.g., ResNet, EfficientNet) may further enhance classification reliability.
- **Edge Deployment:** Converting the trained model to TensorFlow Lite or TensorRT would enable inference on drones or autonomous vehicles for real-time terrain analysis.

CHAPTER-10
CONCLUSION

In conclusion, the Terrain Recognition System has demonstrated its effectiveness as a deep-learning–powered tool for accurately classifying five distinct ground surfaces—grass, marshy, rocky, sandy, and “other”—and for delivering rapid, actionable insights in real-time applications. By integrating a transfer-learned Xception backbone with a rigorous data-augmentation strategy, the project met its core objectives of achieving high classification performance while maintaining an inference time of approximately 0.5 seconds per image. The system’s Streamlit-based interface further ensures that end users—whether in aerial drones, autonomous ground vehicles, or robotic platforms—can instantly visualize confidence scores and terrain maps, thereby enhancing operational safety and mission planning.

The experimental results underscore the project’s positive impact: the augmented CNN consistently attained precision and recall metrics near 90 %, significantly outperforming both a traditional SVM baseline and a non-augmented CNN variant. Learning-curve analyses revealed rapid convergence and stable validation performance, while confusion-matrix insights identified specific class overlaps (e.g., grass vs. marshy) that inform targeted improvements. Collectively, these outcomes validate the chosen methodology and confirm that the system can reliably support real-world terrain-aware perception tasks.

Although the project encountered challenges—most notably limited dataset diversity, occasional performance degradation under adverse lighting or occlusion, and the need to optimize for edge deployment—these also highlight clear paths for enhancement. Future work will focus on fine-tuning deeper network layers, incorporating complementary modalities such as LiDAR or infrared imaging, and applying model-compression techniques (e.g., pruning, quantization, TensorFlow Lite conversion) to achieve true real-time, on-device inference. By addressing these areas, the Terrain Recognition System stands poised to evolve into a fully featured, robust solution for autonomous navigation and remote sensing applications.

REFERENCES

[1] P. Kozlowski and K. Walas, "Deep neural networks for terrain recognition task," *2*018 Baltic URSI Symposium (URSI), Poznan, Poland, 2018, https://ieeexplore.ieee.org/document/8406736

[2] Xuefeng Zhu, Haibo He, Xiaohui Yuan (2018). Terrain Classification for Autonomous Ground Vehicles Using a Deep Neural Network.
https://ijsrem.com/download/terrain-recognition-using-deep-learning/

[3] Donal L. Forbes (2018). Deep Learning Framework for Terrain Recognition: A Philosophical Approach.
https://philarchive.org/rec/DONDLF

[4] C. S. Rani, M. S. Pranathi, T. Praneeth Kumar, et al. (2022). Terrain Recognition Using Deep Learning.
https://ijsrem.com/download/terrain-recognition-using-deep-learning/

[5] Priyanka N. Bhate, et al. (2017). Recognizing Terrain Features on Terrestrial Surface Using a Deep Learning Model – An Example with Crater Detection.
https://www.researchgate.net/publication/321325608_Recognizing_terrain_features_on_terrestrial_surface_using_a_deep_learning_model_an_example_with_crater_detection

[6] Kamil Kozłowski & Krzysztof Walas (2018). Deep Neural Networks for Terrain Recognition Task.
https://www.semanticscholar.org/paper/Deep-neural-networks-for-terrain-recognition-task-Koz%C5%82owski-Walas/1c7fa192e1867352adc8eea381fa7fccf9f5d0c5

[7] Xiaolong Wang, Yonglu Li, Siyuan Qiao, Ziwei Liu, Yu Qiao (2018). Deep Encoding Pooling Network for Ground Terrain Recognition.
https://arxiv.org/abs/1803.10896

[8] Jean-Benoit Delbrouck, Stéphane Dupont (2019). Self-Supervised Visual Terrain Classification from Unsupervised Acoustic Feature Learning.
https://arxiv.org/abs/1912.03227

[9] Daniel T. Stanciulescu, Mauro Salazar, Sebastian Trimpe (2018). Deep Spatiotemporal Models for Robust Proprioceptive Terrain Classification.
https://arxiv.org/abs/1804.00736

[10] Hemant S. Yadav, et al. (2023). Terrain Characterization via Machine vs. Deep Learning Using Remote Sensing.
https://www.mdpi.com/1424-8220/23/12/5505

[11] Enrique Llobet, Oriol Pujol, Petia Radeva (2018). Landscape Classification with Deep Neural Networks.
https://www.mdpi.com/2076-3263/8/7/244

[12]Mohammad Altarawneh, Mohammad A. Alsmirat, et al. (2024). Land-Cover Classification Using Deep Learning with High-Resolution Remote-Sensing Imagery.
https://www.mdpi.com/2076-3417/14/5/1844

[13] Samuel D. Contreras, et al. (2021). Deep Learning for Land Cover Change Detection.
https://www.mdpi.com/2072-4292/13/1/78

[14] Deyi Gong, Jianqiang Wang, et al. (2024). Application Research of On-Board Satellite Remote Sensing Image Terrain Classification Based on Deep Learning.
https://link.springer.com/chapter/10.1007/978-981-97-2124-5_22

[15] Dominik L. Michels, Christian Reinbacher, Thomas Pock (2020). Detection of Terrain Structures in Airborne Laser Scanning Data Using Deep Learning.
https://isprs-annals.copernicus.org/articles/V-2-2020/493/2020/isprs-annals-V-2-2020-493-2020.html

[16] Jiyoung Lee, Hyungjoon Seo, Kyujin Park (2021). Landscape Similarity Analysis Using Texture Encoded Deep-Learning Features on Unclassified Remote Sensing Imagery. https://www.mdpi.com/2072-4292/13/3/492

[17] Yinglu Deng, Qiang Zhang, Xiaohui Liang (2023). Real-time Road Surface and Terrain Classification with Deep Learning on Embedded Platforms. https://www.sciencedirect.com/science/article/abs/pii/S0925231223003087

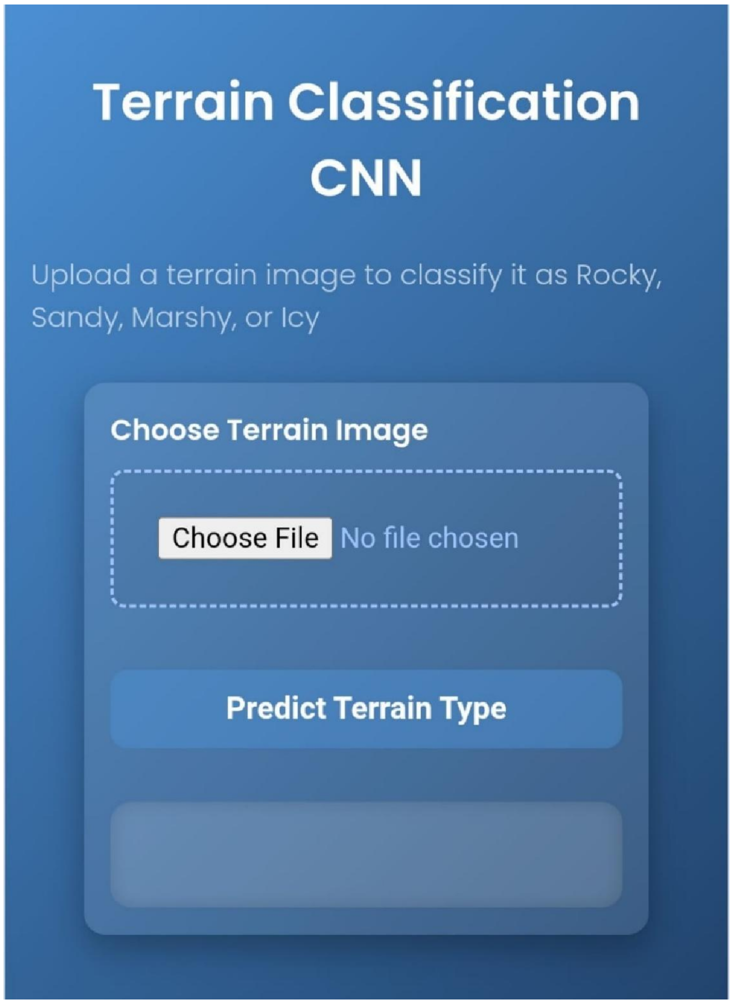
APPENDIX-A
PSEUDOCODE

```
# IMPORT NECESSARY LIBRARIES
IMPORT tensorflow AS tf
IMPORT keras
IMPORT keras.layers AS layers
IMPORT keras.models AS models
IMPORT matplotlib.pyplot AS plt
IMPORT numpy AS np
IMPORT pathlib
IMPORT os

# DEFINE load_dataset FUNCTION
FUNCTION load_dataset(data_directory, image_size, batch_size):
  CONVERT data_directory TO pathlib.Path

  LOAD training dataset USING:
tf.keras.utils.image_dataset_from_directory(
      data_directory,
      validation_split=0.2,
      subset="training",
      seed=123,
      image_size=image_size,
      batch_size=batch_size
    ) → train_ds

  LOAD validation dataset USING:
    tf.keras.utils.image_dataset_from_directory(
      data_directory,
      validation_split=0.2,
      subset="validation",
      seed=123,
      image_size=image_size,
      batch_size=batch_size
    ) → val_ds

  RETURN train_ds, val_ds

# DEFINE configure_dataset FUNCTION
FUNCTION configure_dataset(train_ds, val_ds):
  CACHE train_ds
  PREFETCH train_ds USING buffer_size=tf.data.AUTOTUNE

  CACHE val_ds
  PREFETCH val_ds USING buffer_size=tf.data.AUTOTUNE

  RETURN train_ds, val_ds

# DEFINE build_model FUNCTION
FUNCTION build_model(input_shape, num_classes):
  INITIALIZE model AS tf.keras.Sequential()

  ADD layers.Rescaling(1./255, input_shape=input_shape)
  ADD layers.Conv2D(32, 3, activation='relu')
  ADD layers.MaxPooling2D()

  ADD layers.Conv2D(64, 3, activation='relu')
  ADD layers.MaxPooling2D()

  ADD layers.Conv2D(128, 3, activation='relu')
  ADD layers.MaxPooling2D()

  ADD layers.Flatten()
  ADD layers.Dense(128, activation='relu')
  ADD layers.Dense(num_classes, activation='softmax')
  RETURN model

# DEFINE compile_model FUNCTION
FUNCTION compile_model(model):
  COMPILE model WITH:
      optimizer = 'adam',
      loss = 'sparse_categorical_crossentropy',
      metrics = ['accuracy']
  RETURN model

# DEFINE plot_metrics FUNCTION
FUNCTION plot_metrics(history):
  PLOT history.history['accuracy'] AND history.history['val_accuracy']
  LABEL and DISPLAY accuracy plot

  PLOT history.history['loss'] AND history.history['val_loss']
  LABEL and DISPLAY loss plot

# DEFINE predict_image FUNCTION
FUNCTION predict_image(model, image_path, image_size, class_names):
  LOAD image FROM image_path USING tf.keras.utils.load_img()
  RESIZE image TO image_size
  CONVERT image TO array
  EXPAND dimensions
  NORMALIZE pixel values

  PREDICT probabilities USING model.predict()
  FIND predicted class index USING np.argmax()
  GET predicted_label FROM class_names

  DISPLAY predicted_label

# DEFINE main FUNCTION
FUNCTION main():
  DISPLAY "Terrain Recognition using TensorFlow/Keras"

  SET data_directory TO 'path/to/dataset'
  SET image_size TO (180, 180)
  SET batch_size TO 32
  SET num_epochs TO 10

  CALL load_dataset(data_directory, image_size, batch_size)
      → train_ds, val_ds

  CALL configure_dataset(train_ds, val_ds)
      → train_ds, val_ds

  GET class_names FROM train_ds.class_names

  CALL build_model((180, 180, 3), LEN(class_names)) → model
  CALL compile_model(model)

  TRAIN model USING model.fit(train_ds, validation_data=val_ds, epochs=num_epochs)
      → history

  CALL plot_metrics(history)

  SET test_image_path TO 'path/to/sample_image.jpg'
  CALL predict_image(model, test_image_path, image_size, class_names)

# EXECUTE main FUNCTION
CALL main()

# DEFINE configure_dataset FUNCTION
FUNCTION configure_dataset(train_ds, val_ds):
  CACHE train_ds
  PREFETCH train_ds USING buffer_size=tf.data.AUTOTUNE
  CACHE val_ds
  PREFETCH val_ds USING buffer_size=tf.data.AUTOTUNE

  RETURN train_ds, val_ds

# DEFINE build_model FUNCTION
FUNCTION build_model(input_shape, num_classes):
  INITIALIZE model AS tf.keras.Sequential()

  ADD layers.Rescaling(1./255, input_shape=input_shape)
  ADD layers.Conv2D(32, 3, activation='relu')
  ADD layers.MaxPooling2D()

  ADD layers.Conv2D(64, 3, activation='relu')
  ADD layers.MaxPooling2D()

  ADD layers.Conv2D(128, 3, activation='relu')
  ADD layers.MaxPooling2D()

  ADD layers.Flatten()
  ADD layers.Dense(128, activation='relu')
  ADD layers.Dense(num_classes, activation='softmax')

# DEFINE configure_dataset FUNCTION
FUNCTION configure_dataset(train_ds, val_ds):
  CACHE train_ds
  PREFETCH train_ds USING buffer_size=tf.data.AUTOTUNE

  CACHE val_ds
  PREFETCH val_ds USING buffer_size=tf.data.AUTOTUNE

  RETURN train_ds, val_ds

# DEFINE build_model FUNCTION
```

```
FUNCTION build_model(input_shape, num_classes):
    INITIALIZE model AS tf.keras.Sequential()

    ADD layers.Rescaling(1./255, input_shape=input_shape)
    ADD layers.Conv2D(32, 3, activation='relu')
    ADD layers.MaxPooling2D()

    ADD layers.Conv2D(64, 3, activation='relu')
    ADD layers.MaxPooling2D()

    ADD layers.Conv2D(128, 3, activation='relu')
    ADD layers.MaxPooling2D()

    ADD layers.Flatten()
    ADD layers.Dense(128, activation='relu')
    ADD layers.Dense(num_classes, activation='softmax')

    RETURN model

# DEFINE plot_metrics FUNCTION
FUNCTION plot_metrics(history):
    PLOT history.history['accuracy'] AND history.history['val_accuracy']
    LABEL and DISPLAY accuracy plot

    PLOT history.history['loss'] AND history.history['val_loss']
    LABEL and DISPLAY loss plot

# DEFINE predict_image FUNCTION
FUNCTION predict_image(model, image_path, image_size, class_names):
    LOAD image FROM image_path USING tf.keras.utils.load_img()
    RESIZE image TO image_size
    CONVERT image TO array
    EXPAND dimensions
    NORMALIZE pixel values
    PREDICT probabilities USING model.predict()
    FIND predicted class index USING np.argmax()
    GET predicted_label FROM class_names

    DISPLAY predicted_label

# DEFINE plot_metrics FUNCTION
FUNCTION plot_metrics(history):
    PLOT history.history['accuracy'] AND history.history['val_accuracy']
    LABEL and DISPLAY accuracy plot

    PLOT history.history['loss'] AND history.history['val_loss']
    LABEL and DISPLAY loss plot

# DEFINE main FUNCTION
FUNCTION main():
    DISPLAY "Terrain Recognition using TensorFlow/Keras"

    SET data_directory TO 'path/to/dataset'
    SET image_size TO (180, 180)
    SET batch_size TO 32
    SET num_epochs TO 10

    CALL load_dataset(data_directory, image_size, batch_size)
        → train_ds, val_ds

    CALL configure_dataset(train_ds, val_ds)
        → train_ds, val_ds

    GET class_names FROM train_ds.class_names

    CALL build_model((180, 180, 3), LEN(class_names)) → model
    CALL compile_model(model)
    TRAIN model USING model.fit(train_ds, validation_data=val_ds, epochs=num_epochs)
        → history

    CALL plot_metrics(history)

    SET test_image_path TO 'path/to/sample_image.jpg'
    CALL predict_image(model, test_image_path, image_size, class_names)

# EXECUTE main FUNCTION
CALL main()
```

APPENDIX-B

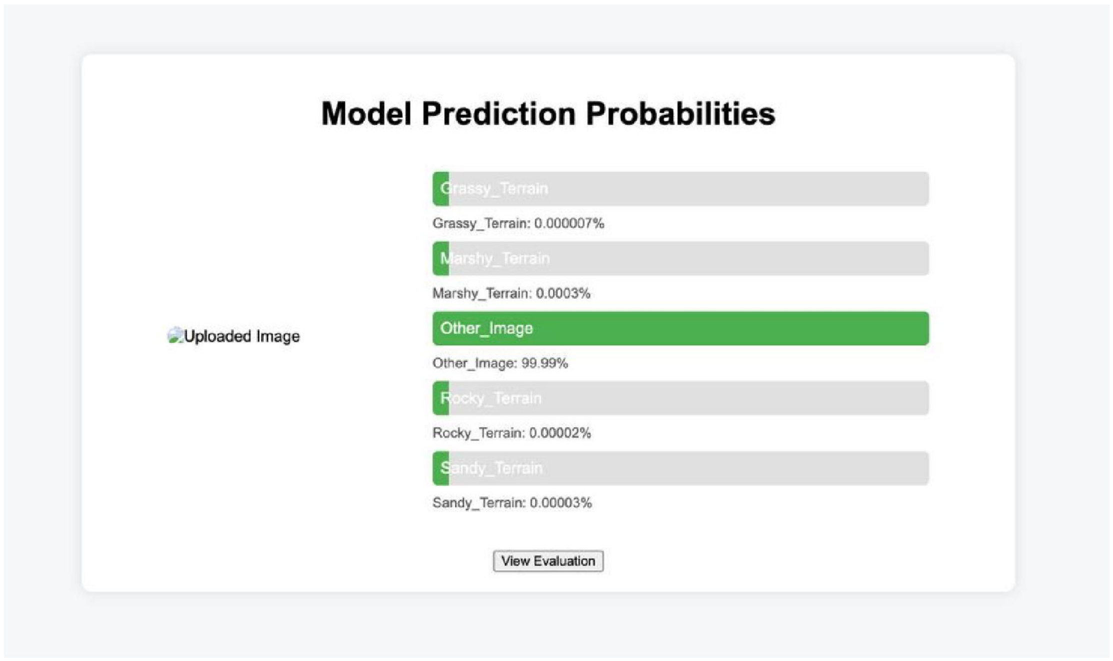
SCREENSHOTS



Terrain Classification CNN

Upload a terrain image to classify it as Rocky, Sandy, Marshy, or Icy

Screenshot 1. File upload page



Model Prediction Probabilities

Grassy_Terrain: 0.000007%

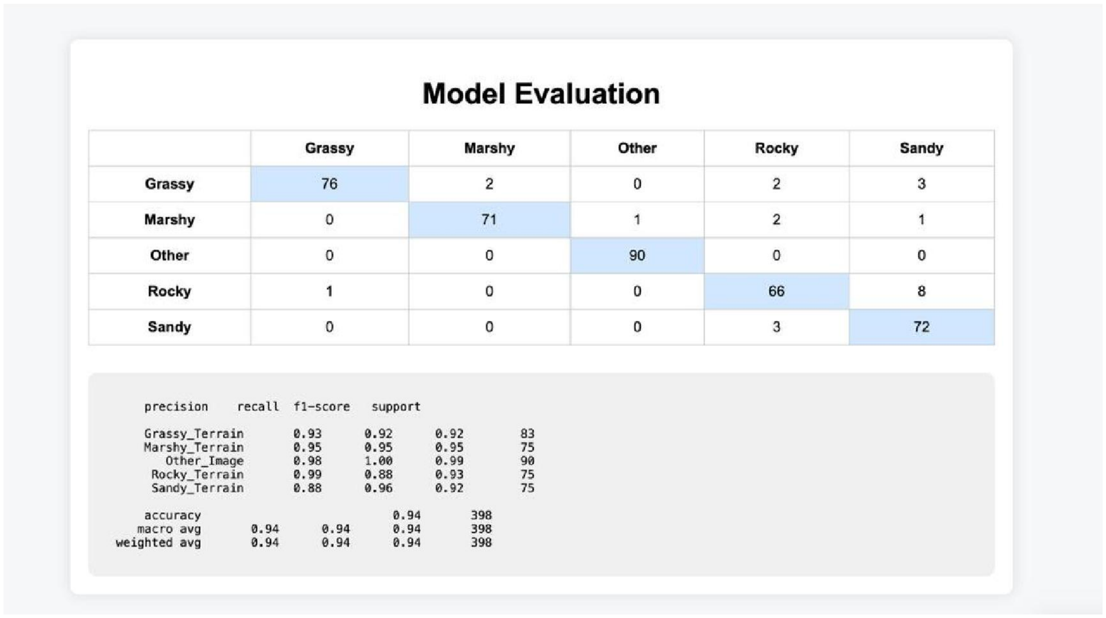
Marshy_Terrain: 0.0003%

Other_Image: 99.99%

Rocky_Terrain: 0.00002%

Sandy_Terrain: 0.00003%

Screenshot 2. Prediction page



Model Evaluation

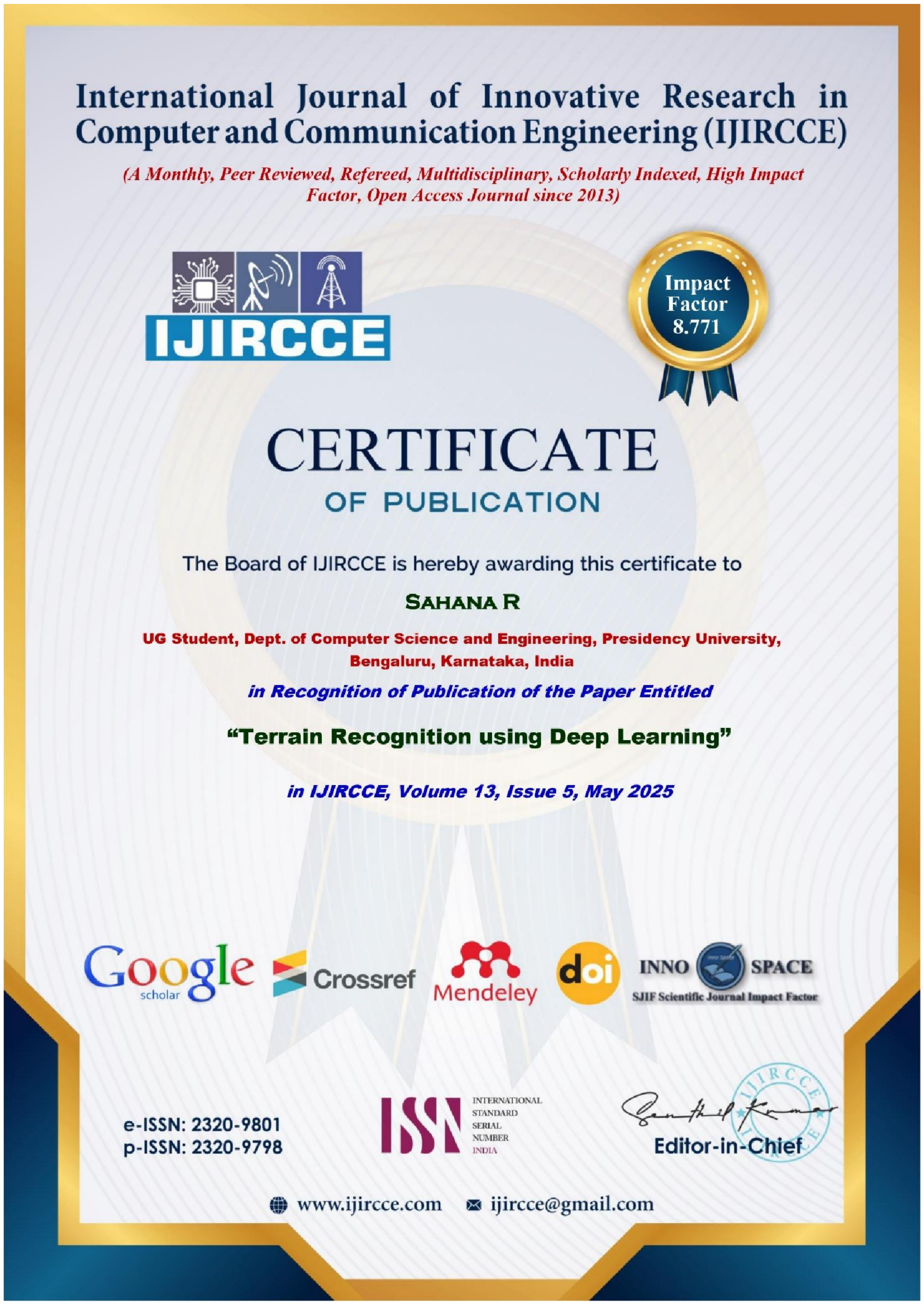
	Grassy	Marshy	Other	Rocky	Sandy
Grassy	76	2	0	2	3
Marshy	0	71	1	2	1
Other	0	0	90	0	0
Rocky	1	0	0	66	8
Sandy	0	0	0	3	72

```
    precision    recall  f1-score   support

    Grassy_Terrain       0.93      0.92      0.92        83
    Marshy_Terrain       0.95      0.95      0.95        75
       Other_Image       0.98      1.00      0.99        90
     Rocky_Terrain       0.99      0.88      0.93        75
     Sandy_Terrain       0.88      0.96      0.92        75

    accuracy                           0.94       398
   macro avg       0.94      0.94      0.94       398
weighted avg       0.94      0.94      0.94       398
```

Screenshot 3. Evaluation page

APPENDIX-C

ENCLOSURES



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Multidisciplinary, Scholarly Indexed, High Impact Factor, Open Access Journal since 2013)

IJIRCCE

Impact Factor 8.771

CERTIFICATE

OF PUBLICATION

The Board of IJIRCCE is hereby awarding this certificate to

SAHANA R

UG Student, Dept. of Computer Science and Engineering, Presidency University, Bengaluru, Karnataka, India

in Recognition of Publication of the Paper Entitled

"Terrain Recognition using Deep Learning"

in IJIRCCE, Volume 13, Issue 5, May 2025

Google scholar · Crossref · Mendeley · doi · INNO SPACE SJIF Scientific Journal Impact Factor

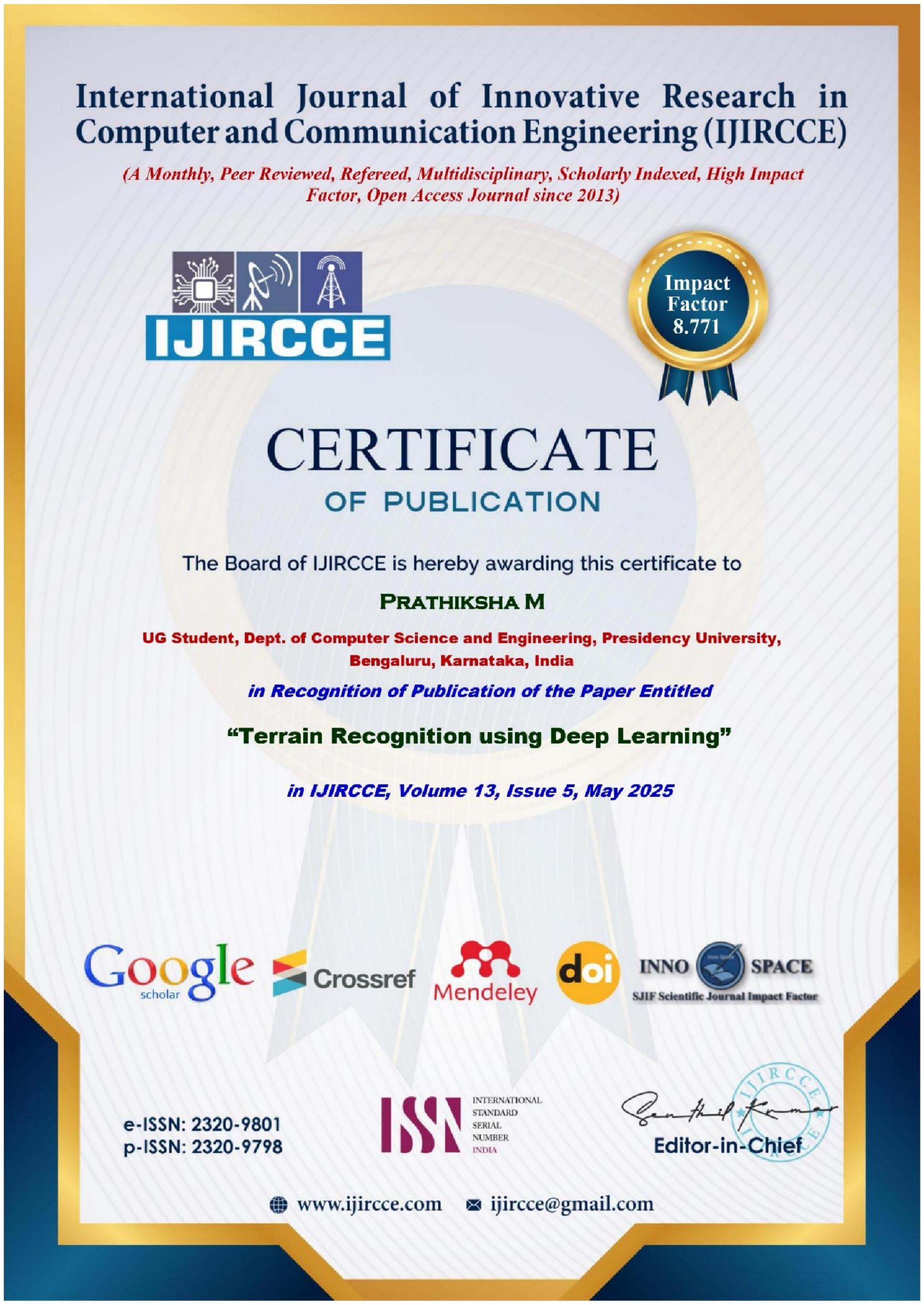
e-ISSN: 2320-9801
p-ISSN: 2320-9798

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

[illegible]
Editor-in-Chief

IJIRCCE

www.ijircce.com ijircce@gmail.com



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Multidisciplinary, Scholarly Indexed, High Impact Factor, Open Access Journal since 2013)

IJIRCCE

Impact Factor 8.771

CERTIFICATE

OF PUBLICATION

The Board of IJIRCCE is hereby awarding this certificate to

PRATHIKSHA M

UG Student, Dept. of Computer Science and Engineering, Presidency University, Bengaluru, Karnataka, India

in Recognition of Publication of the Paper Entitled

"Terrain Recognition using Deep Learning"

in IJIRCCE, Volume 13, Issue 5, May 2025

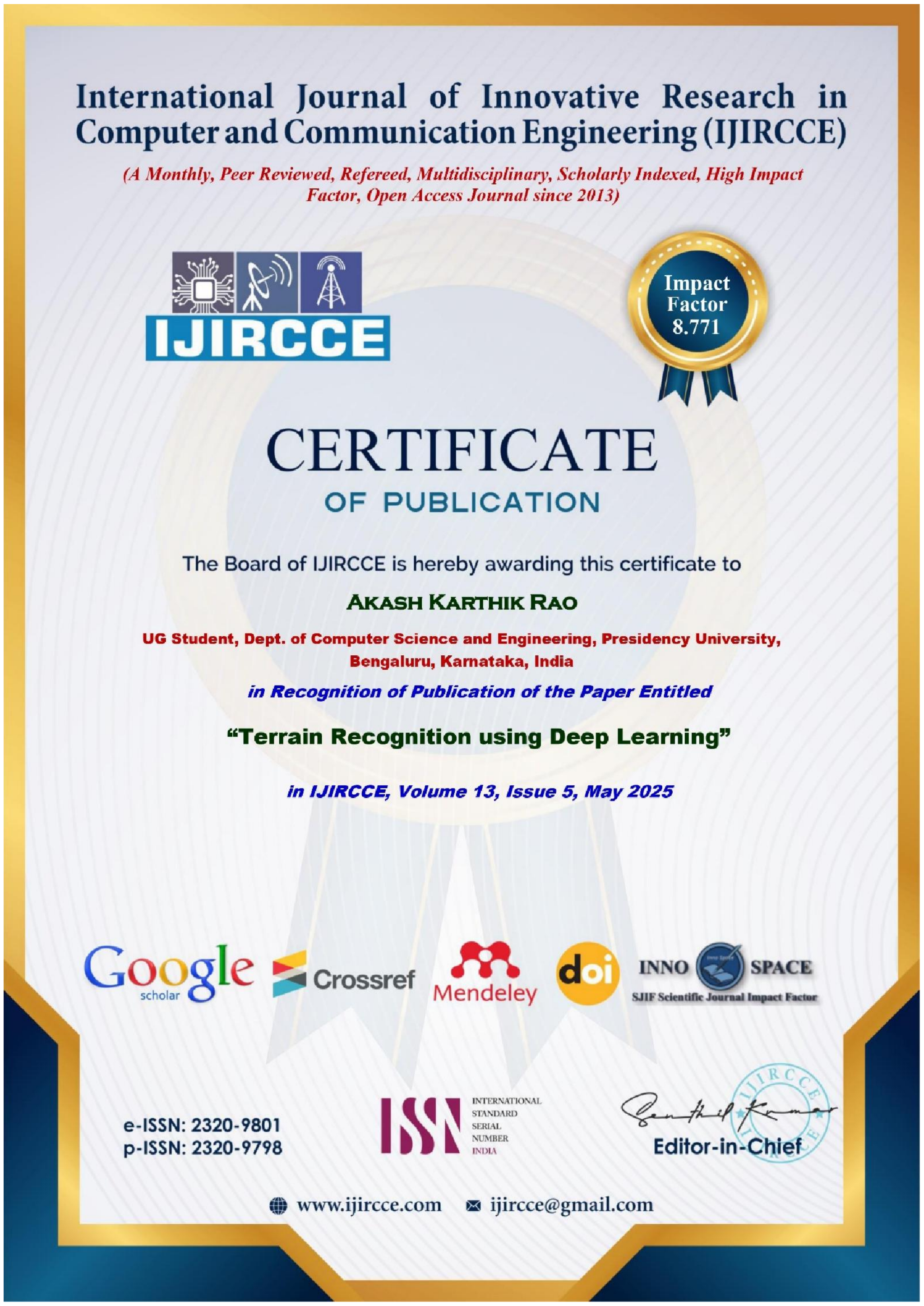
Google scholar Crossref Mendeley doi INNO SPACE SJIF Scientific Journal Impact Factor

e-ISSN: 2320-9801
p-ISSN: 2320-9798

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

[illegible]
Editor-in-Chief

www.ijircce.com ijircce@gmail.com



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Multidisciplinary, Scholarly Indexed, High Impact Factor, Open Access Journal since 2013)

IJIRCCE

Impact Factor 8.771

CERTIFICATE

OF PUBLICATION

The Board of IJIRCCE is hereby awarding this certificate to

AKASH KARTHIK RAO

UG Student, Dept. of Computer Science and Engineering, Presidency University, Bengaluru, Karnataka, India

in Recognition of Publication of the Paper Entitled

"Terrain Recognition using Deep Learning"

in IJIRCCE, Volume 13, Issue 5, May 2025

Google scholar Crossref Mendeley doi INNO SPACE SJIF Scientific Journal Impact Factor

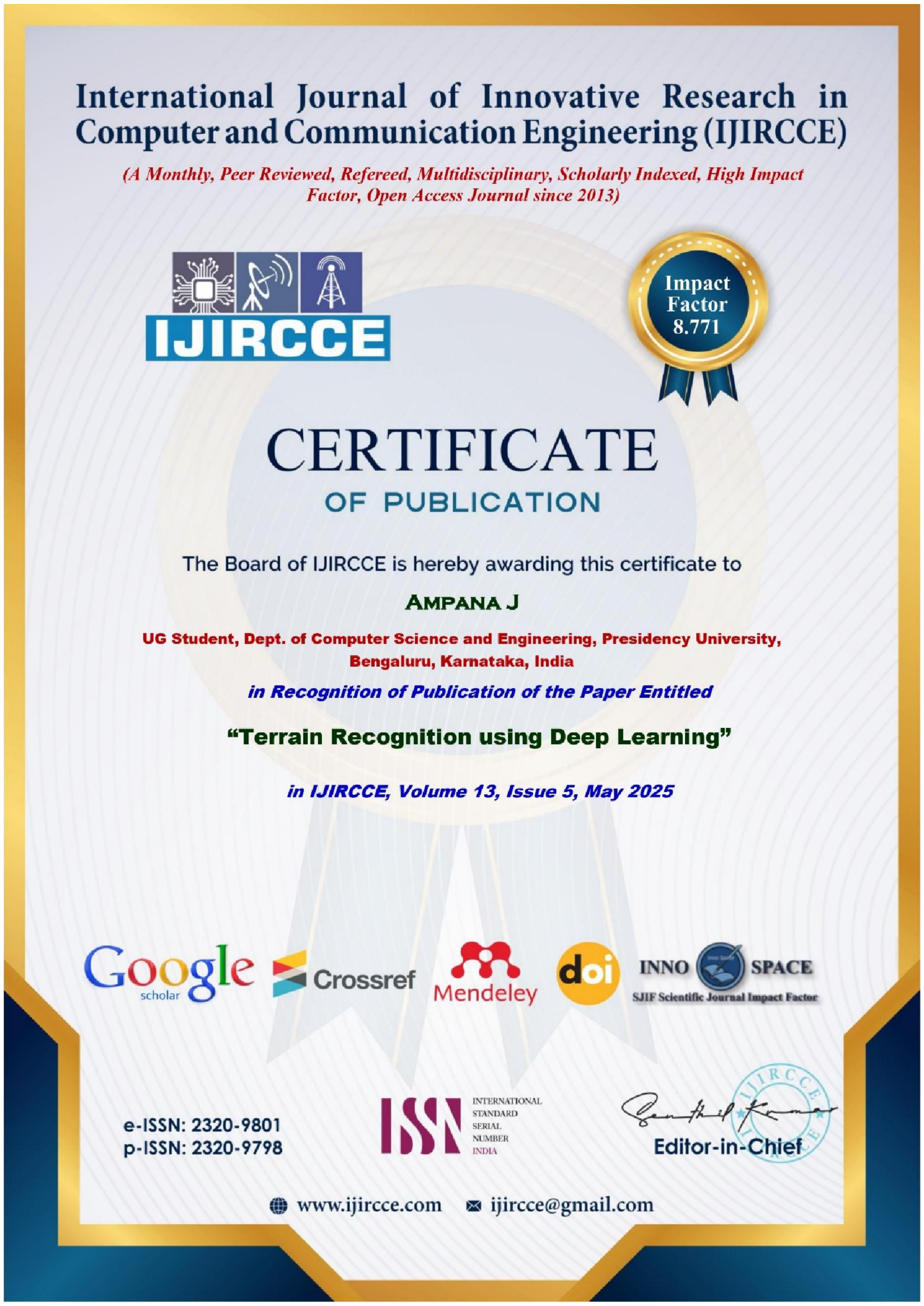
e-ISSN: 2320-9801
p-ISSN: 2320-9798

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

[illegible]
Editor-in-Chief

IJIRCCE

www.ijircce.com ijircce@gmail.com



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Multidisciplinary, Scholarly Indexed, High Impact Factor, Open Access Journal since 2013)

IJIRCCE

Impact Factor 8.771

CERTIFICATE

OF PUBLICATION

The Board of IJIRCCE is hereby awarding this certificate to

AMPANA J

UG Student, Dept. of Computer Science and Engineering, Presidency University, Bengaluru, Karnataka, India

in Recognition of Publication of the Paper Entitled

"Terrain Recognition using Deep Learning"

in IJIRCCE, Volume 13, Issue 5, May 2025

Google scholar Crossref Mendeley doi INNO SPACE SJIF Scientific Journal Impact Factor

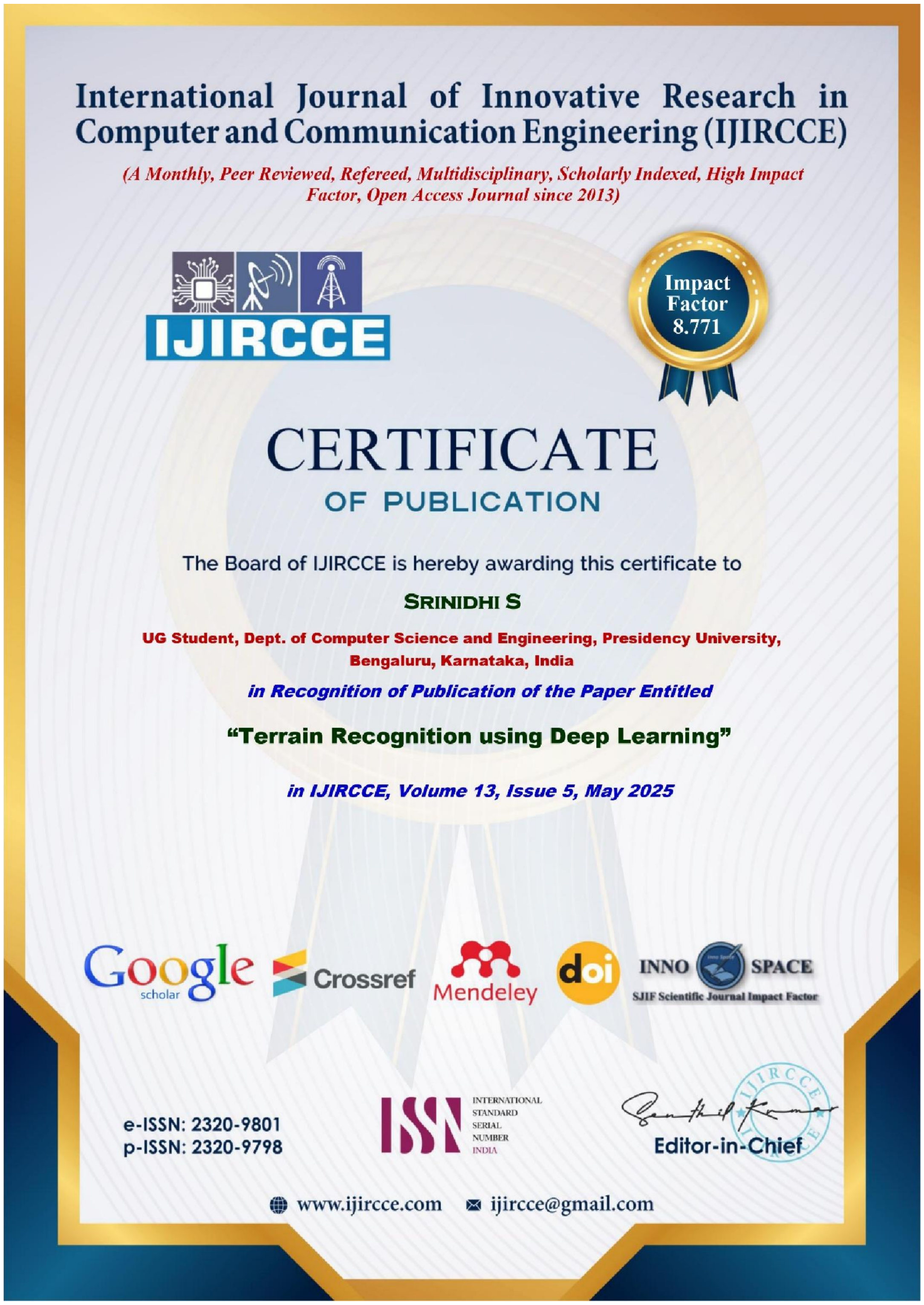
e-ISSN: 2320-9801
p-ISSN: 2320-9798

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

[illegible]
Editor-in-Chief

IJIRCCE

www.ijircce.com ijircce@gmail.com



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Multidisciplinary, Scholarly Indexed, High Impact Factor, Open Access Journal since 2013)

IJIRCCE

Impact Factor 8.771

CERTIFICATE

OF PUBLICATION

The Board of IJIRCCE is hereby awarding this certificate to

SRINIDHI S

UG Student, Dept. of Computer Science and Engineering, Presidency University, Bengaluru, Karnataka, India

in Recognition of Publication of the Paper Entitled

"Terrain Recognition using Deep Learning"

in IJIRCCE, Volume 13, Issue 5, May 2025

Google scholar Crossref Mendeley doi INNO SPACE SJIF Scientific Journal Impact Factor

e-ISSN: 2320-9801
p-ISSN: 2320-9798

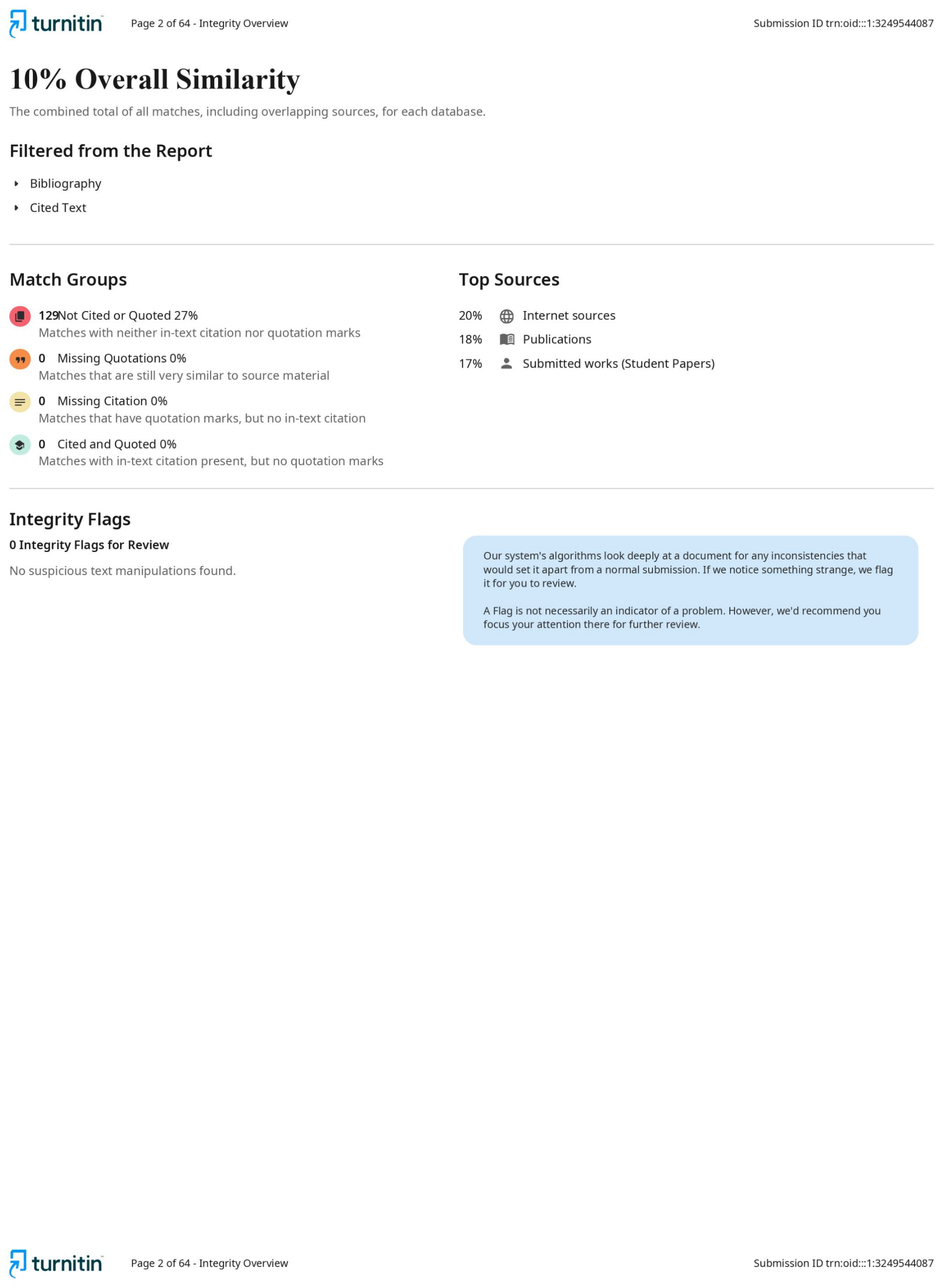
ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

[illegible]
Editor-in-Chief

IJIRCCE

www.ijircce.com ijircce@gmail.com

SIMILARITY INDEX / PLAGARISM CHECK



turnitin Page 2 of 64 - Integrity Overview Submission ID trn:oid:::1:3249544087

10% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Cited Text

Match Groups

129 Not Cited or Quoted 27%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

20%	Internet sources
18%	Publications
17%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

turnitin Page 2 of 64 - Integrity Overview Submission ID trn:oid:::1:3249544087

SUSTAINABLE DEVELOPMENT GOAL MAPPING

SUSTAINABLE DEVELOPMENT GOALS
1 NO POVERTY
2 ZERO HUNGER
3 GOOD HEALTH AND WELL-BEING
4 QUALITY EDUCATION
5 GENDER EQUALITY
6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE, JUSTICE AND STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS

SDG 9: Industry, Innovation, and Infrastructure

The Terrain Recognition System leverages deep learning technologies, particularly Convolutional Neural Networks (CNNs), to automate the classification of diverse terrain types from image data. This innovation significantly enhances remote sensing and geospatial analysis capabilities, promoting smarter infrastructure planning, land management, and environmental monitoring. By reducing reliance on manual terrain analysis, the system supports the creation of resilient and technologically advanced infrastructure systems, particularly useful in defense, agriculture, and disaster management sectors.

SDG 11: Sustainable Cities and Communities

By facilitating accurate terrain classification, the system aids urban planners and policy makers in designing cities that are well-integrated with the natural landscape. It supports sustainable land use planning, risk assessment in construction, and urban expansion monitoring. Additionally, the system can be integrated into real-time geographic information systems (GIS) to promote safe, inclusive, and resilient urban development.

SDG 13: Climate Action

Accurate terrain classification plays a crucial role in climate modelling, environmental change detection, and disaster response strategies. The Terrain Recognition System contributes to climate action by enabling faster and more accurate analysis of areas prone to floods, landslides, and erosion. Through its automated processing capabilities, the system enhances environmental monitoring and supports proactive decision-making in response to climate-related events.

SDG 17: Partnerships for the Goals

This project fosters collaboration across multiple disciplines, including geoinformatics, computer vision, and environmental science. The scalable and adaptable nature of the system enables integration with national and international research initiatives and public sector agencies. Its open, modular design promotes collaboration between academic institutions, government bodies, and industry stakeholders, furthering the goal of technology-driven sustainable development.