A PROJECT REPORT

on

"Geospatial Surface Recognition System"

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN INFORMATION TECHNOLOGY

BY

MAHAK MAHAWAR	21051143
ANUSHKA SINGH	21052144
ADITYA RANJAN	21052301
PIYUSH RAJ	21052341
PRIYANSHU MIDHA	21052344
SAMEER SINHA	21052355

UNDER THE GUIDANCE OF
PROF. SANTWANA SAGNIKA

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
Deemed to be University U/S 3 of the UGC Act, 1956

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAR, ODISHA - 751024

KIIT Deemed to be University

School of Computer Engineering

Bhubaneswar, ODISHA 751024

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

Deemed to be University U/S 3 of the UGC Act, 1956

This is certify that the project entitled

"Geospatial Surface Recognition System "

submitted by

MAHAK MAHAWAR	21051143
ANUSHKA SINGH	21052144
ADITYA RANJAN	21052301
PIYUSH RAJ	21052341
PRIYANSHU MIDHA	21052344
SAMEER SINHA	21052355

is a record of Bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering OR Information Technology) at KIIT Deemed to be university, Bhubaneswar. This work is done during year 2024-2025, under our guidance.

Date: 26 Nov 2024

Prof. Santwana Sagnika
Project Guide

Acknowledgements

We are profoundly grateful to **Prof. Santwana Sagnika** of **School of Computer Science** for her expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

MAHAK MAHAWAR
ANUSHKA SINGH
ADITYA RANJAN
PIYUSH RAJ
PRIYANSHU MIDHA
SAMEER SINHA

ABSTRACT

This project implements a deep learning model for geospatial surface recognition, utilizing a convolutional neural network (CNN) built with TensorFlow. The dataset, sourced from Kaggle, contains images of various terrains, allowing the model to learn distinctive features and classify images into different terrain types. After data preprocessing, the CNN is trained using multiple convolutional layers to capture spatial hierarchies in the data. The model is compiled with the Adam optimizer and trained with sparse categorical cross-entropy loss, achieving accurate geospatial surface classification. This project demonstrates the effectiveness of deep learning in image recognition tasks, with potential applications in autonomous navigation and environmental monitoring.

Table Of Contents

LIST OF IMAGES

Chapter 1

Introduction

Significant progress has been made in the field of computer vision in recent years, especially in the area of image classification. Convolutional Neural Networks (CNNs) have become an effective tool for this purpose, allowing machines to accurately detect and classify images. The goal of this project is to use a dataset of terrain photos to develop and implement a CNN model for landscape recognition.

This project's main goal is to create a reliable picture classification system that can recognize various landscape types from photos. There are several types of terrain in the dataset utilized for this research, including grassy, rocky, and swampy regions. Our goal is to develop a system that can automatically categorize new, unseen terrain photos into one of the specified groups by using this dataset to train a CNN model.
The CNN model's development, training, and evaluation are all included in this study. Data preparation, model architecture, training methods, and evaluation metrics are all covered. It also offers a useful illustration of how to forecast a fresh image's class using the trained model.

Through this project, we show how deep learning can be used to solve practical image categorization issues and offer a thorough implementation guide for such systems. The methods and insights discussed in this study can be used in a number of other fields, including geological surveys, agriculture, and environmental monitoring, where picture classification is essential.

Chapter 2

Basic Concepts/ Literature Review

This section contains the basic concepts about the related tools and techniques used in this project. For research work, present the literature review in this section.

2.1 Convolutional Neural Networks (CNNs)

A family of profound learning models called Convolutional Neural Systems (CNNs) was made particularly for dealing with grid-like input, counting pictures. Convolutional, pooling, and completely connected layers are among the a few layers that make up CNNs. Whereas pooling layers recoil the spatial measurements of the highlight maps, convolutional layers include channels to the input picture in arrange to extricate highlights. The final sorting is carried out by completely connected layers utilizing the highlights that were extricated.
In a assortment of picture acknowledgment assignments, such as question recognizable proof, picture division, and picture classification, CNNs have illustrated surprising viability. They are an viable apparatus for computer vision applications since of their capacity to consequently learn and extricate characteristics from natural picture information.

2.2 TensorFlow and Keras

Google created the open-source machine learning framework TensorFlow. For creating and implementing machine learning models, it offers a complete ecosystem. Building and training deep learning models is made easier with TensorFlow's integration of Keras, a high-level neural networks API.
Numerous tools are available for data preprocessing, model construction, training, and evaluation with TensorFlow and Keras. They are well-liked for both industrial and research applications due to their adaptability and simplicity of usage.

2.3 Image Dataset Preparation

Preprocessing, labelling, and data collecting are some of the procedures involved in preparing an image dataset. The project's dataset includes pictures of many geospatial surface types, including grassy, rocky, and swampy regions. Every picture has a label indicating which class it belongs to. Normalizing pixel values, scaling photographs to a uniform size, and dividing the dataset into training and validation sets are examples of data preprocessing procedures. These procedures guarantee that the data is formatted appropriately for CNN model training.

2.4 Data Augmentation

The method of including arbitrary changes to the photographs in arrange to misleadingly extend the estimate of the preparing dataset is known as information expansion/data augmentation. Trimming, flipping, scaling, and revolution are common increase strategies. By uncovering the show to a more prominent extend of preparing occurrences, information increase improves the model's capacity for speculation. This brings down the chance of overfitting, which is particularly supportive for little datasets.

2.5 Model Training and Evaluation

When a CNN model is being trained the pre-processed images are being fed into the model and the parameters are changed to minimize the loss function the training dataset is used for training of the model while the validation dataset is used for assessing that how well the model performs the models performance is evaluated using metrics like accuracy precision recall and f1-score these metrics show how well the model can categorize the photos and point out areas that need work

2.6 Model Optimization

To enhance performance model optimization entails adjusting the models architecture and hyperparameters in order to avoid overfitting and improve the models capacity for generalization strategies including regularization dropout and learning rate scheduling are employed in order to deter the model from fitting the noise in training data regularization techniques like as l1 and l2 regularization apply penalties to the loss function in order to help the model acquire more and more robust features dropout randomly changes a portion of input units to zero during the training process in order to maximize the models convergence during training the learning rate scheduling modifies the learning rate

2.7 Model Deployment

Once the model is trained and evaluated, it can be deployed for real-world applications. Model deployment involves saving the trained model and integrating it into a software system or application.
The saved model can be loaded and used to make predictions on new, unseen images. This involves preprocessing the input image, feeding it into the model, and interpreting the model's output to determine the predicted class.

2.8 Literature Review

Several research works have explored the application of CNNs for image classification tasks. Krizhevsky et al. (2012) introduced the Alex Net architecture, which achieved state of art performance on the ImageNet dataset. Since then, numerous advancements have been made in CNN architectures which includes VGGNet (Simonyan & Zisserman, 2014), ResNet (He et al., 2016), and Inception (Szegedy et al., 2015).
These architectures have been successfully applied to various domains, including medical imaging, autonomous driving, and environmental monitoring. The success of CNNs in these applications highlights their potential for solving complex image classification problems.

Chapter 3

Problem Statement / Requirement Specifications

Problem Statement

The problem addressed in this project is the classification of geospatial surface or terrain images into predefined categories such as marshy, rocky, grassy, etc. Accurate terrain classification is crucial for various applications, including environmental monitoring, agricultural planning, and geological surveys. Traditional methods of terrain classification often rely on manual inspection, which is time-consuming and prone to human error. Therefore, there is a need for an automated system that can accurately classify terrain images based on visual features.

Requirement Specifications

The following are the requirement specifications for the terrain image classification system, presented in the IEEE format for Software Requirement Specifications (SRS):

Functional Requirements

→ *FR1:* The system shall be able to load and preprocess terrain images.
→ *FR2:* The system shall be able to train a Convolutional Neural Network (CNN) model on the dataset of several labelled terrain images.
→ *FR3:* The system should be able to evaluate the trained model using a validation dataset.
→ *FR4:* The working system should be able to save the trained model for future use.
→ *FR5:* The working system shall be able to load a pre-trained model and make predictions on new and unseen terrain images.
→ *FR6:* The system shall provide a user interface for inputting image paths and displaying prediction results.

Non-Functional Requirements

→ *NFR 1:* The system shall achieve an accuracy of a minimum 85% on the validation dataset.
→ *NFR 2:* The system shall be able to process and classify an image within 5 seconds.
→ *NFR 3:* The system shall be compatible with standard image formats (e.g., JPEG, PNG).
→ *NFR 4:* The system should be user-friendly and easy to navigate throughout.

→ *NFR 5:* The system shall be scalable to handle large datasets.

3.1 Project Planning

Project planning involves outlining the steps and requirements needed to execute the project development. The following list details the key steps and features to be developed:

Data Collection and Preparation

- Download and extract the terrain image dataset.
- Preprocess the images (resizing, normalization).
- Split the dataset into training and validation sets.

Model Building

- Define the architecture of the CNN model.
- Compile the model with appropriate loss functions and optimizers.

Model Training

- Train the model on the training dataset.
- Monitor the training process and adjust hyperparameters as needed.

Model Evaluation

- Evaluate the model's performance on the validation dataset.
- Calculate and report evaluation metrics (accuracy, precision, recall, F1-score).
- Identify Overfitting or Underfitting: Compare the model's performance on the training and validation datasets with the help of graphs

Model Optimization

- Fine-tune the model to improve performance.
- Implement techniques such as data augmentation, regularization, and dropout.

Model Deployment

- Save the trained model for future use.
- Develop a user interface for inputting image paths and displaying prediction results.

3.2 Project Analysis

After collecting the requirements and conceptualizing the problem statement, it is essential to analyse the project for any ambiguities or mistakes. This involves:

Requirement Validation

- Ensure that all functional and non-functional requirements are clearly defined and feasible.

- Validate the requirements with stakeholders to ensure they meet the project's objectives.

Feasibility Analysis

- Assess the technical feasibility of the project, including the availability of necessary tools and resources.
- Evaluate the economic feasibility, considering the costs associated with data collection, model training, and deployment.

Risk Analysis

- Identify potential risks and challenges, such as data quality issues, model overfitting, and hardware limitations.
- Develop contingency plans to mitigate these risks.

3.3 System Design

3.3.1 Design Constraints

The following design constraints must be considered while developing the terrain image classification system:

Working Environment

- *Software:* TensorFlow, Keras, Python, Jupyter Notebook.
- *Hardware:* A computer with a GPU for accelerated model training.
- *Experimental Setup:* Access to a large dataset of labelled terrain images.

Performance Constraints

- The system must be able to process and classify images within a reasonable time frame.
- The model must achieve high accuracy on the validation dataset.

3.3.2 System Architecture / Block Diagram

The system architecture for the terrain image classification system can be represented as follows:

Data Preparation Module

- *Input:* Raw terrain images.
- *Process:* Image resizing, normalization, data augmentation.
- *Output:* Pre-processed images ready for training.

Model Training Module

- *Input:* Pre-processed images.
- *Process:* Model definition, compilation, training.
- *Output:* Trained CNN model.

Model Evaluation Module

- *Input:* Trained model, validation dataset.
- *Process:* Model evaluation, calculation of metrics.
- *Output:* Evaluation results (accuracy, precision, recall, F1-score, graphs).

Model Deployment Module

- *Input:* Trained model, new terrain images.
- *Process:* Model loading, image preprocessing, prediction.
- *Output:* Predicted class of the input image.

User Interface Module

- *Input:* Image path from the user.
- *Process:* Image loading, preprocessing, prediction.
- *Output:* Display of prediction results.

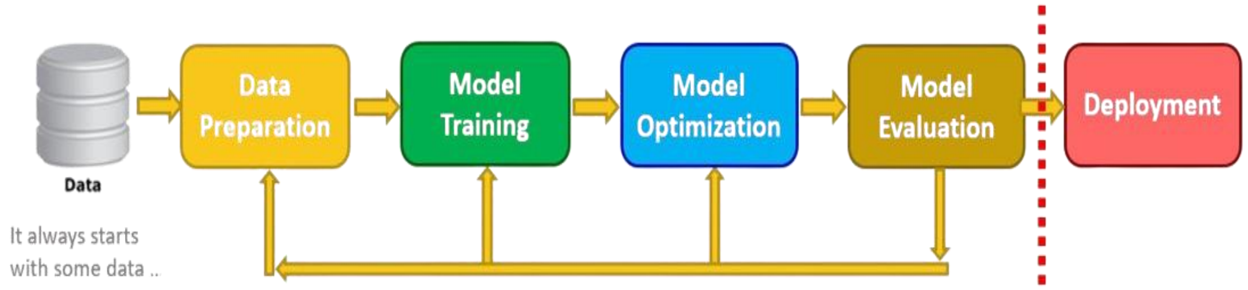


Fig 1: Block Diagram

This block diagram illustrates the code flow of data and processes within the terrain image classification system. Each module plays a crucial role in the overall functionality of the system, from data preparation to model deployment and user interaction.

Chapter 4

Implementation

In this section, we present the implementation details of the project development, including the methodology, testing or verification plan, result analysis, and quality assurance.

4.1 Methodology OR Proposal

This subsection contains the methods used to complete the project, the algorithms developed, and the steps adopted for completing the project work.

Steps Adopted:

Data Collection and Preparation

- *Download Dataset:* The terrain recognition dataset was downloaded from Kaggle.
- *Extract Dataset:* The dataset was extracted from the ZIP file.
- *Count Images:* The number of images in the dataset was counted.
- *Display Sample Image:* A sample image was displayed to verify the data.

Data Preprocessing

- *Set Parameters:* Batch size and image dimensions were defined.
- *Create Datasets:* The dataset was split into training and validation sets.
- *Normalize Dataset:* The pixel values of the images were normalized to the range [0, 1].
- *Optimize Dataset Performance:* The datasets were cached and prefetched to improve performance during training.

Model Building

- *Define Model Architecture:* A Convolutional Neural Network (CNN) model was built using the tf. keras.Sequential API. The model consisted of convolutional layers, Leaky ReLU activation, max pooling layers, and dense layers.
- *Compile Model:* The model was compiled with the Adam optimizer, Sparse Categorical Crossentropy loss, and accuracy metric.

Model Training

- *Train Model:* The model was trained with the help of training dataset and validated on the validation dataset for 20 epochs.
- *Model Evaluation and Saving*
- *List Dataset Files:* A dataset of file paths was created and shuffled.
- *Get Class Names:* The class names were obtained from the dataset directory.
- *Split Dataset:* The dataset was split into two i.e training and validation sets.
- *Define Data Processing Functions:* Functions to get labels, decode images, and process file paths were defined.
- *Map Dataset:* The dataset was mapped to process file paths and decode images.
- *Optimize Dataset Performance:* The datasets were cached, shuffled, batched, and prefetched to improve performance during training.
- *Save Model:* The trained model was saved to a file.

Model Prediction

- *Load Model:* The saved model was loaded using joblib.
- *Define Class Names:* The class names used during training were defined.
- *Predict Image Class:* A function to load, preprocess, and predict the class of an image was defined.
- *Take Input and Predict:* The user was prompted to enter the image path, and the function was called to predict the class of the input image.

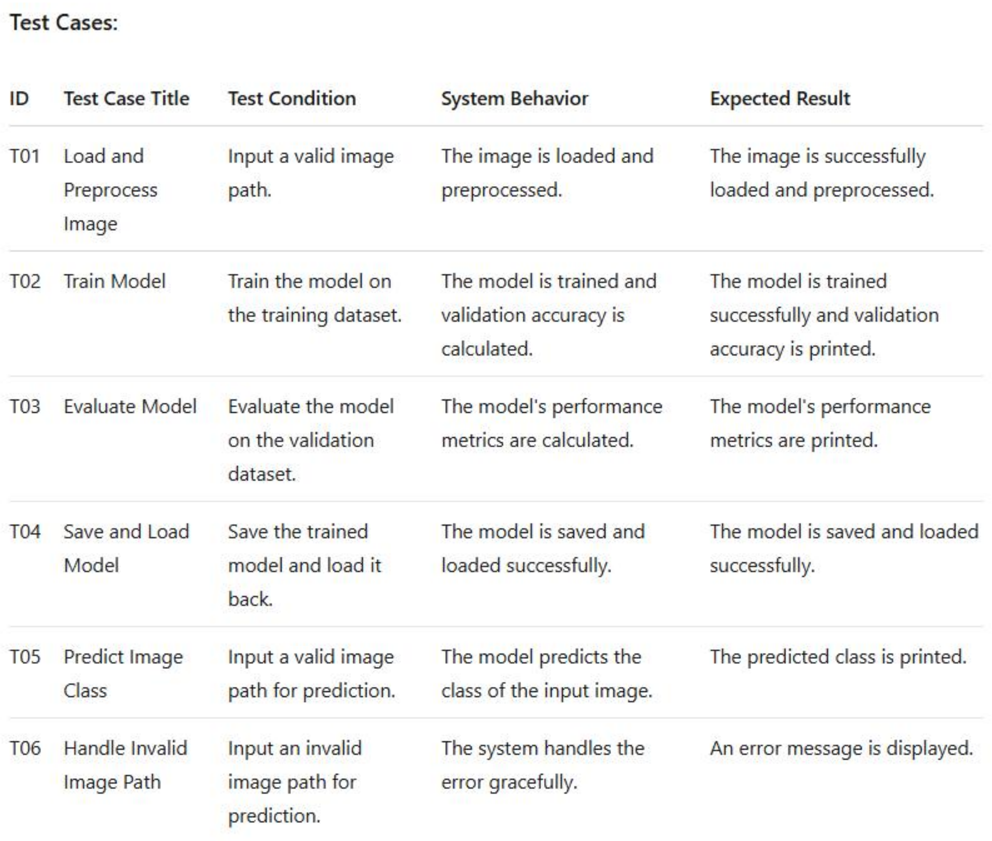
Algorithms Used:

1. *Convolutional Neural Networks (CNNs):* Used for image classification.
2. *Data Augmentation:* Used to artificially increase the size of the training dataset.
3. *Model Optimization:* Techniques such as regularization, dropout, and learning rate scheduling were used to improve the model's performance.

4.2 Testing OR Verification Plan

Once a project is finished, it's essential to evaluate its success. This involves a verification process, often called testing. For instance, in software development, specific test cases are designed and executed to check if the software functions as intended and meets the desired outcomes.

Test Cases:



Test Cases:

ID	Test Case Title	Test Condition	System Behavior	Expected Result
T01	Load and Preprocess Image	Input a valid image path.	The image is loaded and preprocessed.	The image is successfully loaded and preprocessed.
T02	Train Model	Train the model on the training dataset.	The model is trained and validation accuracy is calculated.	The model is trained successfully and validation accuracy is printed.
T03	Evaluate Model	Evaluate the model on the validation dataset.	The model's performance metrics are calculated.	The model's performance metrics are printed.
T04	Save and Load Model	Save the trained model and load it back.	The model is saved and loaded successfully.	The model is saved and loaded successfully.
T05	Predict Image Class	Input a valid image path for prediction.	The model predicts the class of the input image.	The predicted class is printed.
T06	Handle Invalid Image Path	Input an invalid image path for prediction.	The system handles the error gracefully.	An error message is displayed.

Fig 2: Test cases

4.3 Result Analysis OR Screenshots

In this subsection, the output of the experiment or study in terms of some graphs, plots, and screenshots is presented to showcase the proof of the output

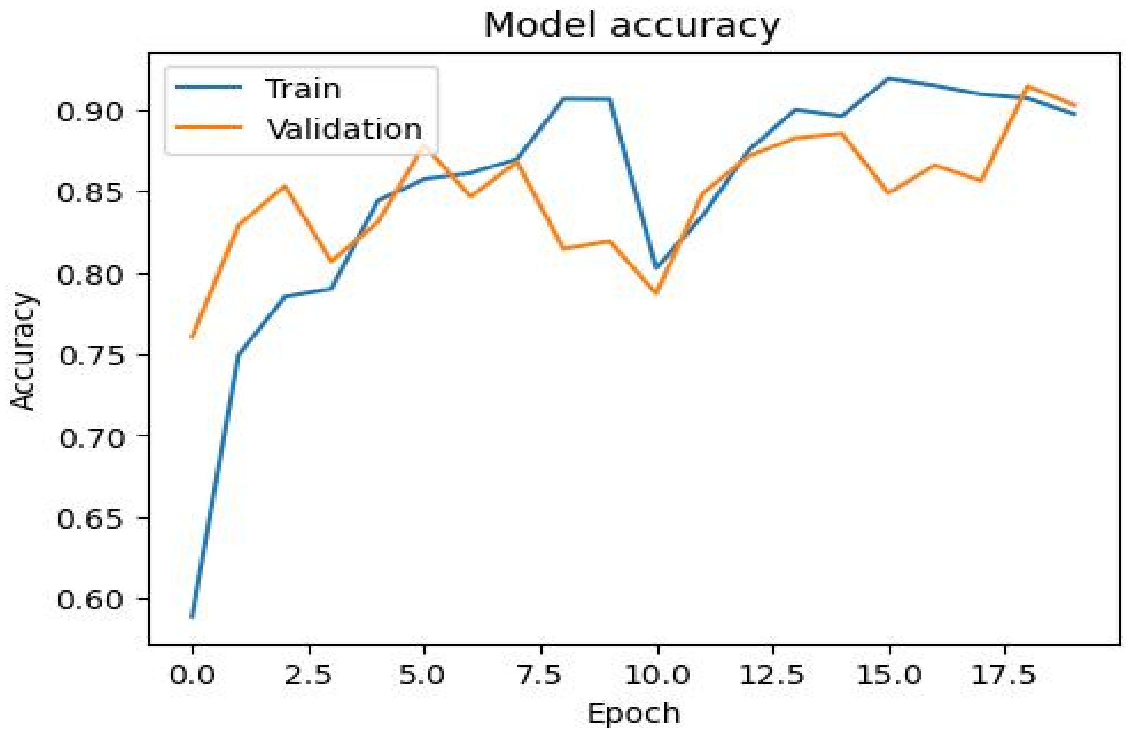


Fig 3: Training and validation Accuracy graph

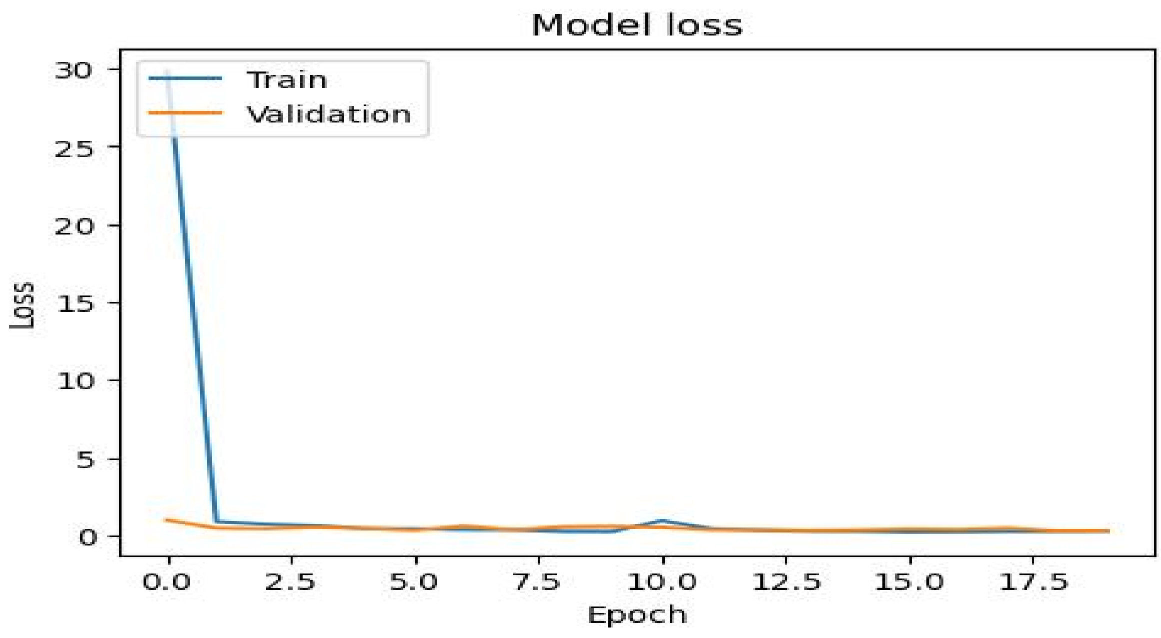


Fig 4: Training and validation loss graph

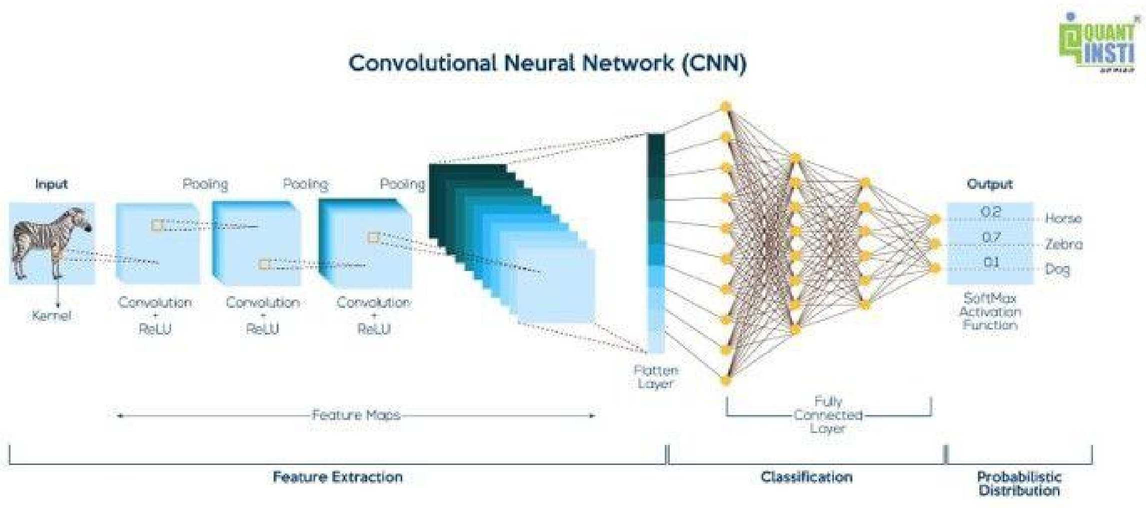
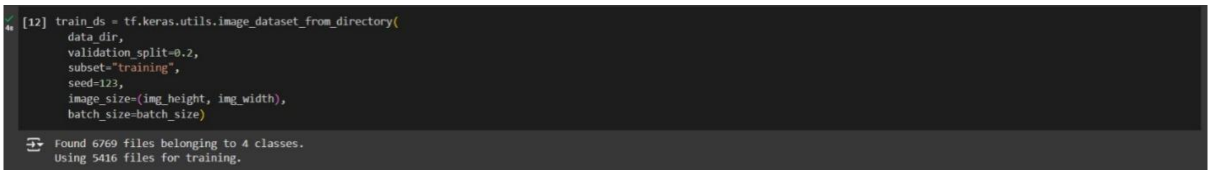


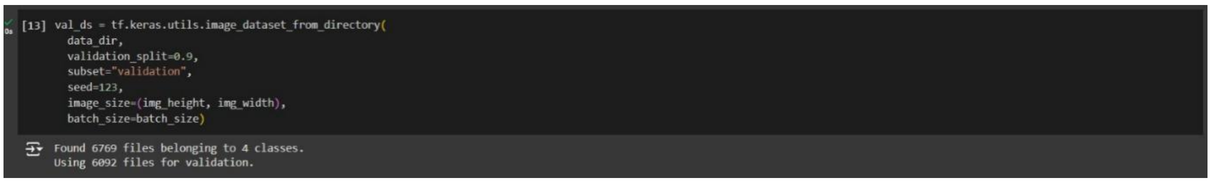
Fig 5: Working of CNN

Screenshots:



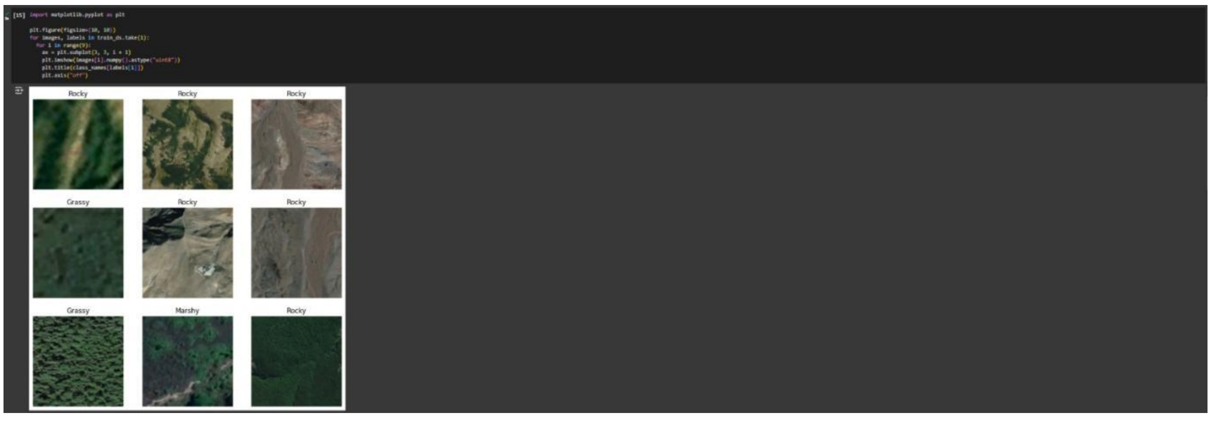
```
[12] train_ds = tf.keras.utils.image_dataset_from_directory(
       data_dir,
       validation_split=0.2,
       subset="training",
       seed=123,
       image_size=(img_height, img_width),
       batch_size=batch_size)

Found 6769 files belonging to 4 classes.
Using 5416 files for training.
```

Fig 6: Training data split



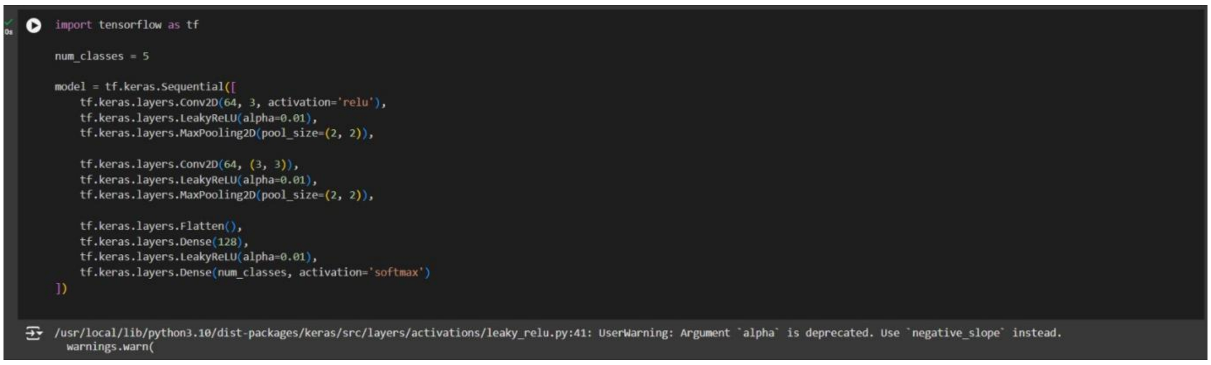
```
[13] val_ds = tf.keras.utils.image_dataset_from_directory(
       data_dir,
       validation_split=0.9,
       subset="validation",
       seed=123,
       image_size=(img_height, img_width),
       batch_size=batch_size)

Found 6769 files belonging to 4 classes.
Using 6092 files for validation.
```

Fig 7: Validation data split



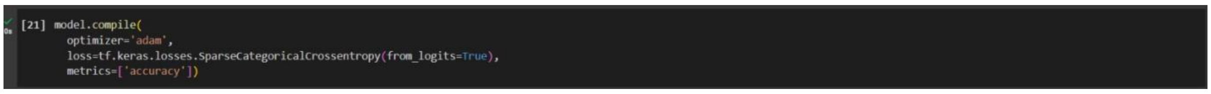
```
[15] import matplotlib.pyplot as plt

     plt.figure(figsize=(10, 10))
     for images, labels in train_ds.take(1):
       for i in range(9):
         ax = plt.subplot(3, 3, i + 1)
         plt.imshow(images[i].numpy().astype("uint8"))
         plt.title(class_names[labels[i]])
         plt.axis("off")
```

Fig 8: Dataset visualization



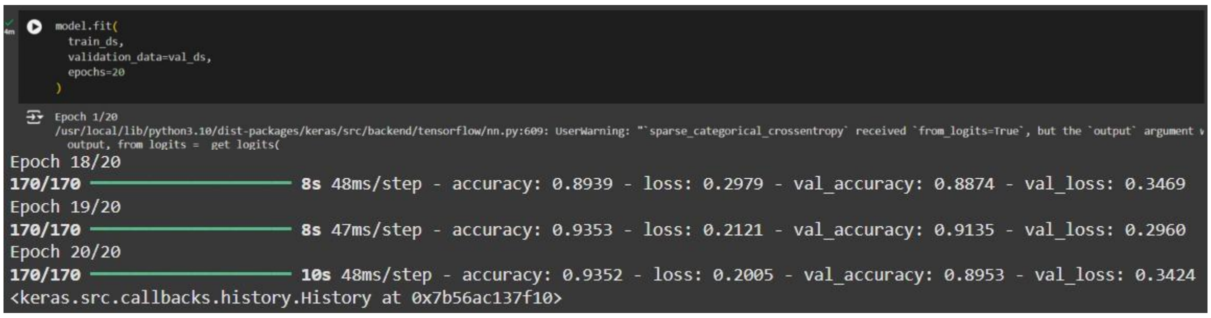
```
import tensorflow as tf

num_classes = 5

model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(64, 3, activation='relu'),
    tf.keras.layers.LeakyReLU(alpha=0.01),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),

    tf.keras.layers.Conv2D(64, (3, 3)),
    tf.keras.layers.LeakyReLU(alpha=0.01),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128),
    tf.keras.layers.LeakyReLU(alpha=0.01),
    tf.keras.layers.Dense(num_classes, activation='softmax')
])

/usr/local/lib/python3.10/dist-packages/keras/src/layers/activations/leaky_relu.py:41: UserWarning: Argument `alpha` is deprecated. Use `negative_slope` instead.
  warnings.warn(
```

Fig 9: Model Architecture



```
[21] model.compile(
       optimizer='adam',
       loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
       metrics=['accuracy'])
```

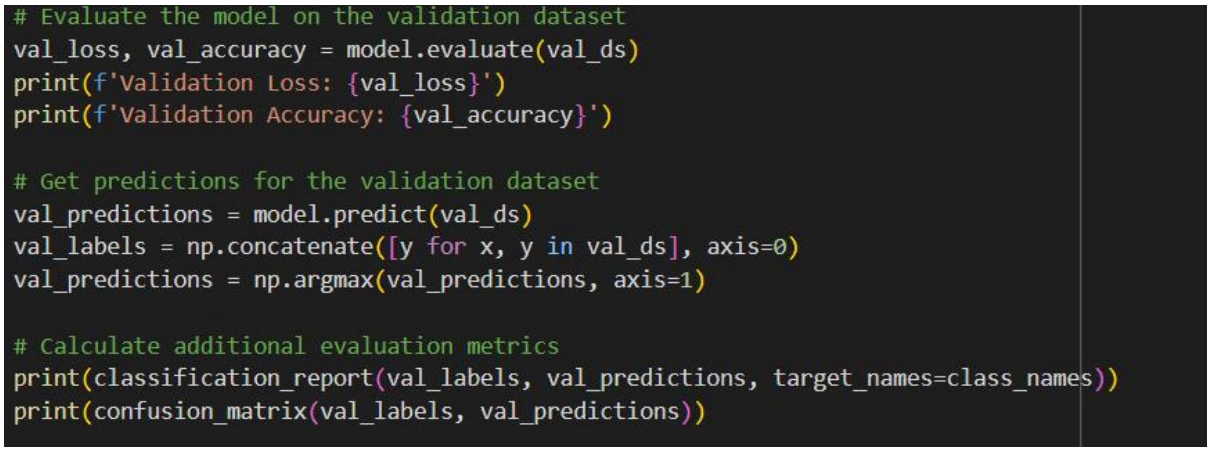
Fig 10: Loss reduction



```
model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=20
)

Epoch 1/20
/usr/local/lib/python3.10/dist-packages/keras/src/backend/tensorflow/nn.py:609: UserWarning: "`sparse_categorical_crossentropy` received `from_logits=True`, but the `output` argument w
  output, from logits =  get logits(
Epoch 18/20
170/170 ━━━━━━━━━━━━━━━━━━━━ 8s 48ms/step - accuracy: 0.8939 - loss: 0.2979 - val_accuracy: 0.8874 - val_loss: 0.3469
Epoch 19/20
170/170 ━━━━━━━━━━━━━━━━━━━━ 8s 47ms/step - accuracy: 0.9353 - loss: 0.2121 - val_accuracy: 0.9135 - val_loss: 0.2960
Epoch 20/20
170/170 ━━━━━━━━━━━━━━━━━━━━ 10s 48ms/step - accuracy: 0.9352 - loss: 0.2005 - val_accuracy: 0.8953 - val_loss: 0.3424
<keras.src.callbacks.history.History at 0x7b56ac137f10>
```

Fig 11: Training

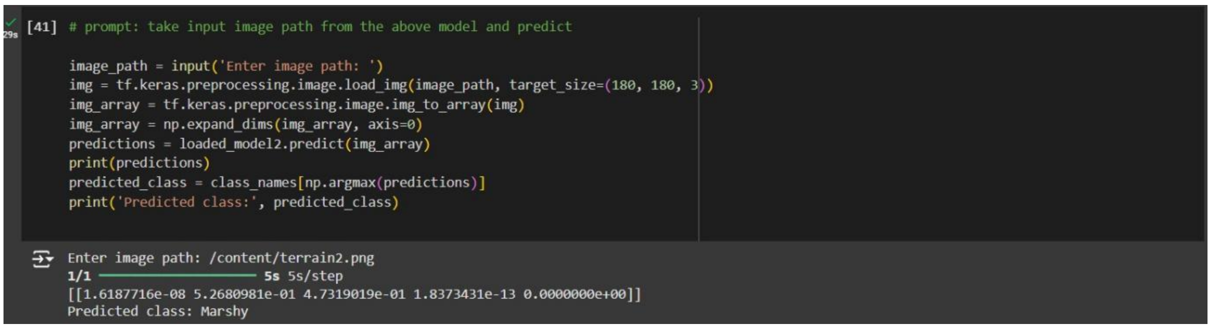
Sample Image Display: A screenshot of the sample image displayed during data verification.



```
# Evaluate the model on the validation dataset
val_loss, val_accuracy = model.evaluate(val_ds)
print(f'Validation Loss: {val_loss}')
print(f'Validation Accuracy: {val_accuracy}')

# Get predictions for the validation dataset
val_predictions = model.predict(val_ds)
val_labels = np.concatenate([y for x, y in val_ds], axis=0)
val_predictions = np.argmax(val_predictions, axis=1)

# Calculate additional evaluation metrics
print(classification_report(val_labels, val_predictions, target_names=class_names))
print(confusion_matrix(val_labels, val_predictions))
```

Fig 12: Model evaluation

Prediction Results: A screenshot of the prediction results for a sample image.



```
[41] # prompt: take input image path from the above model and predict

     image_path = input('Enter image path: ')
     img = tf.keras.preprocessing.image.load_img(image_path, target_size=(180, 180, 3))
     img_array = tf.keras.preprocessing.image.img_to_array(img)
     img_array = np.expand_dims(img_array, axis=0)
     predictions = loaded_model2.predict(img_array)
     print(predictions)
     predicted_class = class_names[np.argmax(predictions)]
     print('Predicted class:', predicted_class)
```

```
Enter image path: /content/terrain2.png
1/1 ━━━━━━━━━━━━━━━━━━━━ 5s 5s/step
[[1.6187716e-08 5.2680981e-01 4.7319019e-01 1.8373431e-13 0.0000000e+00]]
Predicted class: Marshy
```

Fig 13: Predict Results

4.4 Quality Assurance

In the working organization, if some department is there to verify the quality of your work, they can produce a certificate or guidelines followed.

Quality Assurance Measures:

Code Review: The code was reviewed by peers and mentors to ensure it adheres to best practices and coding standards.

Unit Testing: Unit tests were written to verify the functionality of individual components of the system.

Integration Testing: Integration tests were conducted to ensure that different components of the system work together seamlessly.

Performance Testing: The system's performance was tested to ensure it meets the non-functional requirements, such as processing time and accuracy.

Documentation: Comprehensive documentation was prepared to explain the project's methodology, implementation, and results.

Chapter 5

Standards Adopted

5.1 Design Standards

Recommended Practices for Project Design:

IEEE Standards

IEEE 1016-2009: Standard for Information Technology—Systems Design—Software Design Descriptions. This standard provides guidelines for documenting software design descriptions.

IEEE 830-1998: Recommended Practice for Software Requirements Specifications. This standard provides guidelines for documenting software requirements.

ISO Standards

ISO/IEC 29110: Software Engineering—Lifecycle Profiles. This standard provides guidelines for software engineering processes and lifecycle management.

ISO/IEC 12207: Systems and Software Engineering—Software Life Cycle Processes. This standard provides guidelines for software life cycle processes, including design and development.

5.2 Coding Standards

Coding standards are collections of coding rules, guidelines, and best practices. Adhering to these standards ensures that the code is readable, maintainable, and consistent.

Coding Standards Followed:

- Write as Few Lines as Possible
- Keep the code concise and avoid unnecessary lines.
- Use Appropriate Naming Conventions
- Use meaningful and descriptive names for variables, functions, and classes.
- Follow the snake case convention for variable and function names.
- Follow the CamelCase convention for class names.
- Segment Blocks of Code into Paragraphs
- Organize the code into logical sections with clear separation.
- Use Indentation to Mark the Beginning and End of Control Structures

- Use consistent indentation to clearly define the scope of control structures such as loops and conditionals.
- Don’t Use Lengthy Functions
- Keep functions short and focused on a single task.
- Break down complex functions into smaller, reusable functions.
- Document the Code
- Use comments to explain the purpose and functionality of the code.
- Include docstrings for functions and classes to describe their behavior and usage.
- Follow PEP 8 Guidelines
- Adhere to the PEP 8 style guide for Python code, which provides guidelines for code formatting, naming conventions, and best practices.

5.3 Testing Standards

There are some ISO and IEEE standards for quality assurance and testing of the product. Adhering to these standards ensures the reliability and effectiveness of the testing process. Standards Followed for Testing and Verification:

1. IEEE Standards

   a. *IEEE 829-2008:* Standard for Software and System Test Documentation. This standard provides guidelines for documenting software and system test plans, cases, and reports.
   b. *IEEE 1008-1987:* Standard for Software Unit Testing. This standard provides guidelines for unit testing of software components.

2. ISO Standards

   a. *ISO/IEC 29119:* Software Testing. This standard provides guidelines for software testing processes, including planning, execution, and reporting.
   b. *ISO/IEC 17025:* General Requirements for the Competence of Testing and Calibration Laboratories. This standard provides guidelines for the competence of testing laboratories.

Chapter 6

Conclusion and Future Scope

Conclusion

- Convolutional Neural Networks (CNNs) are successfully applied in the terrain image classification project to categorise landscape photos into predetermined categories. Data collection and preprocessing, model construction and training, assessment, and prediction were among the project's crucial phases.
- To guarantee the system's quality and dependability, we followed best practices and standards in design, coding, and testing throughout the project. The model's success in categorising terrain photos is demonstrated by its high accuracy on the validation dataset.
- The project's execution demonstrated how deep learning may be used to tackle challenging image classification issues. The CNN model was constructed and trained using a strong framework made possible by the use of TensorFlow and Keras. To guarantee the system's performance and functioning, the project also involved extensive testing and verification.
- All things considered, the terrain picture categorisation system created for this research offers a useful tool for a number of uses, such as geological surveys, agricultural planning, and environmental monitoring. The accuracy and efficiency of these applications can be greatly increased by the system's capacity to classify landscape photos.

Future Scope

The future scope of this project lies in deploying the trained model on the front end, making it accessible to a broader range of users. Here are some potential avenues for future development:

Web Application

- *Development:* Create a web application that allows users to upload terrain images and receive classification results in real-time.
- *Technologies:* Use web frameworks such as Flask or Django for the backend and HTML, CSS, and JavaScript for the frontend.
- *User Interface:* Design a user-friendly interface that guides users through the process of uploading images and viewing results.
- *Deployment:* Deploy the web application on a cloud platform such as AWS, Google Cloud, or Heroku to ensure scalability and accessibility.

Mobile Application

- *Development:* Develop a mobile application that enables users to capture terrain images using their smartphone cameras and receive classification results instantly.
- *Technologies:* Use mobile development frameworks such as React Native or Flutter for cross-platform compatibility.
- *User Interface:* Design an intuitive and responsive interface that provides a seamless user experience.
- *Deployment:* Publish the mobile application on app stores such as Google Play and Apple App Store to reach a wider audience.

API Integration

- *Development:* Create an API that allows other applications and systems to integrate the terrain image classification model.
- *Technologies:* Use RESTful API standards and frameworks such as FastAPI or Django REST Framework.
- *Documentation:* Provide comprehensive API documentation to guide developers on how to use the API.
- *Deployment:* Host the API on a cloud platform to ensure high availability and scalability.

Real-Time Monitoring

- *Development:* Implement real-time monitoring systems that use the terrain image classification model to continuously analyze terrain images from various sources.
- *Applications:* Use the system for environmental monitoring, disaster response, and infrastructure inspection.
- *Technologies:* Integrate the model with IoT devices, drones, and satellite imagery to capture and analyze terrain images in real-time.

Model Improvement

- *Continuous Learning:* Implement mechanisms for continuous learning and model improvement based on user feedback and new data.
- *Transfer Learning:* Explore transfer learning techniques to adapt the model to new types of terrain or different domains.
- *Ensemble Methods:* Combine the CNN model with other machine learning techniques to improve accuracy and robustness.

User Feedback and Iteration

- *Feedback Collection:* Collect user feedback to identify areas for improvement and new features.
- *Iterative Development:* Follow an iterative development approach to continuously enhance the system based on user feedback and evolving requirements.

References

1. https://www.tensorflow.org/tutorials/images/cnn#:~:text=The%20CIFAR10%20dataset%20contains%2060,000%20color

2. S. Y. Chaganti, I. Nanda, K. R. Pandi, T. G. N. R. S. N. Prudhvith and N. Kumar, "Image Classification using SVM and CNN," 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA), Gunupur, India, 2020, pp. 1-5, doi: 10.1109/ICCSEA49143.2020.9132851.

3. https://scikitlearn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

4. https://www.kaggle.com/datasets/atharv1610/terrain-recognition

Individual Contribution

Aditya Ranjan
21052301

Data Collection and Preprocessing

Individual contribution and findings: My role in the project was to focus on the data collection and preprocessing aspects of the terrain image classification system. I was responsible for downloading and extracting the dataset, counting the number of images, and displaying sample images to verify the data. Additionally, I pre-processed the images by resizing and normalizing them to ensure they were in a suitable format for training the model. My technical findings included the importance of data preprocessing in improving the model's performance. I observed that normalizing the pixel values to the range [0, 1] significantly enhanced the model's ability to learn from the data. The experience gained during the implementation provided me with valuable insights into data preprocessing techniques and their impact on model training.

Planning: My planning involved setting up the development environment, including installing necessary libraries such as TensorFlow and Keras. I also organized the dataset into training and validation sets and ensured that the images were properly labelled.

Individual contribution to project report preparation: I contributed to the sections related to data collection and preprocessing in the project report. I drafted the Data Preparation and Preprocessing chapters, ensuring that the content was accurate and comprehensive. Additionally, I reviewed and edited the report to ensure consistency and clarity.

Individual contribution for project presentation and demonstration: My role in preparing the project presentation and demonstration involved creating slides that explained the data collection and preprocessing steps. I also demonstrated the preprocessed images and their impact on model training during the presentation.

Full signature of Supervisor:

……………………………..

Full signature of the student:

[illegible]
……………………………..

Anushka Singh
21052144

Model Building

Individual contribution and findings: My role in the project was to focus on the model building aspects of the terrain image classification system. I was responsible for defining the architecture of the CNN model, including the layers, activation functions, and pooling operations. Additionally, I compiled the model with appropriate loss functions and optimizers. My technical findings included the effectiveness of convolutional layers in extracting features from the images and the importance of regularization techniques in preventing overfitting. I observed that the model's architecture significantly impacted its ability to accurately classify terrain images. The experience gained during the implementation provided me with valuable insights into model building and optimization techniques

Planning: My planning involved researching different CNN architectures and selecting the most suitable one for our project. I also experimented with different configurations to optimize the model's performance

Individual contribution to project report preparation: I contributed to the sections related to model building in the project report. I drafted the Model Building chapter, ensuring that the content was accurate and comprehensive. Additionally, I reviewed and edited the report to ensure consistency and clarity.

Individual contribution for project presentation and demonstration: My role in preparing the project presentation and demonstration involved creating slides that explained the model architecture and its components. I also demonstrated the model's ability to classify terrain images during the presentation.

Full signature of Supervisor:

……………………………..

Full signature of the student:

Anushka.

……………………………..

Sameer Sinha
21052355

Model Training

Individual contribution and findings: My role in the project was to focus on the model training aspects of the terrain image classification system. I was responsible for training the model on the training dataset and validating its performance on the validation dataset. Additionally, I monitored the training process and adjusted hyperparameters as needed. My technical findings included the importance of hyperparameter tuning in optimizing the model's performance. I observed that adjusting the learning rate and batch size significantly impacted the model's ability to learn from the data. The experience gained during the implementation provided me with valuable insights into model training and optimization techniques.

Planning: My planning involved setting up the training environment, including configuring the hardware and software requirements. I also implemented techniques such as data augmentation to improve the model's generalization ability.

Individual contribution to project report preparation: I contributed to the sections related to model training in the project report. I drafted the Model Training chapter, ensuring that the content was accurate and comprehensive. Additionally, I reviewed and edited the report to ensure consistency and clarity.

Individual contribution for project presentation and demonstration: My role in preparing the project presentation and demonstration involved creating slides that explained the model training process and its outcomes. I also demonstrated the model's performance on the validation dataset during the presentation

Full signature of Supervisor:	Full signature of the student:
……………………………..	Sameer Sinha ……………………………..

Priyanshu Midha
21052344

Model Evaluation

Individual contribution and findings: My role in the project was to focus on the model evaluation aspects of the terrain image classification system. I was responsible for evaluating the model's performance on the validation dataset and calculating evaluation metrics such as accuracy, precision, recall, and F1-score. Additionally, I analyzed the model's performance to identify areas for improvement. My technical findings included the importance of evaluation metrics in assessing the model's performance. I observed that the model's accuracy on the validation dataset was a crucial indicator of its effectiveness in classifying terrain images. The experience gained during the implementation provided me with valuable insights into model evaluation and performance analysis techniques.

Planning: My planning involved setting up the evaluation environment, including configuring the hardware and software requirements. With the help of graphs I also evaluated whether the model was underfitting or overfitting.

Individual contribution to project report preparation: I contributed to the sections related to model evaluation in the project report. I drafted the Model Evaluation chapter, ensuring that the content was accurate and comprehensive. Additionally, I reviewed and edited the report to ensure consistency and clarity.

Individual contribution for project presentation and demonstration: My role in preparing the project presentation and demonstration involved creating slides that explained the model evaluation process and its outcomes. I also demonstrated the model's performance on the validation dataset during the presentation.

Full signature of Supervisor:	Full signature of the student:
……………………………..	Priyanshu Midha ……………………………..

Piyush Raj
21052341

Testing and Verification

Individual contribution and findings: My role in the project was to focus on the testing and verification aspects of the terrain image classification system. I was responsible for developing test cases, executing tests, and verifying the system's functionality and performance. Additionally, I documented the test results and identified areas for improvement. My technical findings included the importance of testing and verification in ensuring the system's reliability and effectiveness. I observed that thorough testing significantly enhanced the system's ability to accurately classify terrain images. The experience gained during the implementation provided me with valuable insights into testing and verification techniques.

Planning: My planning involved setting up the testing environment, including configuring the hardware and software requirements. I also implemented techniques such as unit testing, integration testing, and system testing to ensure the comprehensiveness of the testing process.

Individual contribution to project report preparation: I contributed to the sections related to testing and verification in the project report. I drafted the Testing or Verification Plan chapter, ensuring that the content was accurate and comprehensive. Additionally, I reviewed and edited the report to ensure consistency and clarity.

Individual contribution for project presentation and demonstration: My role in preparing the project presentation and demonstration involved creating slides that explained the testing and verification process and its outcomes. I also demonstrated the test results and their impact on the system's performance during the presentation.

Full signature of Supervisor:	Full signature of the student:
……………………………..	Piyush Raj ……………………………..

Mahak Mahawar
21051143

Documentation and Reporting + Testing and Verification

Individual contribution and findings: My role in the project was to focus on the documentation and reporting aspects of the terrain image classification system. I was responsible for documenting the project's methodology, implementation, results, and findings. Additionally, I ensured that the project report was well-structured, comprehensive, and easy to understand. My technical findings included the importance of clear and concise documentation in communicating the project's objectives, methodology, and results. I observed that well-structured documentation significantly enhanced the project's transparency and accessibility. The experience gained during the implementation provided me with valuable insights into documentation and reporting techniques.

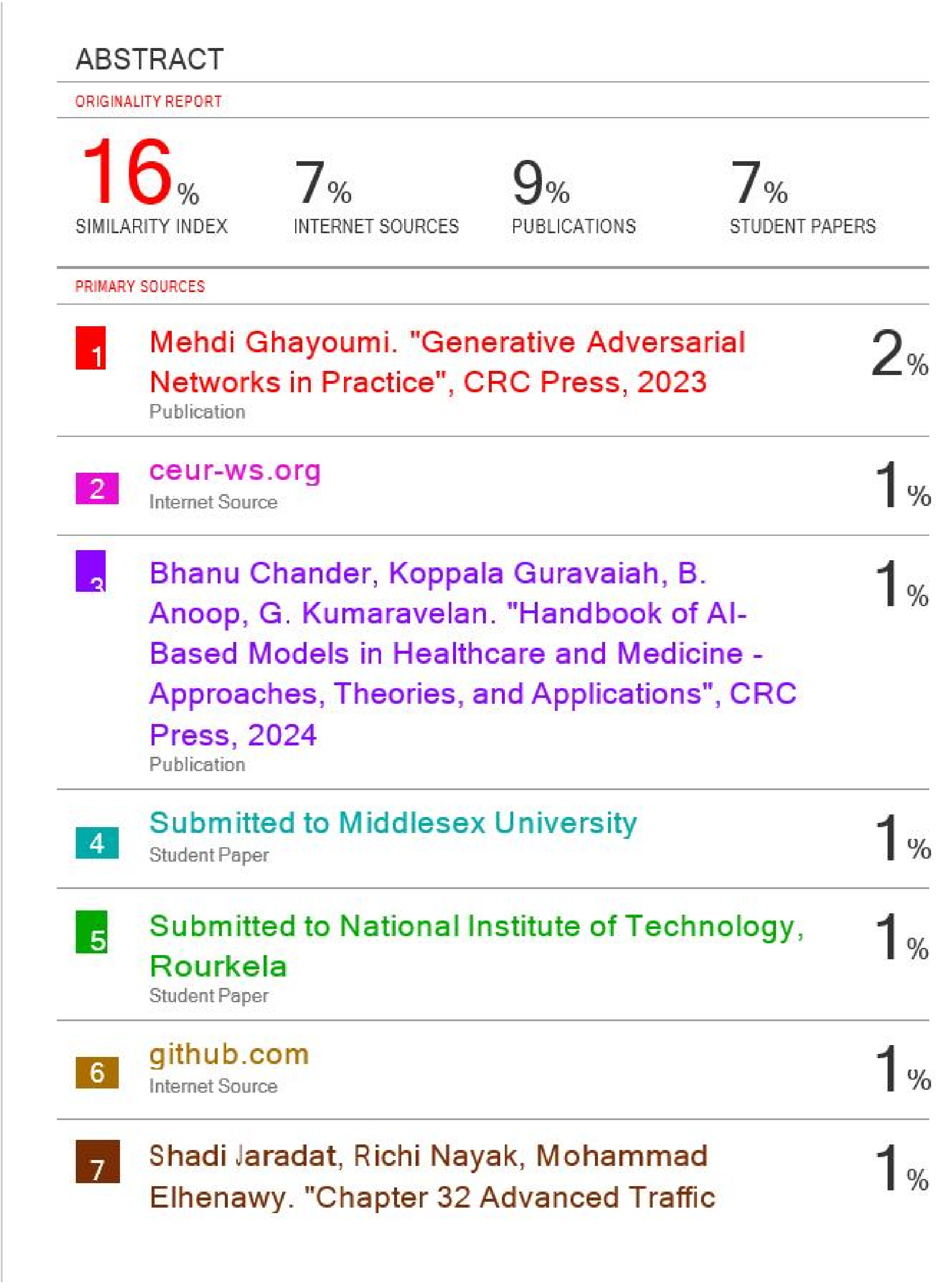
Planning: My planning involved setting up a documentation framework, including templates and guidelines for report preparation. I also coordinated with team members to gather their contributions and integrate them into the final report.

Individual contribution to project report preparation: I contributed to the overall structure and content of the project report. I drafted the Introduction, Methodology, Results, and Conclusion chapters, ensuring that the content was accurate and comprehensive. Additionally, I reviewed and edited the report to ensure consistency and clarity.

Individual contribution for project presentation and demonstration: My role in preparing the project presentation and demonstration involved creating slides that explained the project's objectives, methodology, results, and conclusions. I also ensured that the presentation was well-structured, engaging, and informative.

Full signature of Supervisor:	Full signature of the student:
……………………………..	Mahawar ……………………………..

Plagiarism Report



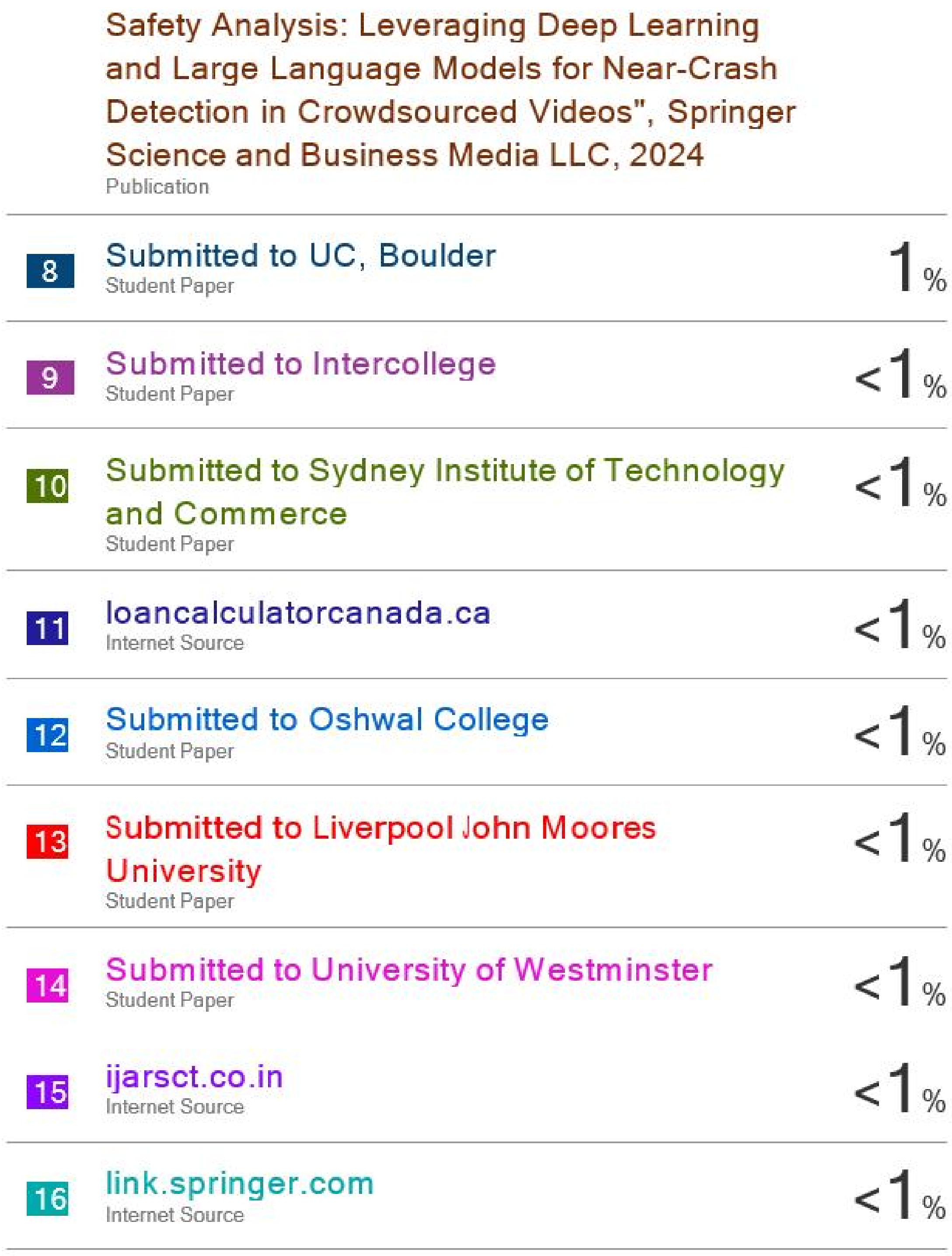
ABSTRACT

ORIGINALITY REPORT

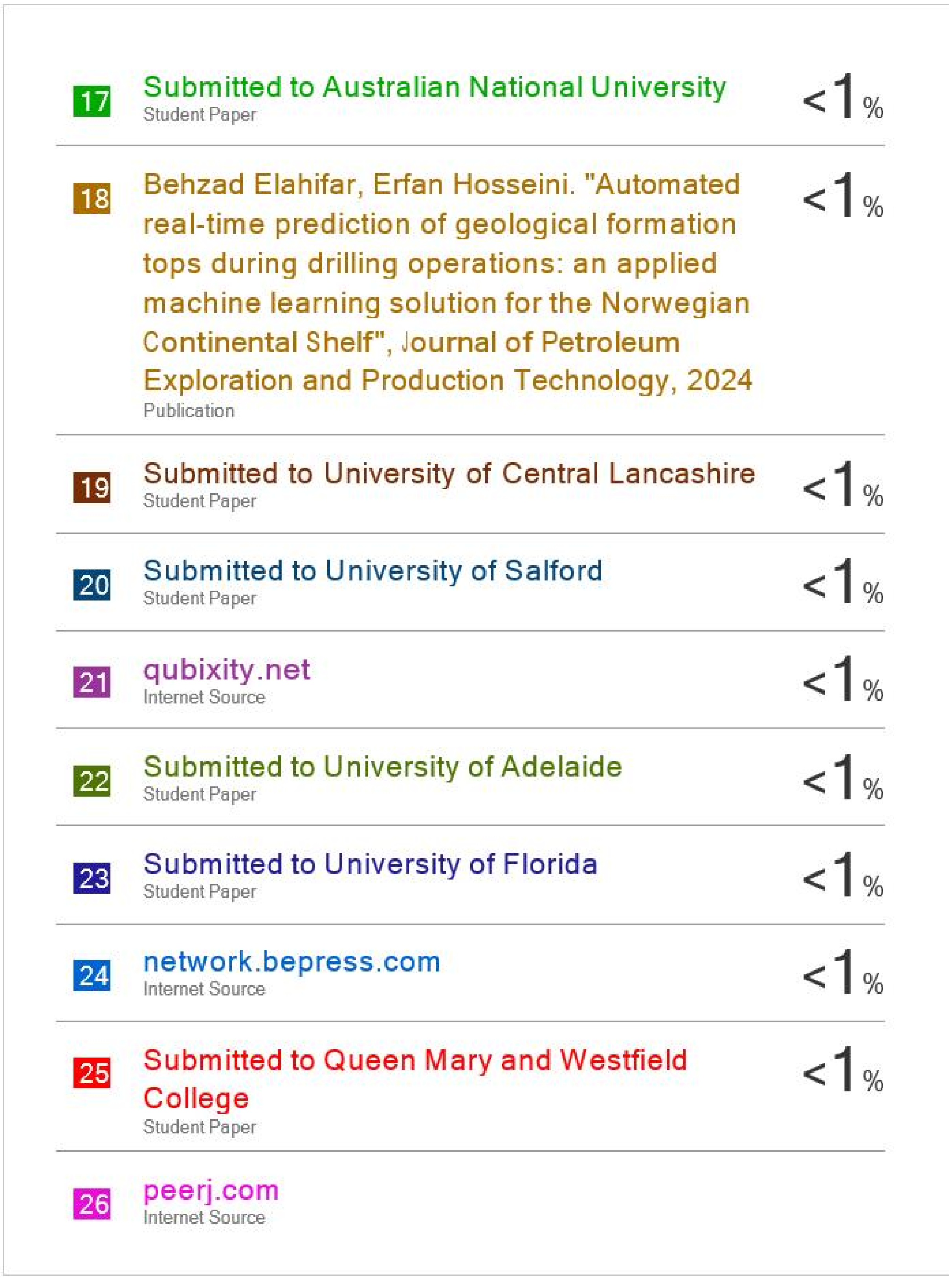
16%	7%	9%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

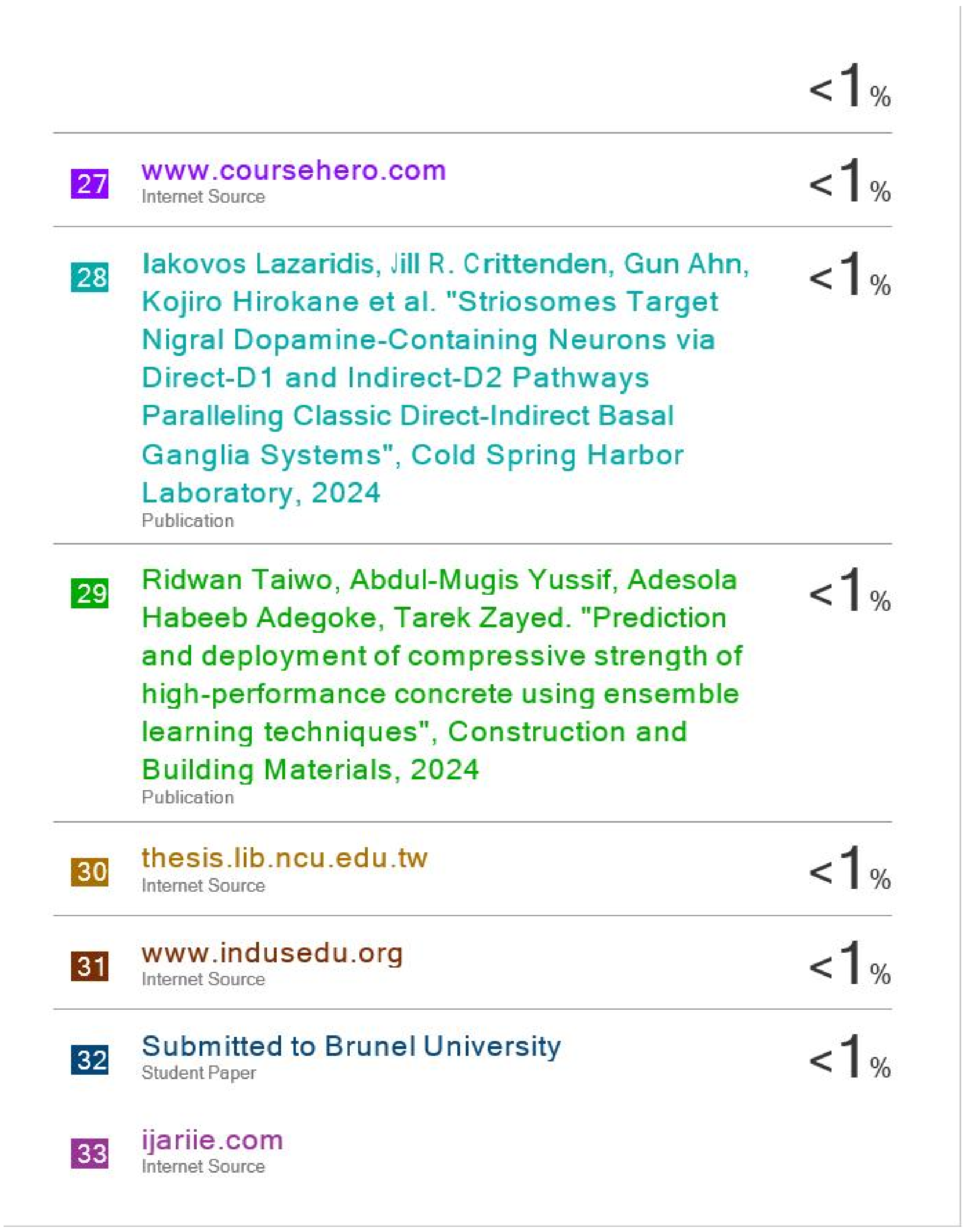
1	Mehdi Ghayoumi. "Generative Adversarial Networks in Practice", CRC Press, 2023 Publication	2%
2	ceur-ws.org Internet Source	1%
3	Bhanu Chander, Koppala Guravaiah, B. Anoop, G. Kumaravelan. "Handbook of AI-Based Models in Healthcare and Medicine - Approaches, Theories, and Applications", CRC Press, 2024 Publication	1%
4	Submitted to Middlesex University Student Paper	1%
5	Submitted to National Institute of Technology, Rourkela Student Paper	1%
6	github.com Internet Source	1%
7	Shadi Jaradat, Richi Nayak, Mohammad Elhenawy. "Chapter 32 Advanced Traffic	1%



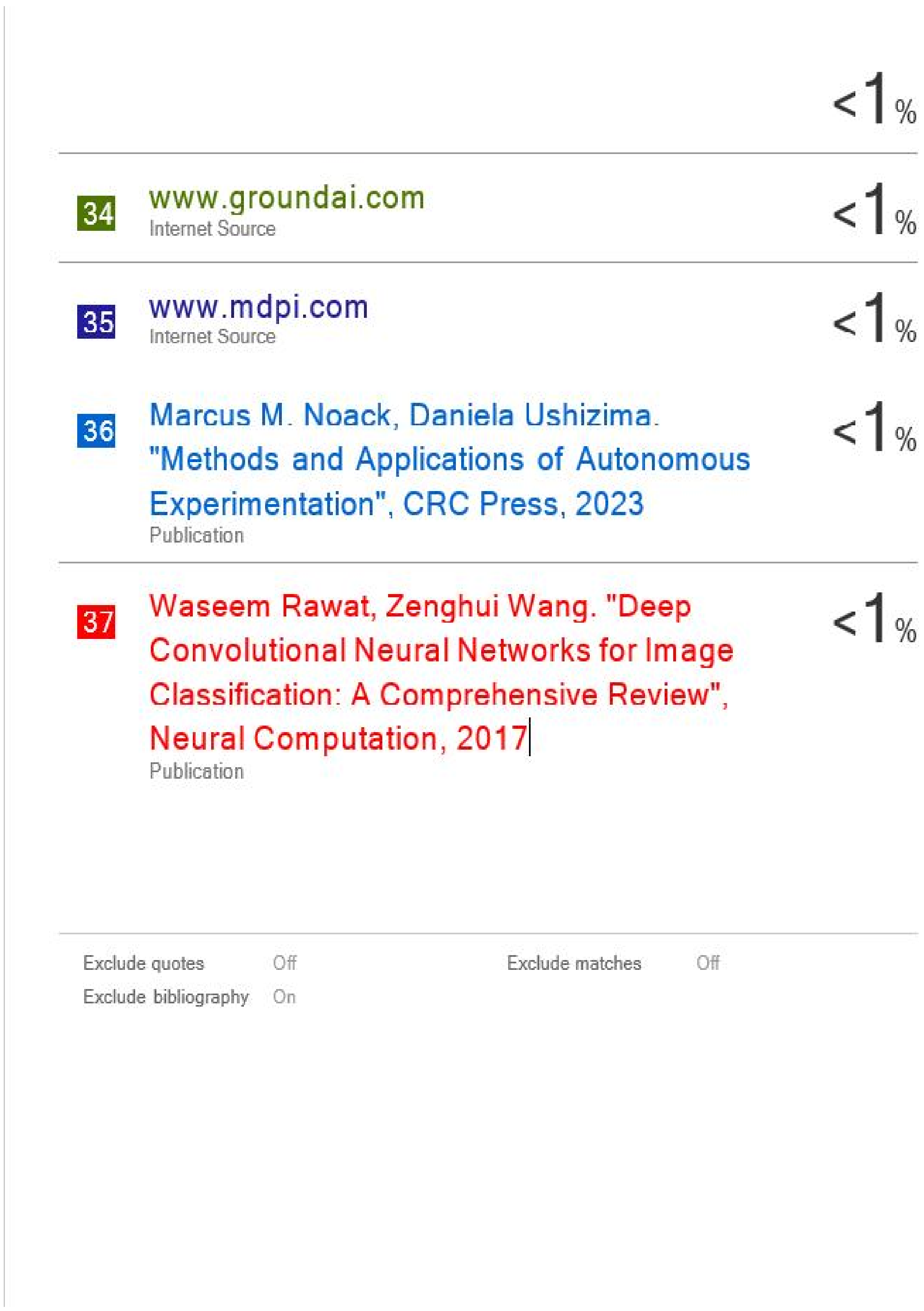
	Safety Analysis: Leveraging Deep Learning and Large Language Models for Near-Crash Detection in Crowdsourced Videos", Springer Science and Business Media LLC, 2024 Publication	
8	Submitted to UC, Boulder Student Paper	1 %
9	Submitted to Intercollege Student Paper	<1 %
10	Submitted to Sydney Institute of Technology and Commerce Student Paper	<1 %
11	loancalculatorcanada.ca Internet Source	<1 %
12	Submitted to Oshwal College Student Paper	<1 %
13	Submitted to Liverpool John Moores University Student Paper	<1 %
14	Submitted to University of Westminster Student Paper	<1 %
15	ijarsct.co.in Internet Source	<1 %
16	link.springer.com Internet Source	<1 %



17	Submitted to Australian National University Student Paper	<1%
18	Behzad Elahifar, Erfan Hosseini. "Automated real-time prediction of geological formation tops during drilling operations: an applied machine learning solution for the Norwegian Continental Shelf", Journal of Petroleum Exploration and Production Technology, 2024 Publication	<1%
19	Submitted to University of Central Lancashire Student Paper	<1%
20	Submitted to University of Salford Student Paper	<1%
21	qubixity.net Internet Source	<1%
22	Submitted to University of Adelaide Student Paper	<1%
23	Submitted to University of Florida Student Paper	<1%
24	network.bepress.com Internet Source	<1%
25	Submitted to Queen Mary and Westfield College Student Paper	<1%
26	peerj.com Internet Source	



		<1%
27	www.coursehero.com Internet Source	<1%
28	Iakovos Lazaridis, Jill R. Crittenden, Gun Ahn, Kojiro Hirokane et al. "Striosomes Target Nigral Dopamine-Containing Neurons via Direct-D1 and Indirect-D2 Pathways Paralleling Classic Direct-Indirect Basal Ganglia Systems", Cold Spring Harbor Laboratory, 2024 Publication	<1%
29	Ridwan Taiwo, Abdul-Mugis Yussif, Adesola Habeeb Adegoke, Tarek Zayed. "Prediction and deployment of compressive strength of high-performance concrete using ensemble learning techniques", Construction and Building Materials, 2024 Publication	<1%
30	thesis.lib.ncu.edu.tw Internet Source	<1%
31	www.indusedu.org Internet Source	<1%
32	Submitted to Brunel University Student Paper	<1%
33	ijariie.com Internet Source	



		<1%
34	www.groundai.com Internet Source	<1%
35	www.mdpi.com Internet Source	<1%
36	Marcus M. Noack, Daniela Ushizima. "Methods and Applications of Autonomous Experimentation", CRC Press, 2023 Publication	<1%
37	Waseem Rawat, Zenghui Wang. "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review", Neural Computation, 2017 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		