**Remote Sensing Image Scene Classification Meets Deep Learning Challenges, Methods, Benchmarks, and Opportunities**

**1. INTRODUCTION:**

With the proliferation of powerful sensor embedded smart-

phones, crowdsensing has become a leading paradigm which

leverages the pervasive smartphone users to collect data

efﬁciently. In a typical crowdsensing application, a server

posts the required sensing information and recruits a set of

smartphone users to collect sensing data. After smartphone

users send sensing data to the server, the server aggregates

the sensing data to measure phenomena of common inter-

est, i.e., real-time trafﬁc conditions, environmental pollution

quality or environmental noise pollution.

The accuracy of estimating the common interest depends

on the high quality contributions of highly skilled users.

While providing the high quality contributions, smartphone

users consume their energy and the resources of their smart-

phones such as battery, storage and computing power. In addi-

tion, users may expose themselves to potential privacy threats

as the sensed data contain time or location tags. Thus, the

contributors should be given enough rewards to compensate

for their resource consumption or potential privacy leaks.

As is known to all, a user wants to maximize her own proﬁt,

and may lie or impersonate others to get more payment.

Therefore, the design of a secure and truthful incentive mech-

anism is particularly important.

Many incentive mechanisms have been proposed and

implemented, such as the reputation systems and monetary

approaches. Reputation systems [1] can help identify uncoop-

erative users, but ignore a formal speciﬁcation and analysis of

the incentive types and suffer sybil attacks [2] and whitewash

attacks [3]. Monetary approaches [26] could be the most

promising due to their explicit and ﬂexible incentive meth-

ods. Most monetary schemes use pricing strategies to design

truthful incentive mechanisms, in which the server and smart-

phone users cannot increase their utility by cheating or col-

luding with others. While some other privacy-preserving

incentive mechanisms have been proposed for protectin

Remote sensing (RS) is an active research subject in the area of satellite image analysis for the discrete categorization of images into various scene category classes based on image content. The satellite sensors periodically generate volumes of images that require effective feature processing for various computer vision applications, such as scene labeling, feature localization, image recognition, scene parsing, street scene segmentation, and many others. Several image feature analysis methods have been developed to this effect. References groups the feature analysis methods into three categories: (a) low-level, which focuses on human-engineering skills, (b) medium-level, i.e., unsupervised methods that automatically learn features from images, and (c) high-level, i.e., deep learning methods that rely on supervised learning for feature analysis and representation.

The satellite-generated images vary in texture, shape, color, spectrum information, scale, etc. Additionally, remote sensing images exhibit the following characteristics:

* Complex spatial arrangements. Remotely sensed images have significant variations in the semantics (for instance, the scene images; of agriculture, airport, commercial areas, and residential areas are typical examples of varying scene image semantics). Extracting the semantic features from images requires effective computer vision techniques.
* Low inter-class variance. Some scene images are similar (e.g., agriculture and forest, dense residential areas, and residential-area). This characteristic is referred to as low intra-class variance. Achieving accurate scene classification under this circumstance requires well-calibrated computer vision techniques.
* High intra-class variance. Those same class scene images are commonly taken at varying angles, scales, and viewpoints. This diverse variation of same-class images requires well-designed computer vision approaches that can extract the same pattern features from the remotely sensed images regardless of their variations.
* Noise: Remotely sensed images are taken under varying atmospheric conditions and at different seasons. The scene images may have variable illumination conditions and require robust feature-learning techniques against varying weather circumstances.

Recent studies have indicated that data-driven deep learning models attain state-of-the-art results in scene classification owing to their abilities in learning high-level abstract features from images. Developments in hardware for graphic processing units (GPUs) provide the capabilities to process the vast amount of data on deep learning frameworks. Deep learning gives an architecture platform for feature learning methods that comprise several processing steps to learn remote sensing image features at different abstraction levels. Convolutional neural networks (CNNs) are good at abstracting local features and progressively expanding their receptive fields for more abstractions. Transfer-based deep learning models work on the premise that fundamental elements of images are the same; thus, they utilize pre-trained models that are trained on large-scale datasets for remote sensing applications. Other studies develop models that fuse different CNNs in exploring their performances in scene classification. The application of transfer CNNs pre-trained models for feature extraction in remote sensing is limiting because they need to consider the features of remote sensing images. That is, remote sensing images are unique and vary in terms of background information, imaging angle, and spatial layout, factors that the CNNs pre-training models assume. Successful CNN-based deep learning models in scene classification, object detection, and semantic segmentation were incorporated into remote sensing subject to resolve the classic challenges efficiently since deep learning networks demonstrate to perform better in image classification, object detection, and semantic segmentation jobs.

Transformer-based deep learning methods such as the excellent teacher network guiding small networks (ET-GNet), and the label-free self-distillation contrastive with transformer architecture (Last) can learn long-range contextual information. An integration framework combines vision-transformer and CNNs to attain impressive results with remote sensing public datasets. Although deep learning methods attain awe-inspiring results in scene classification and object detection, they must improve to deliver practical and scientific problems. First, deep learning methods rely on available datasets and do not utilize geography knowledge or features, often resulting in inaccurate predictions. Second, lack of sufficient labeled datasets for training deep learning methods to generalize in new geographical regions. Due to these challenges, new research directions in geography-aware deep learning models are emerging. This research paradigm fuses knowledge and data in designing deeply blended deep learning models differently. Geography-aware deep learning is an emerging research area in remote sensing, and the research directions in this area include regional knowledge/features, physical knowledge/features, and spatial knowledge/features. The deep learning approaches for fusing geography knowledge and feature are majorly focusing on rule-based, semantic-networks, object-based, physical model-based, and neural network-based.

**1.1 Objective of the project:**

Remote sensing image scene classification, which aims at labelling remote sensing images with a set of semantic categories based on their contents, has broad applications in a range of fields. Propelled by the powerful feature learning capabilities of deep neural networks, remote sensing image scene classification driven by deep learning has drawn remarkable attention and achieved significant breakthroughs. However, to the best of our knowledge, a comprehensive review of recent achievements regarding deep learning for scene classification of remote sensing images is still lacking. Considering the rapid evolution of this field, this article provides a systematic survey of deep learning methods for remote sensing image scene classification by covering more than 160 papers. To be specific, we discuss the main challenges of remote sensing image scene classification and survey: first, autoencoder-based remote sensing image scene classification methods; second, convolutional neural network-based remote sensing image scene classification methods; and third, generative adversarial network-based remote sensing image scene classification methods. In addition, we introduce the benchmarks used for remote sensing image scene classification and summarize the performance of more than two dozen of representative algorithms on three commonly used benchmark datasets. Finally, we discuss the promising opportunities for further research.

**2. LITERATURE SURVEY:**

**Land-use scene classification using multi-scale completed local binary patterns**

In this paper, we introduce the completed local binary patterns (CLBP) operator for the first time on remote sensing land-use scene classification. To further improve the representation power of CLBP, we propose a multi-scale CLBP (MS-CLBP) descriptor to characterize the dominant texture features in multiple resolutions. Two different kinds of implementations of MS-CLBP equipped with the kernel-based extreme learning machine are investigated and compared in terms of classification accuracy and computational complexity. The proposed approach is extensively tested on the 21-class land-use dataset and the 19-class satellite scene dataset showing a consistent increase on performance when compared to the state of the arts.

**Multiresolution gray-scale and rotation invariant texture classification with local binary patterns**

Presents a theoretically very simple, yet efficient, multiresolution approach to gray-scale and rotation invariant texture classification based on local binary patterns and nonparametric discrimination of sample and prototype distributions. The method is based on recognizing that certain local binary patterns, termed "uniform," are fundamental properties of local image texture and their occurrence histogram is proven to be a very powerful texture feature. We derive a generalized gray-scale and rotation invariant operator presentation that allows for detecting the "uniform" patterns for any quantization of the angular space and for any spatial resolution and presents a method for combining multiple operators for multiresolution analysis. The proposed approach is very robust in terms of gray-scale variations since the operator is, by definition, invariant against any monotonic transformation of the gray scale. Another advantage is computational simplicity as the operator can be realized with a few operations in a small neighborhood and a lookup table. Experimental results demonstrate that good discrimination can be achieved with the occurrence statistics of simple rotation invariant local binary patterns.

**A new paradigm for remote sensing**

Information extraction is a key activity for remote sensing images. A common distinction exists between knowledge-driven and data-driven methods. Knowledge-driven methods have advanced reasoning ability and interpretability, but have difficulty in handling complicated tasks since prior knowledge is usually limited when facing the highly complex spatial patterns and geoscience phenomena found in reality. Data-driven models, especially those emerging in machine learning (ML) and deep learning (DL), have achieved substantial progress in geoscience and [remote sensing applications](https://www.sciencedirect.com/topics/earth-and-planetary-sciences/remote-sensing-application). Although DL models have powerful feature learning and representation capabilities, traditional DL has inherent problems including working as a black box and generally requiring a large number of labeled training data. The focus of this paper is on methods that integrate domain knowledge, such as geoscience knowledge and geoscience features (GK/GFs), into the design of DL models. The paper introduces the new paradigm of geoscience-aware deep learning (GADL), in which GK/GFs and DL models are combined deeply to extract information from remote sensing data. It first provides a comprehensive summary of GK/GFs used in GADL, which forms the basis for subsequent integration of GK/GFs with DL models. This is followed by a taxonomy of approaches for integrating GK/GFs with DL models. Several approaches are detailed using illustrative examples. Challenges and research prospects in GADL are then discussed. Developing more novel and advanced methods in GADL is expected to become the prevailing trend in advancing remotely sensed information extraction in the future.

**LaST: Label-Free Self-Distillation Contrastive Learning With Transformer Architecture for Remote Sensing Image Scene Classification**

The increase in self-supervised learning (SSL), especially contrastive learning, has enabled one to train deep neural network models with unlabeled data for remote sensing image (RSI) scene classification. Nevertheless, it still suffers from the following issues: 1) the performance of the contrastive learning method is significantly impacted by the hard negative sample (HNS) issue, since the RSI scenario is complex in semantics and rich in surface features; 2) the multiscale characteristic of RSI is missed in the existing contrastive learning methods; and 3) as the backbone of a deep learning model, especially in the case of limited annotation, a convolutional neural network (CNN) does not include the adequate receptive field of convolutional kernels to capture the broad contextual information of RSI. In this regard, we propose label-free self-distillation contrastive learning with a transformer architecture (LaST). We introduce the self-distillation contrastive learning mechanism to address the HNS issue. Specifically, the LaST architecture comprises two modules: scale alignment with a multicrop module and a long-range dependence capture backbone module. In the former, we present global–local crop and scale alignment to encourage local-to-global correspondence and acquire multiscale relations. Then, the distorted views are fed into a transformer as a backbone, which is good at capturing the long-range-dependent contextual information of the RSI while maintaining the spatial smoothness of the learned features. Experiments on public datasets show that in the downstream scene classification task, LaST improves the performance of the self-supervised trained model by a maximum of 2.18% compared to the HNS-impacted contrastive learning approaches, and only 1.5% of labeled data can achieve the performance of supervised training CNNs with 10% labeled data. Moreover, this letter supports the integration of a transformer architecture and self-supervised paradigms in RSI.

**Vision Transformer: An Excellent Teacher for Guiding Small Networks in Remote Sensing Image Scene Classification**

Scene classification is an active research topic in the remote sensing community, and complex spatial layouts with various types of objects bring huge challenges to classification. Convolutional neural network (CNN)-based methods attempt to explore the global features by gradually expanding the receptive field, while long-range contextual information is ignored. Vision transformer (ViT) can extract contextual features, but the learning ability of local information is limited, and it has a large computational complexity simultaneously. In this article, an end-to-end method is exploited by employing ViT as an excellent teacher for guiding small networks (ET-GSNet) in the remote sensing image scene classification. In the ET-GSNet, ResNet18 is selected as the student model, which integrates the superiorities of the two models via knowledge distillation (KD), and the computational complexity does not increase. In the KD process, the ViT and ResNet18 are optimized together without independent pretraining, and the learning rate of teacher model gradually decreases until zero, while the weight coefficient of the KD loss module is doubled. Based on the above procedures, dark knowledge from the teacher model can be transferred to the student model more smoothly. Experimental results on the four public remote sensing datasets demonstrate that the proposed ET-GSNet method possesses the superior classification performance compared to some state-of-the-art (SOTA) methods. In addition, we evaluate the ET-GSNet on a fine-grained ship recognition dataset, and the results show that our method has good generalization for different tasks in terms of some metrics.

**Semantic Segmentation of Urban Buildings from VHR Remote Sensing Imagery Using a Deep Convolutional Neural Network**

Urban building segmentation is a prevalent research domain for very high resolution (VHR) remote sensing; however, various appearances and complicated background of VHR remote sensing imagery make accurate semantic segmentation of urban buildings a challenge in relevant applications. Following the basic architecture of U-Net, an end-to-end deep convolutional neural network (denoted as DeepResUnet) was proposed, which can effectively perform urban building segmentation at pixel scale from VHR imagery and generate accurate segmentation results. The method contains two sub-networks: One is a cascade down-sampling network for extracting feature maps of buildings from the VHR image, and the other is an up-sampling network for reconstructing those extracted feature maps back to the same size of the input VHR image. The deep residual learning approach was adopted to facilitate training in order to alleviate the degradation problem that often occurred in the model training process. The proposed DeepResUnet was tested with aerial images with a spatial resolution of 0.075 m and was compared in performance under the exact same conditions with six other state-of-the-art networks—FCN-8s, SegNet, DeconvNet, U-Net, ResUNet and DeepUNet. Results of extensive experiments indicated that the proposed DeepResUnet outperformed the other six existing networks in semantic segmentation of urban buildings in terms of visual and quantitative evaluation, especially in labeling irregular-shape and small-size buildings with higher accuracy and entirety. Compared with the U-Net, the F1 score, Kappa coefficient and overall accuracy of DeepResUnet were improved by 3.52%, 4.67% and 1.72%, respectively. Moreover, the proposed DeepResUnet required much fewer parameters than the U-Net, highlighting its significant improvement among U-Net applications. Nevertheless, the inference time of DeepResUnet is slightly longer than that of the U-Net, which is subject to further improvement.

**3. SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM:**

In Existing system, we discuss the main challenges of remote sensing image scene classification and survey: first, autoencoder-based remote sensing image scene classification methods; second, convolutional neural network-based remote sensing image scene classification methods; and third, generative adversarial network-based remote sensing image scene classification methods.

**Disadvantages:**

1. Less Accuracy.
2. Unable to Recognize.

**3.2 PROPOSED SYSTEM:**

In propose paper author is giving review and challenges on various deep learning scene classification algorithms called VGG16, Auto Encoder-decoder and GAN (generative adversarial network). All algorithms face challenges and get confuse to classify scenes images which has common region. In propose paper author saying VGG19 is giving best result and author evaluate all algorithms performance by using various dataset such as NWPU, UC Merced and many more. Training all algorithms using various dataset may take days of time so we have trained VGG19 with NWPU dataset to classify 19 different remote scenes images. In below screen showing dataset used to train VGG19 algorithm

**Advantages**

1. Easy to Recognize.

2.More Accuracy.

**Modules**

To implement this project, we have designed following modules

1. Upload NWPU Satellite Scenes Dataset: using this module we will upload dataset to application
2. Preprocess Dataset: using this module we will read all images and then resize all images to equal size and then normalize and shuffle images and then split dataset into train and test where application use 80% dataset images for training and 20% for testing
3. Train VGG19 Algorithm: using this module we will feed 80% images as input to VGG19 to trained a model and this model will be applied on 20% test images to calculate prediction accuracy
4. VGG19 Training Graph: using this module we will plot VGG19 train and test graph
5. Remote Scenes Classification: using this module we will upload Remote Satellite test image and then VGG will predict or classify scenes from that image

**3.3. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

**Umbrella Activity**

**Umbrella Activity**

**Umbrella Activity**

1. Feasibility Study
2. TEAM FORMATION
3. Project Specification PREPARATION

Business Requirement Documentation

ANALYSIS & DESIGN

CODE

UNIT TEST

DOCUMENT CONTROL

ASSESSMENT

TRAINING

INTEGRATION & SYSTEM TESTING

DELIVERY/INSTALLATION

ACCEPTANCE TEST

Requirements Gathering

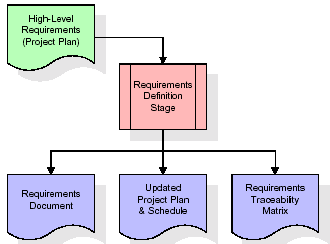
SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

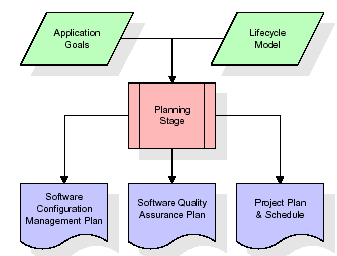
In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

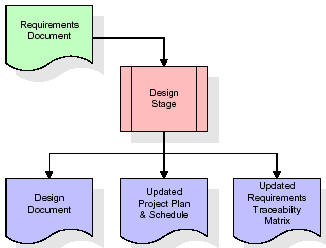
The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

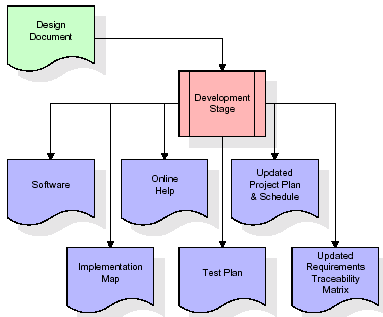
**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

  
When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

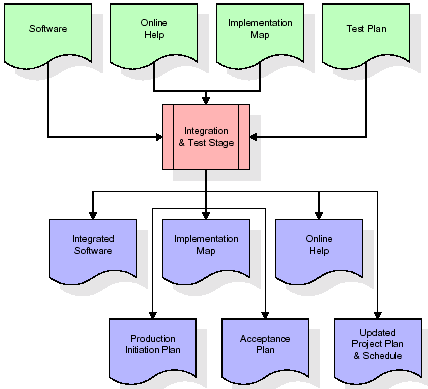
The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.

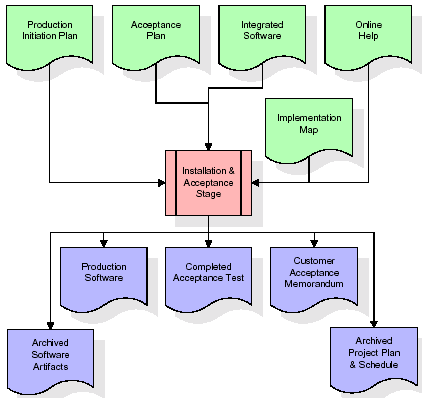


The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.4. Software Requirement Specification**

**3.4.1. Overall Description**

A Software Requirements Specification (SRS) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) is a complete description of the behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Nonfunctional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the [systems development lifecycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* [Business requirements](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms what must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify. Preliminary investigation examines project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:
* **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, there is nominal expenditure and economic feasibility for certain.

* **Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So, there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

* **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web-based user interface for audit workflow at NIC-CSD. Thus, it provides an easy access to. the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

**3.4.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user-friendly python Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through python capabilities.

**Software Interfaces**

The required software is python.

**SYSTEM REQUIREMENT:**

**HARDWARE REQUIREMENTS:**

# Processor - Intel i3(min)

* Speed - 1.1 GHz
* RAM - 4GB (min)
* Hard Disk - 500 GB

**SOFTWARE REQUIREMENTS:**

* Operating System - Windows10(min)
* Programming Language - Python (3.7.0)

**4. SYSTEM DESIGN**

**4. SYSTEM DESIGN**

**CLASS DIAGRAM:**

The class diagram is the main building block of object-oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake



**USECASE DIAGRAM:**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.



**SEQUENCE DIAGRAM:**

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.



**COLLABORATION DIAGRAM:**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.



**COMPONENT DIAGRAM:**

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.



**DEPLOYMENT DIAGRAM:**

A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g., JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

**ACTIVITY DIAGRAM:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So, the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.

Upload Dataset

Preprocess Dataset

Train Transfer Learning VGG19

Train Transfer Learning MobileNetV2

Training Graph

Remote Scenes Classification

**Data flow:**

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

User

1. Upload Dataset 2. successfully Upload Dataset

3. Preprocess Dataset 4. Successfully Preprocess Dataset

5. Train VGG19 6. Successfully Train VGG19

7. Train MobileNetV2 8. Successfully Train MobileNetV2

9. Training Graph 10. Successfully Training Graph

11.Remote Scenes Classification 12. Successfully Remote Scenes Classification

**5. IMPLEMETATION**

**5.1 PYTHON**

\* One of the most popular languages is Python. Guido van Rossum released this language in 1991. Python is available on the Mac, Windows, and Raspberry Pi operating systems. The syntax of Python is simple and identical to that of English. When compared to Python, it was seen that the other language requires a few extra lines.

\*It is an interpreter-based language because code may be run line by line after it has been written. This implies that rapid prototyping is possible across all platforms. Python is a big language with a free, binary-distributed interpreter standard library.

\* It is inferior to maintenance that is conducted and is straightforward to learn. It is an object-oriented, interpreted programming language. It supports several different programming paradigms in addition to object-oriented programming, including functional and procedural programming.

\* It supports several different programming paradigms in addition to object-oriented programming, including practical and procedural programming. Python is mighty while maintaining a relatively straightforward syntax. Classes, highly dynamic data types, modules, and exceptions are covered. Python can also be utilised by programmes that require programmable interfaces as an external language.

Here are some key features and characteristics of Python:

* Readability: Python emphasizes code readability with its clean and intuitive syntax. It uses indentation and whitespace to structure code blocks, making it easy to understand and maintain.
* Easy to Learn: Python's simplicity and readability make it an excellent choice for beginners. Its straightforward syntax and extensive documentation make it accessible for newcomers to programming.
* Interpreted Language: Python is an interpreted language, meaning that it doesn't need to be compiled before running. The Python interpreter reads and executes the code directly, making the development process faster and more interactive.
* Cross-platform Compatibility: Python is available for major operating systems like Windows, macOS, and Linux. This cross-platform compatibility allows developers to write code once and run it on different platforms without modifications.
* Large Standard Library: Python comes with a vast standard library that provides ready-to-use modules and functions for various tasks. It covers areas such as file I/O, networking, regular expressions, databases, and more, saving developers time and effort.
* Extensible and Modular: Python supports modular programming, enabling developers to organize code into reusable modules and packages. Additionally, Python allows integrating modules written in other languages, such as C or C++, providing flexibility and performance optimizations.
* Wide Range of Libraries and Frameworks: Python has a vibrant ecosystem with numerous third-party libraries and frameworks. These libraries, such as NumPy, pandas, TensorFlow, and Django, extend Python's capabilities for specific domains, making it a powerful tool for diverse applications.
* Object-Oriented: Python supports object-oriented programming (OOP) principles, allowing developers to create and work with classes and objects. OOP provides a structured approach to code organization, promoting code reuse and modularity.
* Dynamic Typing: Python is dynamically typed, meaning variable types are determined at runtime. Developers do not need to declare variable types explicitly, which enhances flexibility and simplifies code writing.

**5.2 Installation**

To install Python on your computer, follow these basic steps:

* Step 1: Visit the Python website Go to the official Python website at <https://www.python.org/>.
* Step 2: Select the operating system Choose the appropriate installer for your operating system. Python supports Windows, macOS, and various Linux distributions. Make sure to select the correct version that matches your operating system.
* Step 3: Check which version of Python is installed; if the 3.7.0 version is not there, uninstall it through the control panel and
* Step 4: Install Python 3.7.0 using Cmd.
* Step 5: Install the all libraries that required to run the project
* Step 6: Run

**5.3 Python Features:**

1. **Easy:** Because Python is a more accessible and straightforward language, Python programming is easier to learn.
2. **Interpreted language:** Python is an interpreted language, therefore it can be used to examine the code line by line and provide results.
3. **Open Source:** Python is a free online programming language since it is open-source.
4. **Portable:** Python is portable because the same code may be used on several computer standard
5. **libraries:** Python offers a sizable library that we may utilize to create applications quickly.
6. **GUI:** It stands for GUI (Graphical User Interface)
7. **Dynamical typed:** Python is a dynamically typed language, therefore the type of the value will be determined at runtime.

**5.4 Python GUI (Tkinter)**

* Python provides a wide range of options for GUI development (Graphical User Interfaces).
* Tkinter, the most widely used GUI technique, is used for all of them.
* The Tk GUI toolkit offered by Python is used with the conventional Python interface.
* Tkinter is the easiest and quickest way to write Python GUI programs.
* Using Tkinter, creating a GUI is simple.
* A part of Python's built-in library is Tkinter. The GUI programs were created.
* Python and Tkinter together give a straightforward and quick way. The Tk GUI toolkit's object-oriented user interface is called Tkinter.

Making a GUI application is easy using Tkinter. Following are the steps:

1) Install the Tkinter module in place.

2) The GUI applicatioMakeske the primary window

3) Include one or more of the widgets mentioned above in the GUI application.

4) Set up the main event loop such that it reacts to each user-initiated event.

Although Tkinter is the only GUI framework included in the Python standard library, Python includes a GUI framework. The default library for Python is called Tkinter. Tk is a scripting language often used in designing, testing, and developing GUIs. Tk is a free, open-source widget toolkit that may be used to build GUI applications in a wide range of computer languages.

**5.5 Python IDLE**

* Python IDLE offers a full-fledged file editor, which gives you the ability to write and execute Python programs from within this program. The built-in file editor also includes several features, like code completion and automatic indentation, that will speed up your coding workflow.
* Guido Van Rossum named Python after the British comedy group Monty Python while the name IDLE was chosen to pay tribute to Eric Idle, who was one of the Monty Python's founding members. IDLE comes bundled with the default implementation of the Python language since the 01.5. 2b1 release
* IDLE is used to execute statements similar to Python Shell. IDLE is used to create, modify, and execute Python code. IDLE provides a fully-featured text editor to write Python scripts and provides features like syntax highlighting, auto-completion, and smart indent.
* IDLE has two modes: interactive and script. We wrote our first program, “Hello, World!” in interactive mode. Interactive mode immediately returns the results of commands you enter into the shell. In script mode, you will write a script and then run it.
* The IDE Python IDLE is a good place to start as it helps you become familiar with the way Python works and understand its syntax. This IDE is good to start programming in Python due to its great debugger, but once you are fluent and start developing projects it is necessary to jump to another, more complete IDE.
* Python IDLE (Integrated Development and Learning Environment) is an interactive development environment included with the Python programming language. It provides a convenient way to write, execute, and debug Python code.

When you install Python, IDLE is typically installed along with it. To open IDLE, you can follow these steps:

* Open the command prompt (Windows) or terminal (macOS/Linux).
* Type "idle" and press Enter. Alternatively, you can specify the version with "idle3" or "idle2" for Python 3 or Python 2, respectively.
* Once IDLE is launched, you will see the Python shell, which is an interactive environment where you can type and execute Python code directly.

Here are some features and functionalities provided by Python IDLE:

* Editor: IDLE includes a text editor where you can write your Python code. It offers syntax highlighting, automatic indentation, and code completion to enhance your coding experience.
* Interactive Shell: The Python shell in IDLE allows you to execute Python code interactively. You can type commands, statements, or function calls directly in the shell, and Python will execute them immediately.
* Debugging: IDLE provides basic debugging capabilities to help you find and fix errors in your code. You can set breakpoints, step through code, inspect variables, and track the program's execution.
* Python Help: IDLE provides access to the Python documentation and built-in help. You can access the help menu to find information about Python modules, functions, classes, and more.
* Script Execution: In addition to the interactive shell, IDLE allows you to run Python scripts stored in files. You can write your code in the editor and execute it as a script to see the output or interact with the program.
* Customization: IDLE can be customized to suit your preferences. You can modify settings related to syntax highlighting, indentation, fonts, and more.
* Python IDLE serves as a beginner-friendly development environment and learning tool. It is suitable for writing small scripts, testing code snippets, experimenting with Python features, and learning the language's basics. However, for more advanced development projects, you may consider using other code editors or integrated development environments (IDEs) that provide additional features and better project management capabilities.

**5.6 Libraries**

In Python, libraries (also referred to as modules or packages) are collections of pre-written code that provide additional functionality and tools to extend the capabilities of the Python language. Libraries contain reusable code that developers can leverage to perform specific tasks without having to write everything from scratch.

Python libraries are designed to solve common problems, such as handling data, performing mathematical operations, interacting with databases, working with files, implementing networking protocols, creating graphical user interfaces (GUIs), and much more. They provide ready-to-use functions, classes, and methods that simplify complex operations and save development time.

**Libraries in Python offer various advantages:**

* Code Reusability:
* Efficiency:
* Collaboration
* Domain-Specific Functionality
* To use a Python library, you need to install it first.

There are some libraries following:

* **Pandas:**

Pandas are a Python computer language library for data analysis and manipulation. It offers a specific operation and data format for handling time series and numerical tables. It differs significantly from the release3-clause of the BSD license. It is a well-liked open-source of opinion that is utilized in machine learning and data analysis.

Pandas are a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Pandas are a Python library used for working with data sets.

* It has functions for analysing, cleaning, exploring, and manipulating data.
* The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.
* Pandas allow us to analyse big data and make conclusions based on statistical theories.
* Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science. Pandas are a Python library for data analysis. Started by Wes McKinney in 2008 out of a need for a powerful and flexible quantitative analysis tool, pandas have grown into one of the most popular Python libraries. It has an extremely active community of contributors. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself.

* **NumPy:**

The NumPy Python library for multi-dimensional, big-scale matrices adds a huge number of high-level mathematical functions. It is possible to modify NumPy by utilizing a Python library. Along with line, algebra, and the Fourier transform operations, it also contains several matrices-related functions.

NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

* NumPy is a Python library used for working with arrays.
* It also has functions for working in domain of linear algebra, Fourier transform, and matrices.
* NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
* NumPy stands for Numerical Python.
* In Python we have lists that serve the purpose of arrays, but they are slow to process.
* NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
* The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.
* Arrays are very frequently used in data science, where speed and resources are very important.
* **Matplotlib:**

It is a multi-platform, array-based data visualization framework built to interact with the whole SciPy stack. MATLAB is proposed as an open-source alternative. Matplotlib is a Python extension and a cross-platform toolkit for graphical plotting and visualization.

Matplotlib is a popular Python library for creating static, animated, and interactive visualizations. It provides a flexible and comprehensive set of tools for generating plots, charts, histograms, scatter plots, and more. Matplotlib is widely used in various fields, including data analysis, scientific research, and data visualization.

Here are some key features and functionalities of the Matplotlib library:

* Plotting Functions
* Customization Options
* Multiple Interfaces
* Integration with NumPy and pandas
* Subplots and Figures:
* Saving and Exporting
* **Scikit-learn:**

The most stable and practical machine learning library for Python is scikit-learn. Regression, dimensionality reduction, classification, and clustering are just a few of the helpful tools it provides through the Python interface for statistical modeling and machine learning. It is an essential part of the Python machine learning toolbox used by JP Morgan. It is frequently used in various machine learning applications, including classification and predictive analysis.

Scikit-learn (also referred to as sklearn) is a widely used open-source machine learning library for Python. It provides a comprehensive set of tools and algorithms for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, model selection, and pre-processing.

Here are some key features and functionalities of the Scikit-learn library:

* Easy-to-Use Interface:
* Broad Range of Algorithms:
* Data Pre-processing and Feature Engineering:
* Model Evaluation and Validation:
* Integration with NumPy and pandas:
* Robust Documentation and Community Support:
* **Keras:**

\* Google's Keras is a cutting-edge deep learning API for creating neural networks. It is created in Python and is designed to simplify the development of neural networks. Additionally, it enables the use of various neural networks for computation. Deep learning models are developed and tested using the free and open-source Python software known as Keras.

Keras is a high-level deep learning library for Python. It is designed to provide a user-friendly and intuitive interface for building and training deep learning models. Keras acts as a front-end API, allowing developers to define and configure neural networks while leveraging the computational backend engines, such as Tensor Flow or Theano.

Here are some key features and functionalities of the Keras library:

* User-Friendly API
* Multi-backend Support
* Wide Range of Neural Network Architectures
* Pre-trained Models and Transfer Learning:
* Easy Model Training and Evaluation:
* GPU Support:
* **h5py:**

\* The h5py Python module offers an interface for the binary HDF5 data format. Thanks to p5py, the top can quickly halt the vast amount of numerical data and alter it using the NumPy library. It employs common syntax for Python, NumPy, and dictionary arrays.

h5py is a Python library that provides a simple and efficient interface for working with datasets and files in the Hierarchical Data Format 5 (HDF5) format. HDF5 is a versatile data format commonly used for storing and managing large volumes of numerical data.

Here are some key features and functionalities of the h5py library:

* + HDF5 File Access
  + Dataset Handling:
  + Group Organization:
  + Attributes:
  + Compatibility with NumPy
  + Performance
* **Tensor flow**

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. TensorFlow is an end-to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models.

TensorFlow is a popular open-source library for machine learning and deep learning. It provides a comprehensive set of tools, APIs, and computational resources for building and training various types of machine learning models, especially neural networks.

Here are some key features and functionalities of TensorFlow:

* Neural Network Framework:
* Computational Graphs
* Automatic Differentiation
* GPU and TPU Support
* Distributed Computing
* Deployment Capabilities
* **Tkinter**

Tkinter is an acronym for "Tk interface". Tk was developed as a GUI extension for the Tcl scripting language by John Ousterhout. The first release was in 1991. Tkinter is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact, it's the only framework built into the Python standard library.

Tkinter is a standard Python library used for creating graphical user interfaces (GUIs). It provides a set of modules and classes that allow you to develop interactive and visually appealing desktop applications.

Here are some key features and functionalities of Tkinter:

* Cross-Platform Compatibility
* Simple and Easy-to-Use
* Widgets and Layout Management
* Event-Driven Programming
* Customization and Styling
* Integration with Other Libraries
* **NLTK**

NLTK is a toolkit build for working with NLP in Python. It provides us various text processing libraries with a lot of test datasets. A variety of tasks can be performed using NLTK such as tokenizing, parse tree visualization, etc NLTK (Natural Language Toolkit) is the go-to API for NLP (Natural Language Processing) with Python. It is a really powerful tool to pre-process text data for further analysis like with ML models for instance. It helps convert text into numbers, which the model can then easily work with.

NLTK (Natural Language Toolkit) is a Python library widely used for working with human language data and implementing natural language processing (NLP) tasks. It provides a set of tools, corpora, and resources for tasks such as tokenization, stemming, tagging, parsing, sentiment analysis, and more.

Here are some key features and functionalities of NLTK:

* Text Processing
* Part-of-Speech Tagging
* Named Entity Recognition
* Chunking and Parsing
* Sentiment Analysis:
* WordNet Integration:
* **Scipy**

SciPy is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data.

SciPy is a powerful scientific computing library for Python that provides a wide range of mathematical algorithms and functions. It builds upon NumPy, another fundamental library for numerical computing, and extends its capabilities by adding additional tools for scientific and technical computing tasks.

Here are some key features and functionalities of SciPy:

* Numerical Integration:
* Optimization and Root Finding
* Linear Algebra
* Signal and Image Processing
* Statistics

**5.2 Sample Code:**

**RemoteSceneClassification.py**

from tkinter import messagebox

from tkinter import \*

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

from tkinter.filedialog import askopenfilename

import cv2

import random

import numpy as np

from keras.utils.np\_utils import to\_categorical

from keras.layers import MaxPooling2D

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.layers import Convolution2D

from keras.models import Sequential

from sklearn.model\_selection import train\_test\_split

from keras.applications import DenseNet169

from sklearn.metrics import accuracy\_score

from keras.callbacks import ModelCheckpoint

import pickle

import os

from keras.models import load\_model

from sklearn.metrics import precision\_score

from sklearn.metrics import recall\_score

from sklearn.metrics import f1\_score

from sklearn.metrics import accuracy\_score

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix

import seaborn as sns

from keras.models import load\_model

from keras.applications import VGG19

from keras.applications import MobileNetV2

main = tkinter.Tk()

main.title("Remote Sensing Image Scene Classification Meets Deep Learning: Challenges, Methods, Benchmarks, and Opportunities")

main.geometry("1300x1200")

global filename

global classifier

global labels, X, Y, X\_train, y\_train, X\_test, y\_test, classifier

def readLabels(filename):

global labels

labels = []

for root, dirs, directory in os.walk(filename):

for j in range(len(directory)):

name = os.path.basename(root)

if name not in labels:

labels.append(name)

def getID(name):

index = 0

for i in range(len(labels)):

if labels[i] == name:

index = i

break

return index

def uploadDataset():

global filename

global labels

labels = []

filename = filedialog.askdirectory(initialdir=".")

pathlabel.config(text=filename)

text.delete('1.0', END)

text.insert(END,filename+" loaded\n\n");

readLabels(filename)

text.insert(END,"Remote Scenes found in dataset are\n\n")

for i in range(len(labels)):

text.insert(END,labels[i]+"\n")

def processDataset():

text.delete('1.0', END)

global filename, X, Y, X\_train, y\_train, X\_test, y\_test

if os.path.exists("model/X.txt.npy"):

X = np.load('model/X.txt.npy')

Y = np.load('model/Y.txt.npy')

else:

for root, dirs, directory in os.walk(filename):

for j in range(len(directory)):

name = os.path.basename(root)

if 'Thumbs.db' not in directory[j]:

img = cv2.imread(root+"/"+directory[j])

img = cv2.resize(img, (32,32))

im2arr = np.array(img)

im2arr = im2arr.reshape(32,32,3)

X.append(im2arr)

label = getID(name)

Y.append(label)

print(name+" "+str(label))

X = np.asarray(X)

Y = np.asarray(Y)

np.save('model/X.txt',X)

np.save('model/Y.txt',Y)

X = X.astype('float32')

X = X/255

text.insert(END,"Dataset Preprocessing Completed\n")

text.insert(END,"Total images found in dataset : "+str(X.shape[0])+"\n\n")

indices = np.arange(X.shape[0])

np.random.shuffle(indices)

X = X[indices]

Y = Y[indices]

Y = to\_categorical(Y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2) #split dataset into train and test

text.insert(END,"80% images are used to train VGG19 : "+str(X\_train.shape[0])+"\n")

text.insert(END,"20% images are used to train VGG19 : "+str(X\_test.shape[0])+"\n")

def trainVGG19():

text.delete('1.0', END)

global filename, X, Y, X\_train, y\_train, X\_test, y\_test, classifier, labels

vgg = VGG19(input\_shape=(X\_train.shape[1], X\_train.shape[2], X\_train.shape[3]), include\_top=False, weights="imagenet")

vgg.trainable = False

classifier = Sequential()

classifier.add(vgg)

classifier.add(Convolution2D(32, 1, 1, input\_shape = (X\_train.shape[1], X\_train.shape[2], X\_train.shape[3]), activation = 'relu'))

classifier.add(MaxPooling2D(pool\_size = (1, 1)))

classifier.add(Convolution2D(32, 1, 1, activation = 'relu'))

classifier.add(MaxPooling2D(pool\_size = (1, 1)))

classifier.add(Flatten())

classifier.add(Dense(output\_dim = 256, activation = 'relu'))

classifier.add(Dense(output\_dim = y\_train.shape[1], activation = 'softmax'))

print(classifier.summary())

classifier.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])

if os.path.exists("model/vgg\_weights.hdf5") == False:

model\_check\_point = ModelCheckpoint(filepath='model/vgg\_weights.hdf5', verbose = 1, save\_best\_only = True)

hist = classifier.fit(X\_train, y\_train, batch\_size = 32, epochs = 20, validation\_data=(X\_test, y\_test), callbacks=[model\_check\_point], verbose=1)

f = open('model/vgg\_history.pckl', 'wb')

pickle.dump(hist.history, f)

f.close()

else:

classifier = load\_model("model/vgg\_weights.hdf5")

predict = classifier.predict(X\_test)

predict = np.argmax(predict, axis=1)

testY = np.argmax(y\_test, axis=1)

p = precision\_score(testY, predict,average='macro') \* 100

r = recall\_score(testY, predict,average='macro') \* 100

f = f1\_score(testY, predict,average='macro') \* 100

a = accuracy\_score(testY,predict)\*100

text.insert(END,"VGG19 Transfer Learning Performance\n")

text.insert(END,"VGG19 Accuracy : "+str(a)+"\n")

text.insert(END,"VGG19 Precision : "+str(p)+"\n")

text.insert(END,"VGG19 Recall : "+str(r)+"\n")

text.insert(END,"VGG19 FSCORE : "+str(f)+"\n\n")

conf\_matrix = confusion\_matrix(testY, predict)

plt.figure(figsize =(6, 6))

ax = sns.heatmap(conf\_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap="viridis" ,fmt ="g");

ax.set\_ylim([0,len(labels)])

plt.title("VGG19 Confusion matrix")

plt.ylabel('True class')

plt.xlabel('Predicted class')

plt.show()

def trainMobileNet():

global filename, X, Y, X\_train, y\_train, X\_test, y\_test, classifier, labels

#creating Pretrained MobileNetV2 object as mobilenet

mobilenet = MobileNetV2(input\_shape=(X\_train.shape[1], X\_train.shape[2], X\_train.shape[3]), include\_top=False, weights="imagenet")

mobilenet.trainable = False

#creating custom CNN model object as mobilenet\_model

mobilenet\_model = Sequential()

mobilenet\_model.add(mobilenet)#now transfer learning mobilenet pretrained model to custome mobilenet\_model object

#now defining new training layers to transfer learning mobilenet object

mobilenet\_model.add(Convolution2D(32, 1, 1, input\_shape = (X\_train.shape[1], X\_train.shape[2], X\_train.shape[3]), activation = 'relu'))

mobilenet\_model.add(MaxPooling2D(pool\_size = (1, 1)))

mobilenet\_model.add(Convolution2D(32, 1, 1, activation = 'relu'))

mobilenet\_model.add(MaxPooling2D(pool\_size = (1, 1)))

mobilenet\_model.add(Flatten())

mobilenet\_model.add(Dense(output\_dim = 256, activation = 'relu')) #define output layer

mobilenet\_model.add(Dense(output\_dim = y\_train.shape[1], activation = 'softmax'))

print(mobilenet\_model.summary())

mobilenet\_model.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])

mobilenet\_model = load\_model("model/mobilenet\_weights.hdf5")

predict = mobilenet\_model.predict(X\_test)

predict = np.argmax(predict, axis=1)

testY = np.argmax(y\_test, axis=1)

p = precision\_score(testY, predict,average='macro') \* 100

r = recall\_score(testY, predict,average='macro') \* 100

f = f1\_score(testY, predict,average='macro') \* 100

a = accuracy\_score(testY,predict)\*100

text.insert(END,"MobileNetV2 Transfer Learning Performance\n")

text.insert(END,"MobileNetV2 Accuracy : "+str(a)+"\n")

text.insert(END,"MobileNetV2 Precision : "+str(p)+"\n")

text.insert(END,"MobileNetV2 Recall : "+str(r)+"\n")

text.insert(END,"MobileNetV2 FSCORE : "+str(f)+"\n\n")

conf\_matrix = confusion\_matrix(testY, predict)

plt.figure(figsize =(6, 6))

ax = sns.heatmap(conf\_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap="viridis" ,fmt ="g");

ax.set\_ylim([0,len(labels)])

plt.title("MobileNetV2 Confusion matrix")

plt.ylabel('True class')

plt.xlabel('Predicted class')

plt.show()

def graph():

f = open('model/vgg\_history.pckl', 'rb')

graph = pickle.load(f)

f.close()

vgg\_accuracy = graph['val\_accuracy']

vgg\_error = graph['val\_loss']

f = open('model/mobilenet\_history.pckl', 'rb')

graph = pickle.load(f)

f.close()

mobilenet\_accuracy = graph['val\_accuracy']

mobilenet\_error = graph['val\_loss']

plt.figure(figsize=(10,6))

plt.grid(True)

plt.xlabel('EPOCH')

plt.ylabel('Accuracy/Loss')

plt.plot(vgg\_accuracy, 'ro-', color = 'green')

plt.plot(vgg\_error, 'ro-', color = 'red')

plt.plot(mobilenet\_accuracy, 'ro-', color = 'blue')

plt.plot(mobilenet\_error, 'ro-', color = 'yellow')

plt.legend(['VGG19 Accuracy', 'VGG19 Loss', 'MobileNetV2 Accuracy', 'MobileNetV2 Loss'], loc='upper left')

plt.title('VGG19 & MobileNetV2 Training Accuracy & Loss Graph')

plt.show()

def classifyScenes():

global classifier, labels

filename = filedialog.askopenfilename(initialdir="testImages")

image = cv2.imread(filename)

img = cv2.resize(image, (32, 32))

im2arr = np.array(img)

im2arr = im2arr.reshape(1,32,32,3)

img = np.asarray(im2arr)

img = img.astype('float32')

img = img/255

preds = classifier.predict(img)

predict = np.argmax(preds)

img = cv2.imread(filename)

img = cv2.resize(img, (700,400))

cv2.putText(img, 'Remote Scene Classified as : '+labels[predict], (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,0.7, (0, 0, 255), 2)

cv2.imshow('Remote Scene Classified as : '+labels[predict], img)

cv2.waitKey(0)

def close():

main.destroy()

font = ('times', 16, 'bold')

title = Label(main, text='Remote Sensing Image Scene Classification Meets Deep Learning: Challenges, Methods, Benchmarks, and Opportunities',anchor=W, justify=CENTER)

title.config(bg='DodgerBlue3', fg='white')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 13, 'bold')

upload = Button(main, text="Upload NWPU Satellite Scenes Dataset", command=uploadDataset)

upload.place(x=50,y=100)

upload.config(font=font1)

pathlabel = Label(main)

pathlabel.config(bg='yellow4', fg='white')

pathlabel.config(font=font1)

pathlabel.place(x=50,y=150)

processButton = Button(main, text="Preprocess Dataset", command=processDataset)

processButton.place(x=50,y=200)

processButton.config(font=font1)

trainButton = Button(main, text="Train Transfer Learning VGG19", command=trainVGG19)

trainButton.place(x=50,y=250)

trainButton.config(font=font1)

mobilenetButton = Button(main, text="Train Transfer Learning MobileNetV2", command=trainMobileNet)

mobilenetButton.place(x=50,y=300)

mobilenetButton.config(font=font1)

graphButton = Button(main, text="VGG19 & MobileNetV2 Training Graph", command=graph)

graphButton.place(x=50,y=350)

graphButton.config(font=font1)

classifyButton = Button(main, text="Remote Scenes Classification", command=classifyScenes)

classifyButton.place(x=50,y=400)

classifyButton.config(font=font1)

exitButton = Button(main, text="Exit", command=close)

exitButton.place(x=50,y=450)

exitButton.config(font=font1)

font1 = ('times', 12, 'bold')

text=Text(main,height=25,width=78)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=370,y=100)

text.config(font=font1)

main.config(bg='LightPink1')

main.mainloop()

**6. TESTING:**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

## Implementation

## The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifying as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

## Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property function as a unit. The test data should be chosen such that it passed through all possible condition. Actually, testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

### System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus, the code was exhaustively checked for all possible correct data and the outcomes were also checked.

**Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus, all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example, the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

**Integration Testing**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus, the mapping of jobs with resources is done correctly by the system.

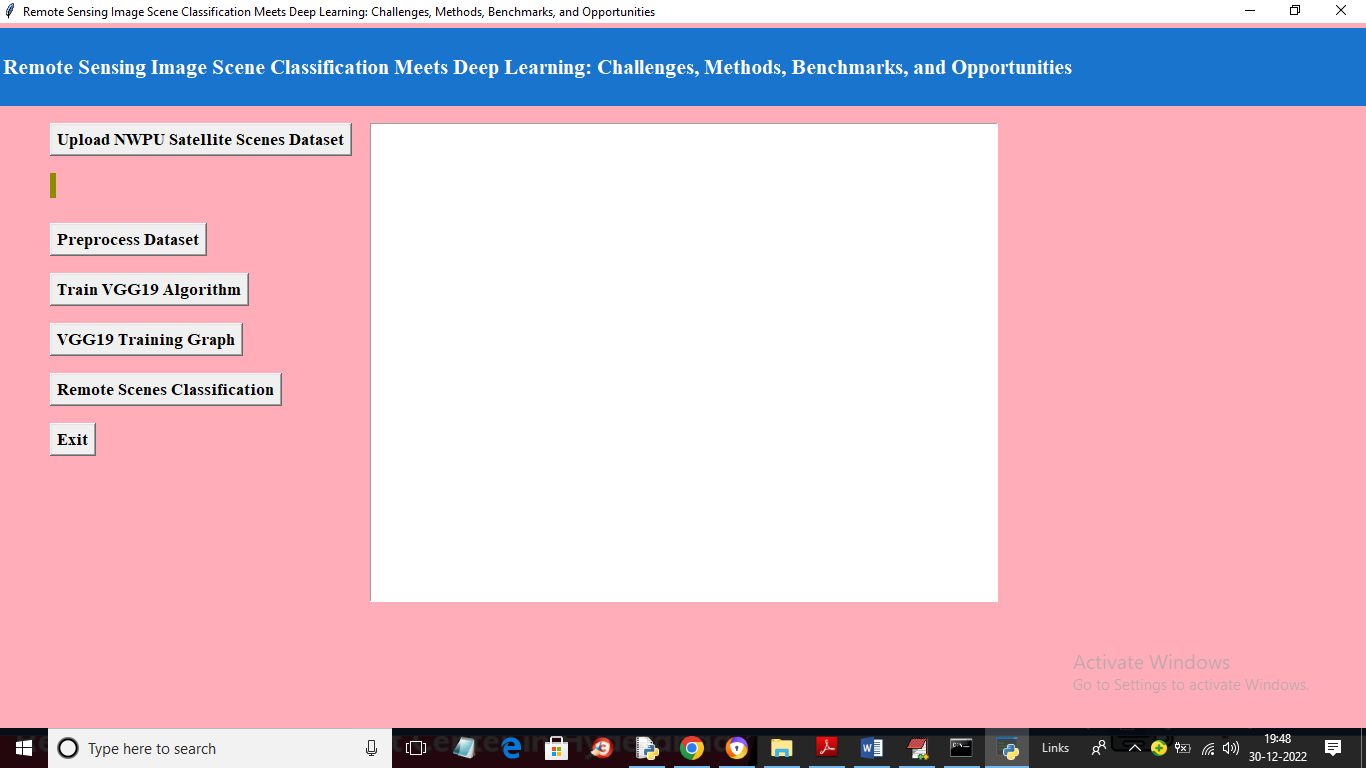
**Acceptance Testing**

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

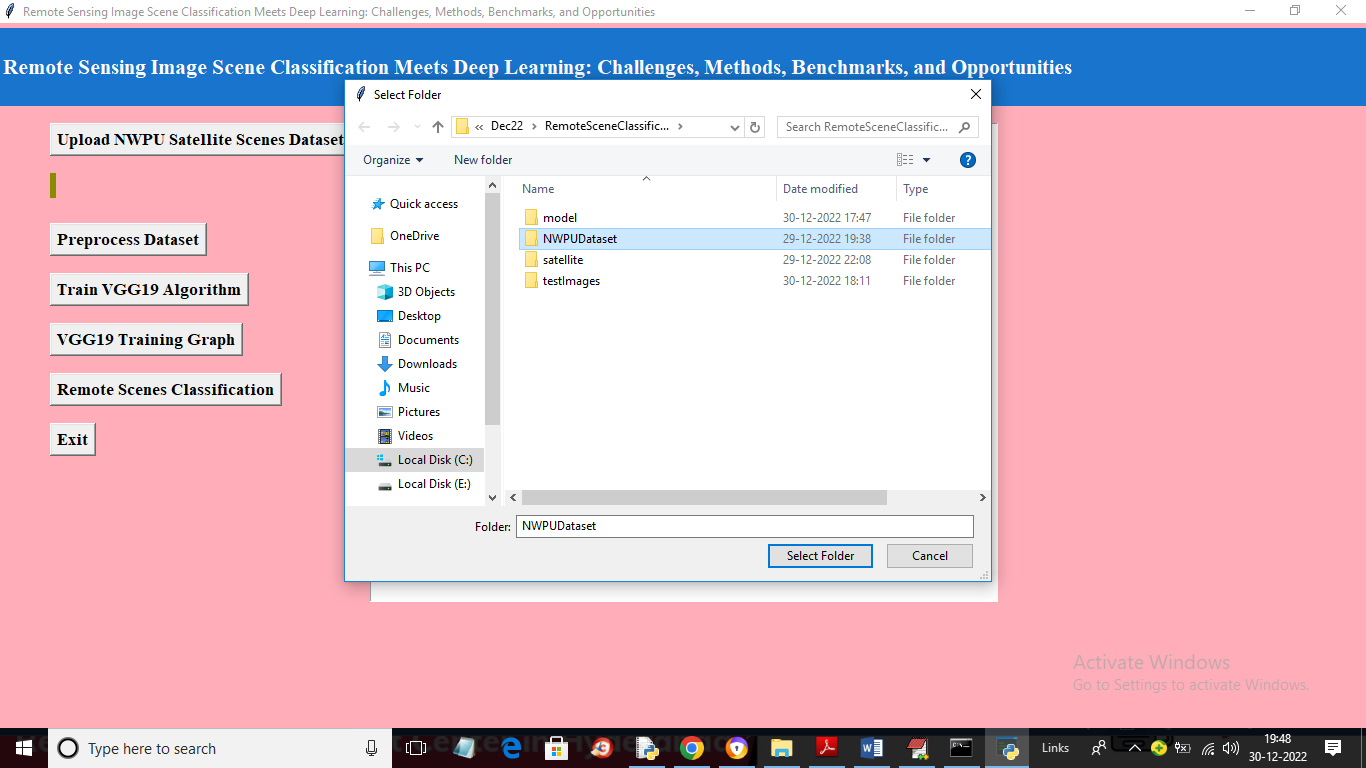
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | | **Actual** |
| 01 | Upload Dataset | Verify Dataset Uploaded or not | If Datasets May not uploaded | we cannot do any further operations | we can do further operations | | High | High |
| 02 | Preprocess Dataset | Verify Preprocess Dataset  submitted or not | If Preprocess Dataset  is not submitted | we cannot do any further operations | we can do further operations | | High | High |
| 03 | Train Transfer Learning VGG19 | Verify  Transfer Learning VGG19 trained  or not | If Train Transfer Learning VGG19  not obtained | we cannot do any further operations | we can do further operations | | High | High |
| 04 | Train Transfer Learning MobileNetV2 | Verify Transfer Learning MobileNetV2 trained or not | If Transfer Learning MobileNetV2 not trained | we cannot do any further operations | we can do further operations | | High | High |
| 05 | Training Graph | Verify Training Graph  submitted or not | If Training Graph  is not submitted | we cannot do any further operations | we can do further operations | | High | High |
| 06 | Remote Scene Classification | Verify Remote  SceneClassification obtained or not | If Remote Scene Classification  not obtained | we cannot do any further operations | we can do further operations | | High | High |

**7. SCREENSHOTS:**

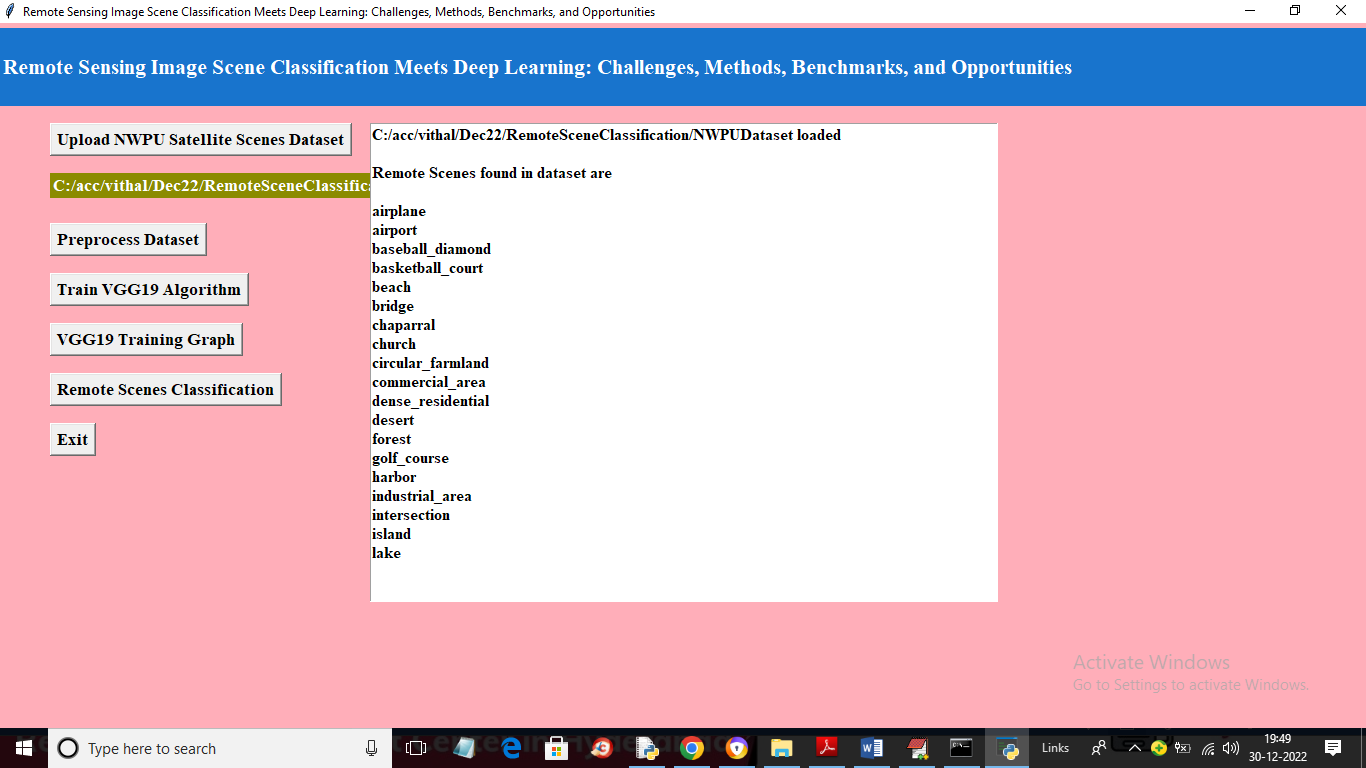
To run project double click on ‘run.bat’ file to get below screen



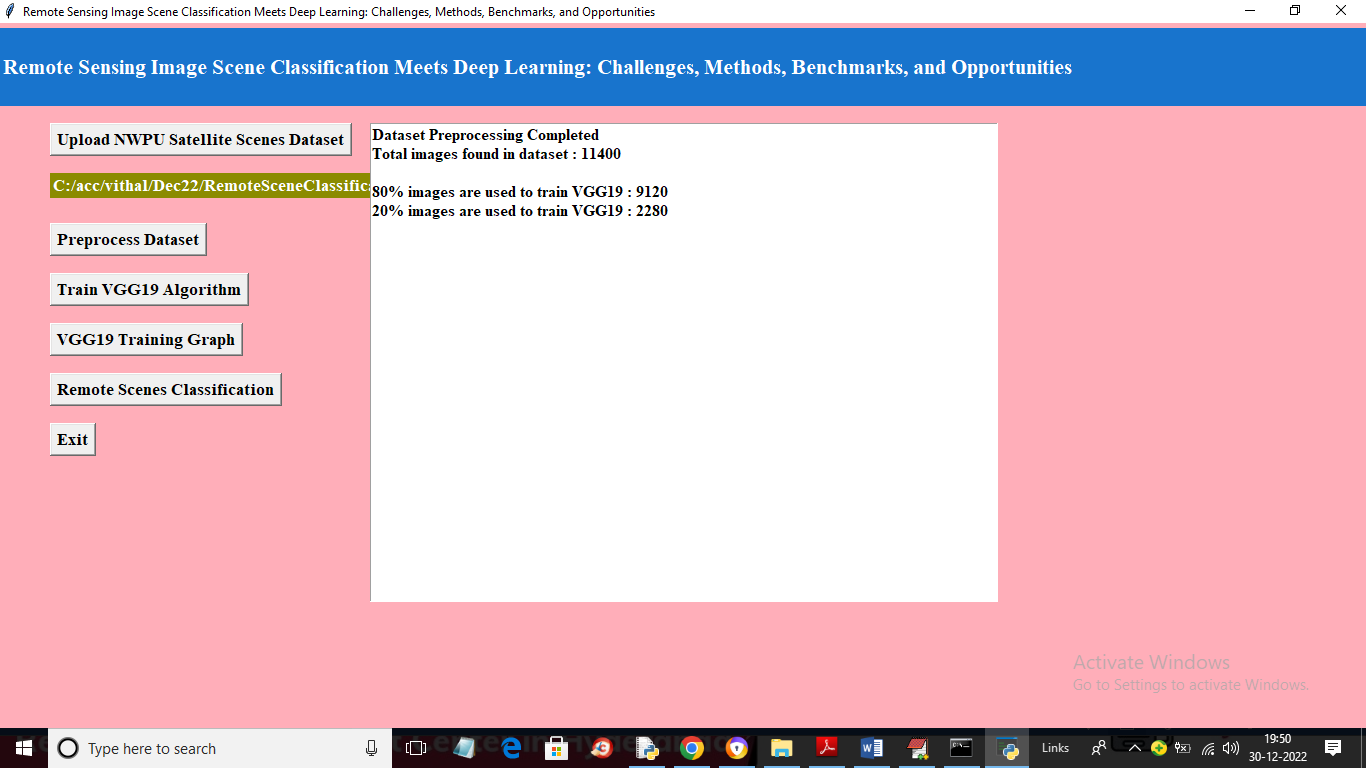
In above screen click on ‘Upload NWPU Satellite Scenes Dataset’ button to upload dataset and get below output



In above screen selecting and uploading entire ‘NWPU’ dataset folder and then click on ‘Select Folder’ button to load dataset and get below output



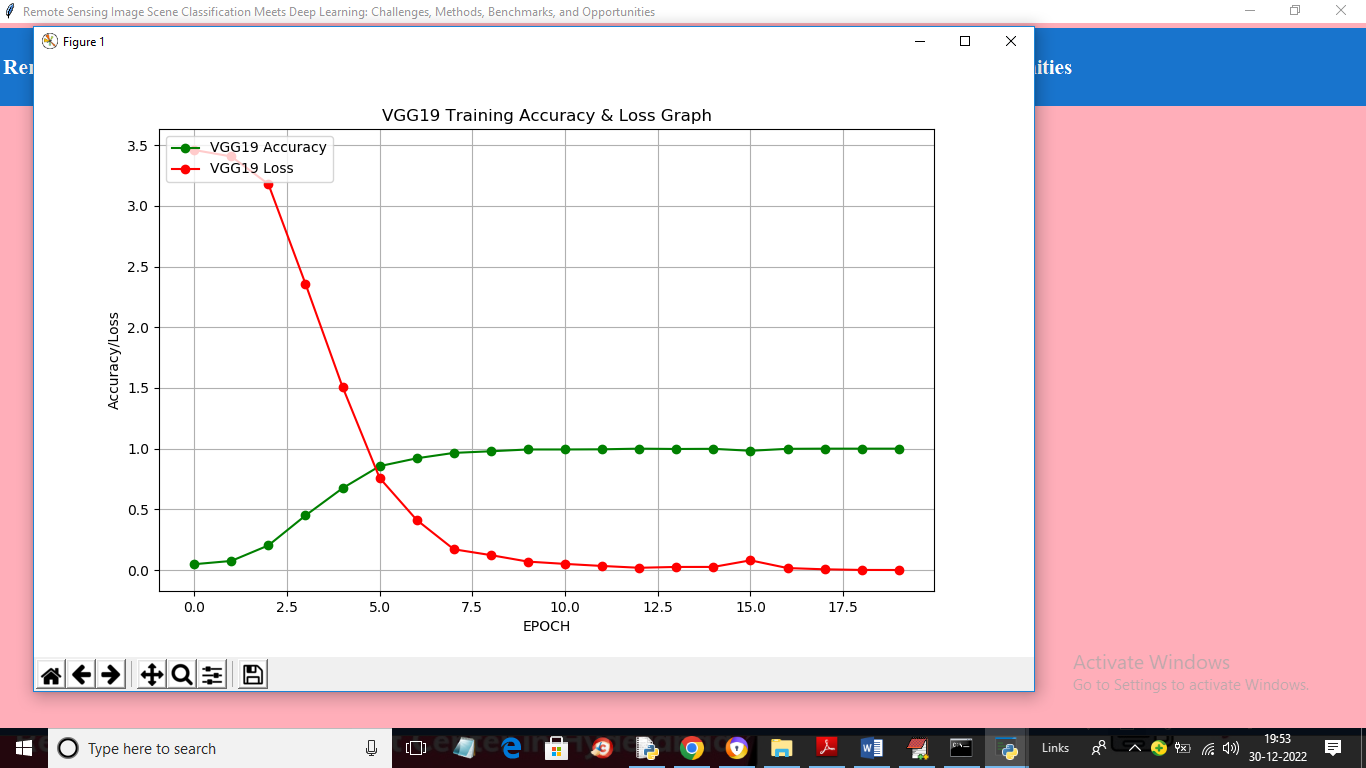
In above screen dataset loaded and we can see the names of scenes available in dataset and now click on ‘Preprocess Dataset’ button to process all images and get below output



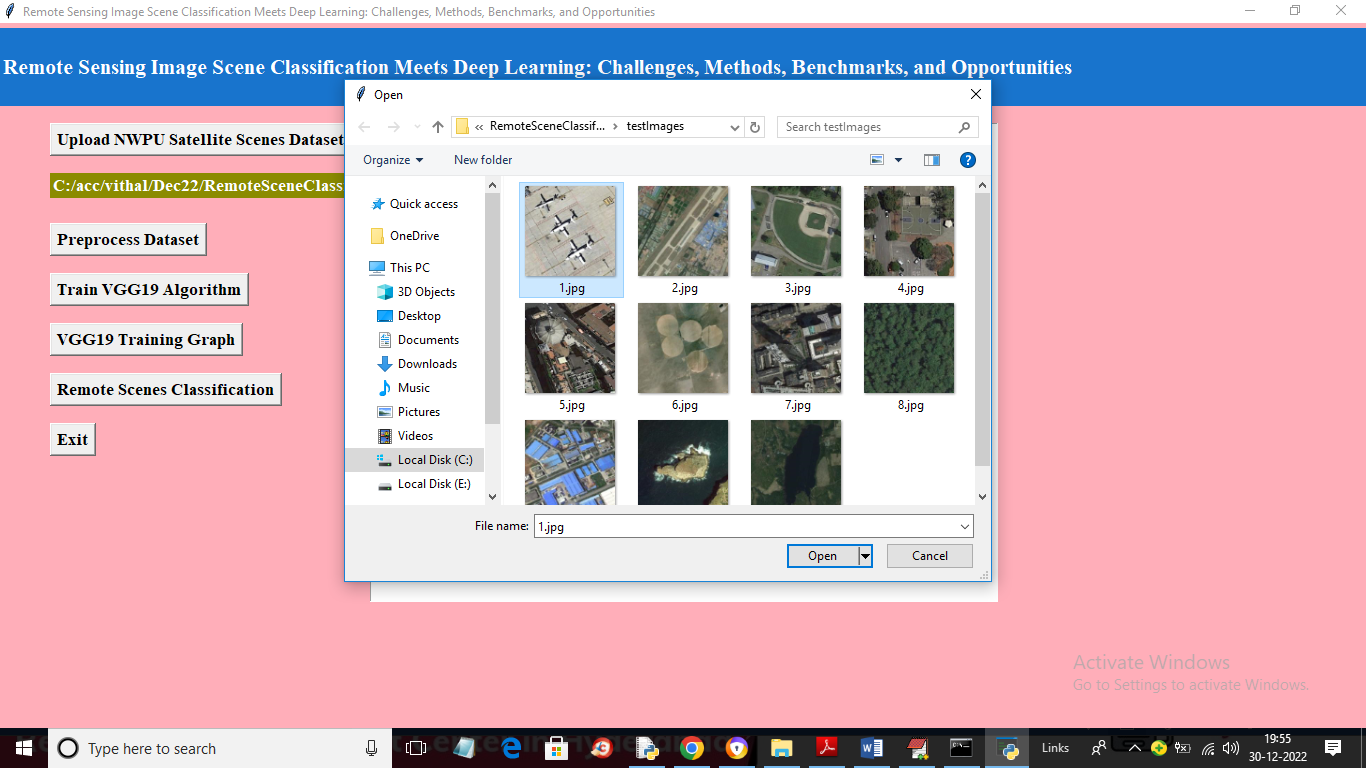
In above screen all images are processed and then application spilt them into train and test where we can see total images, training and testing images size and now click on ‘TrainVGG19 Algorithm’ button to train VGG on given image and get evaluate on test images to calculate its prediction accuracy



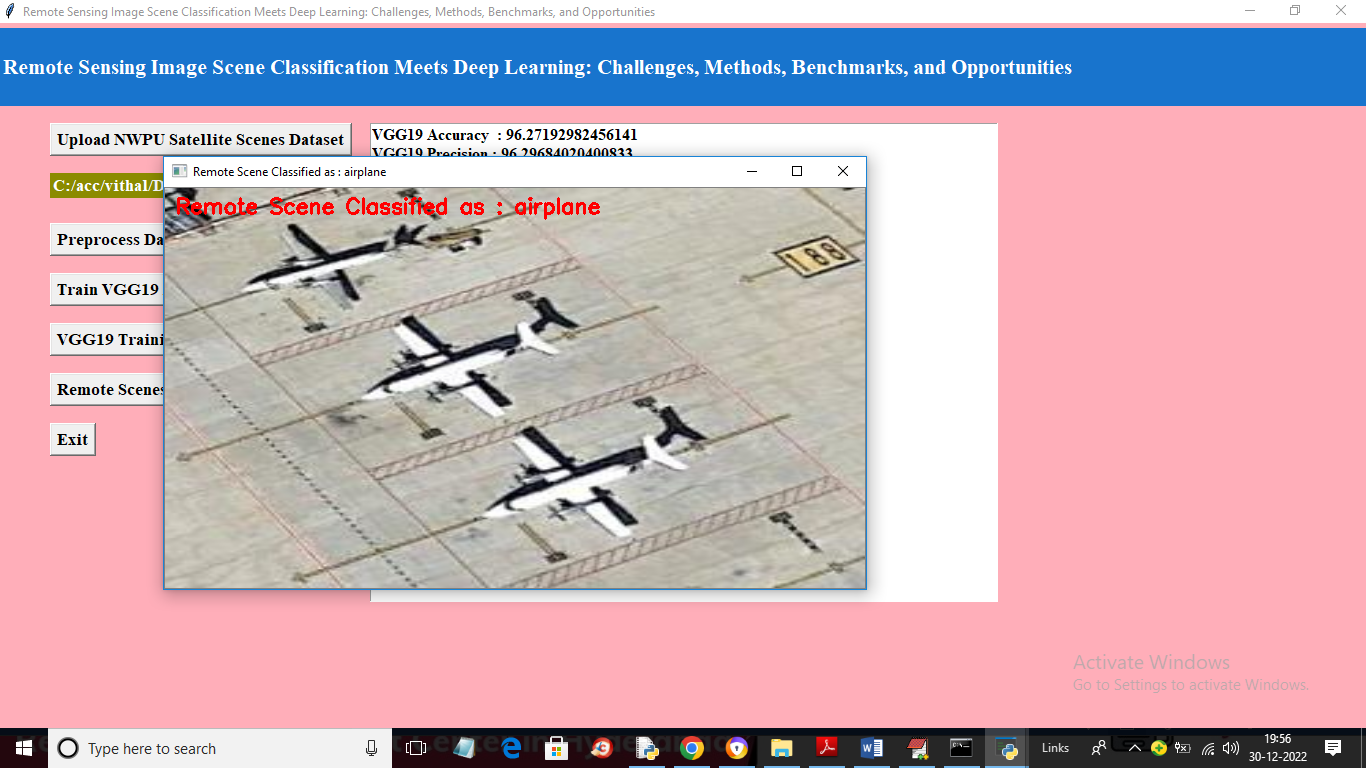
In above screen with VGG19 we got 96% accuracy and in confusion matrix graph x-axis represents Predicted Labels and y-axis represents True labels and the count in diagnol in different colours represents correct prediction count and all blue colour boxes represents incorrect prediction count which are very few and now close above graph and then click on ‘VGG19 Training Graph’ button to get below graph



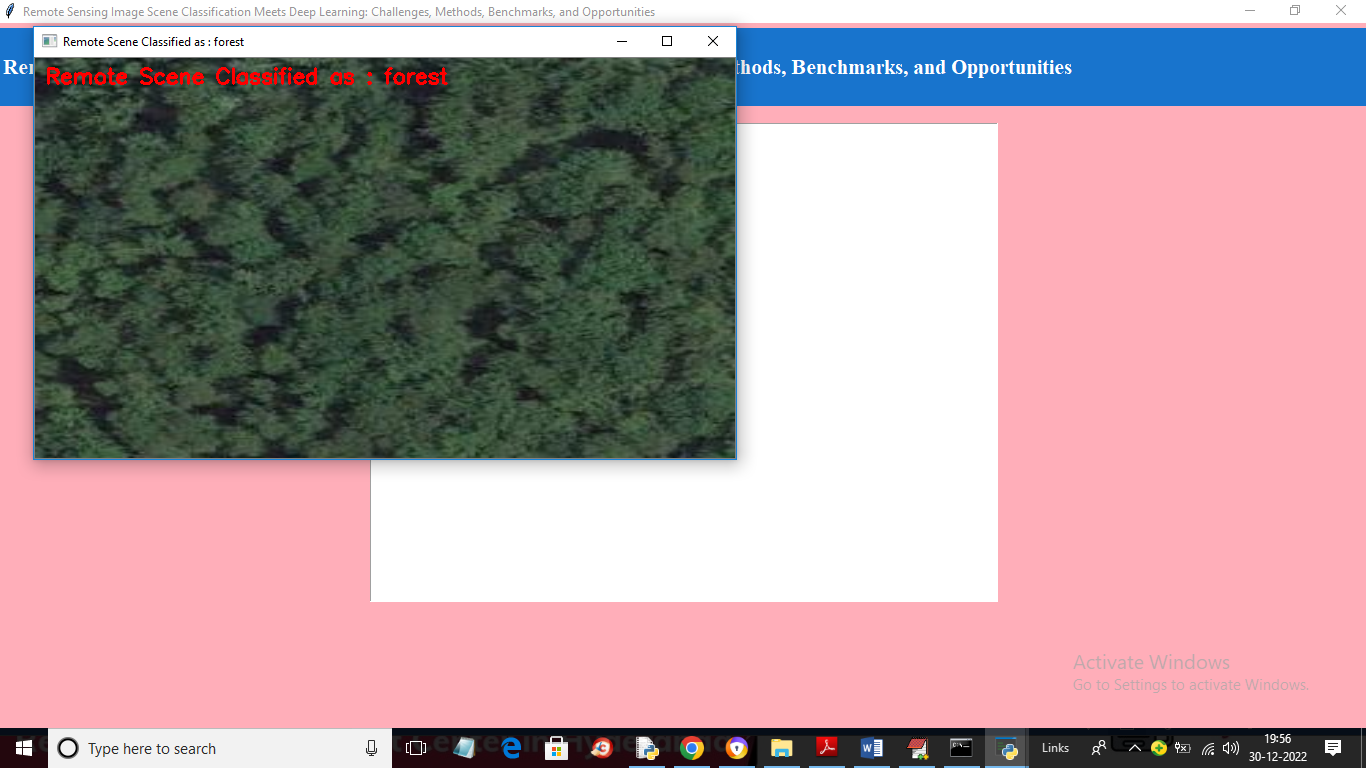
In above graph x-axis represents training epoch and y-axis represents training accuracy and loss. Green line represents accuracy and red line represents loss and with each increasing epoch accuracy got increase and reached closer to 1 and loss got decreased and reached closer to 0. Now close above graph and then click on ‘Remote Scenes Classification’ button to upload image and classify scenes

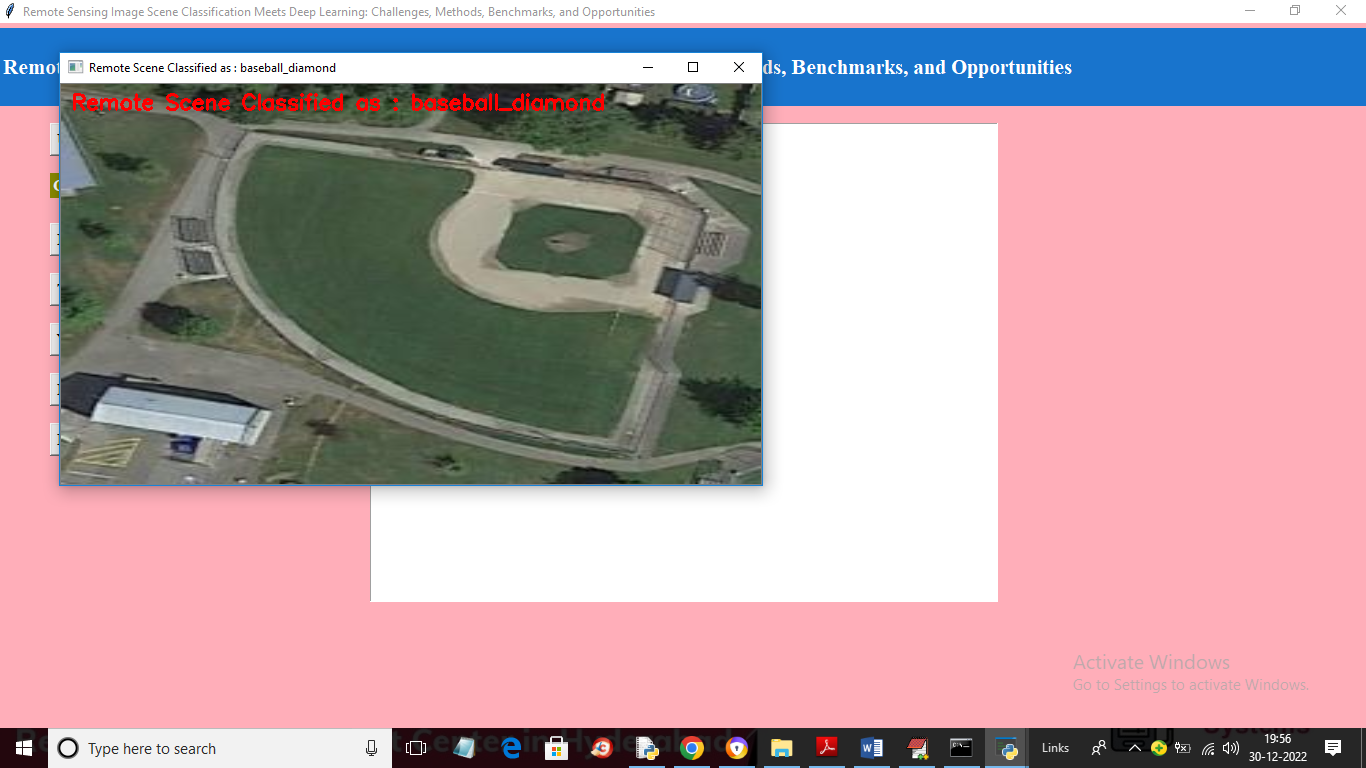


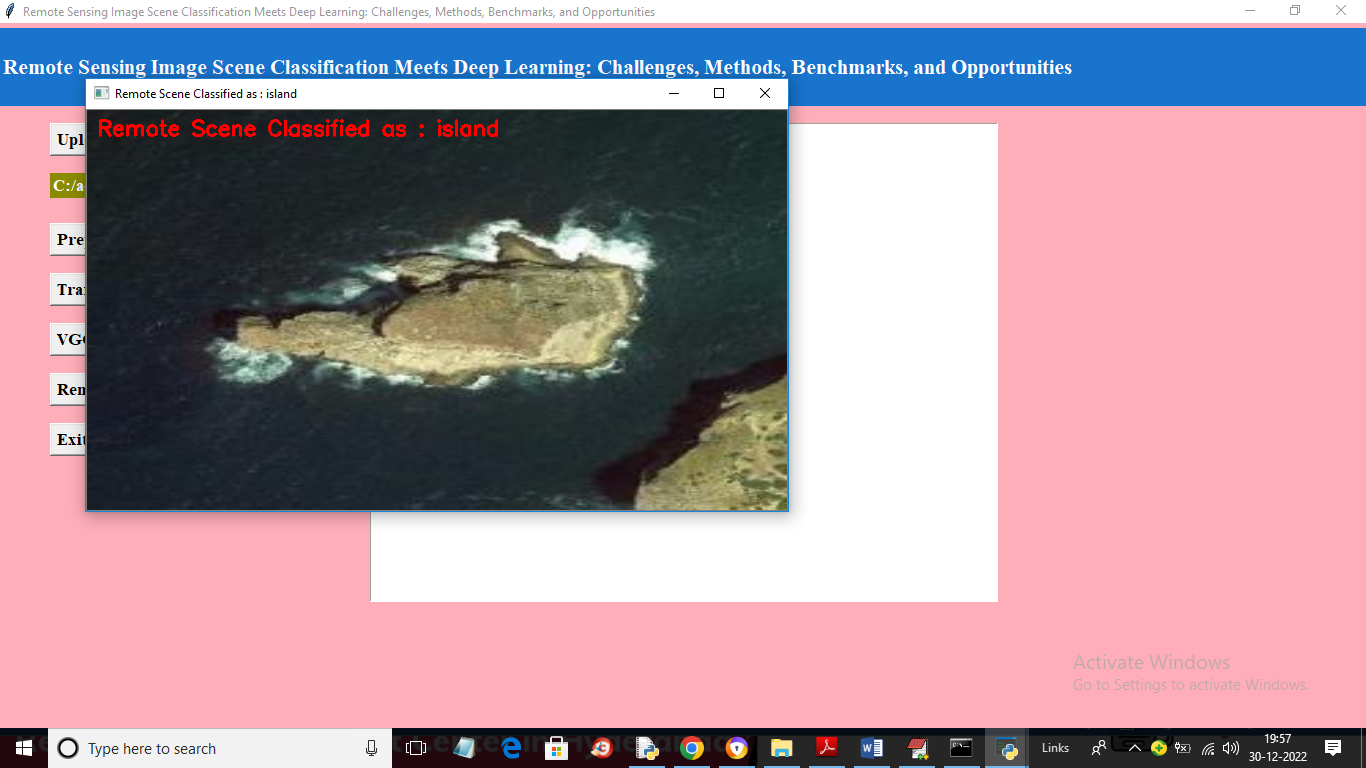
In above screen selecting and uploading ‘1.jpg’ and then click on ‘Open’ button to load image and get classification from VGG19



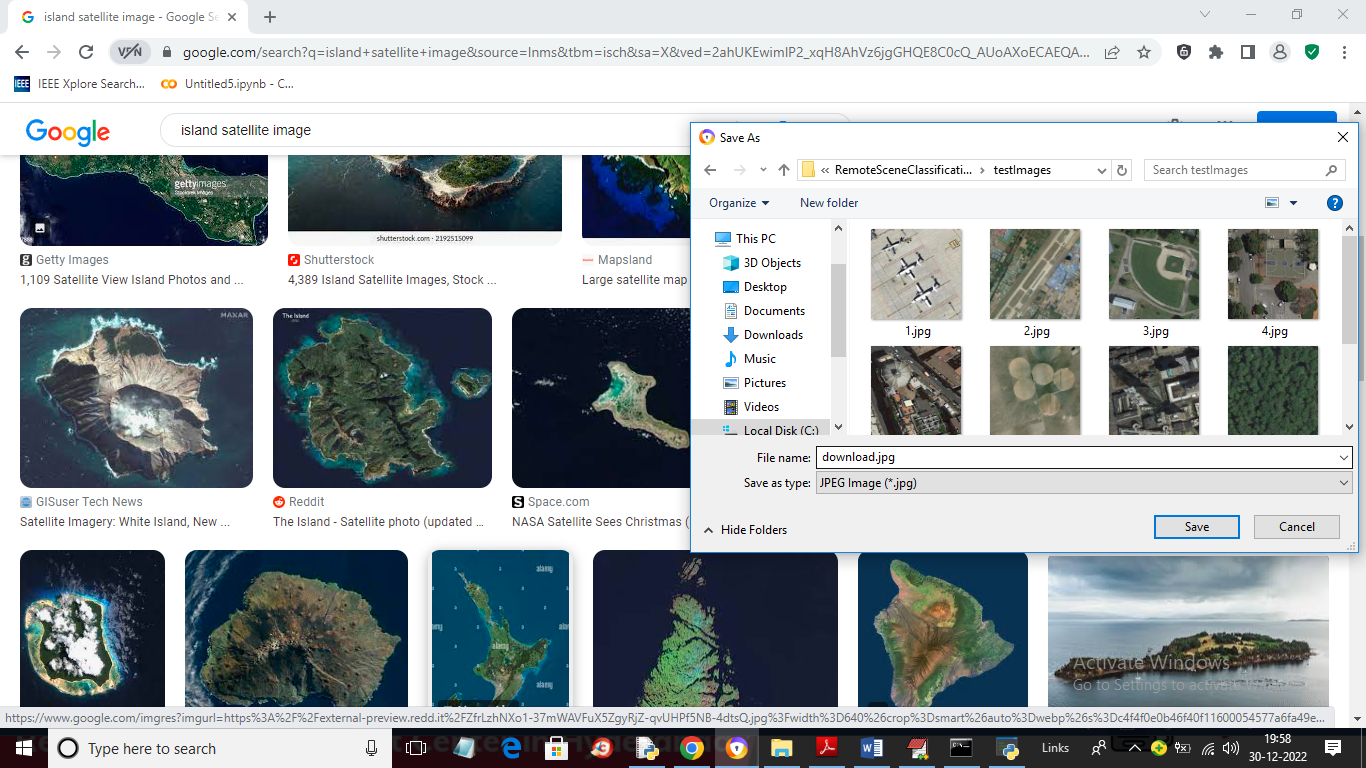
In above screen in red colour text image scenes classified as ‘airplane’ and similarly you can upload and test other images



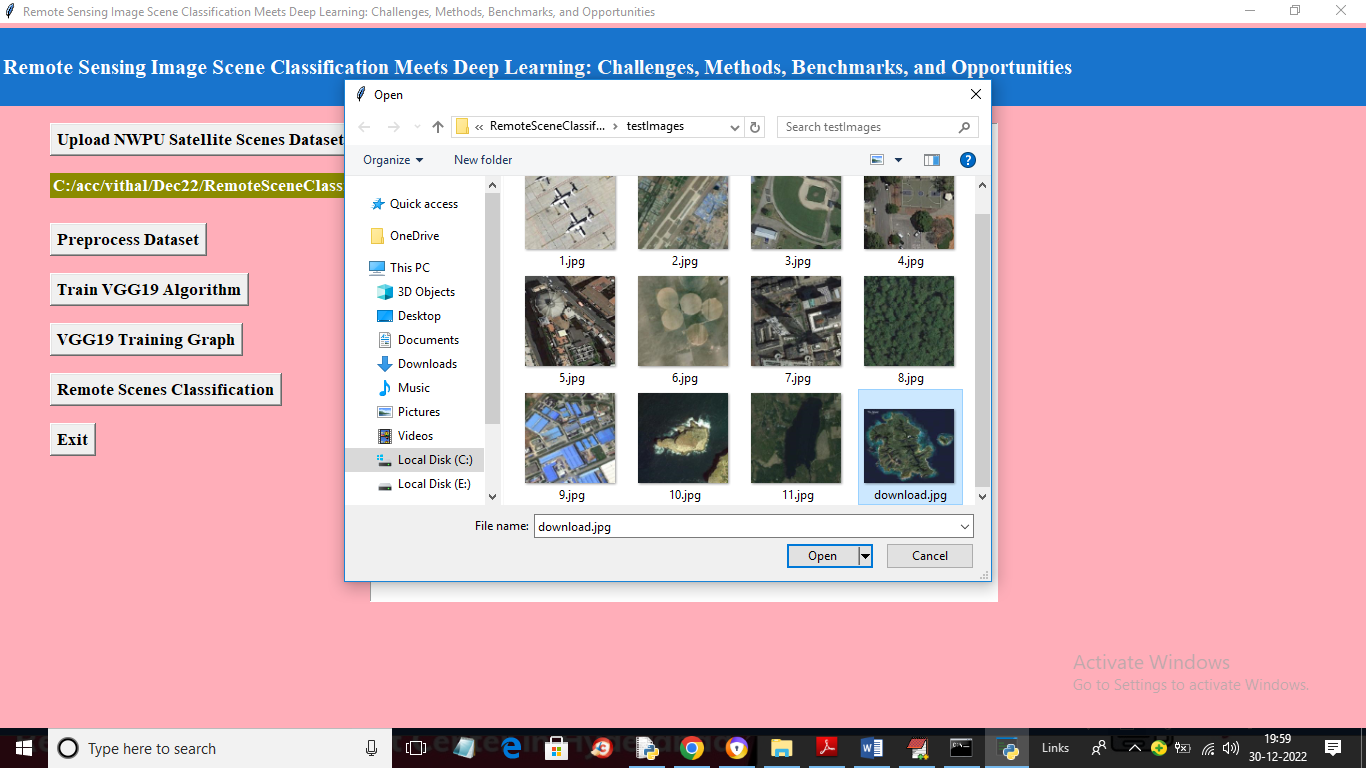




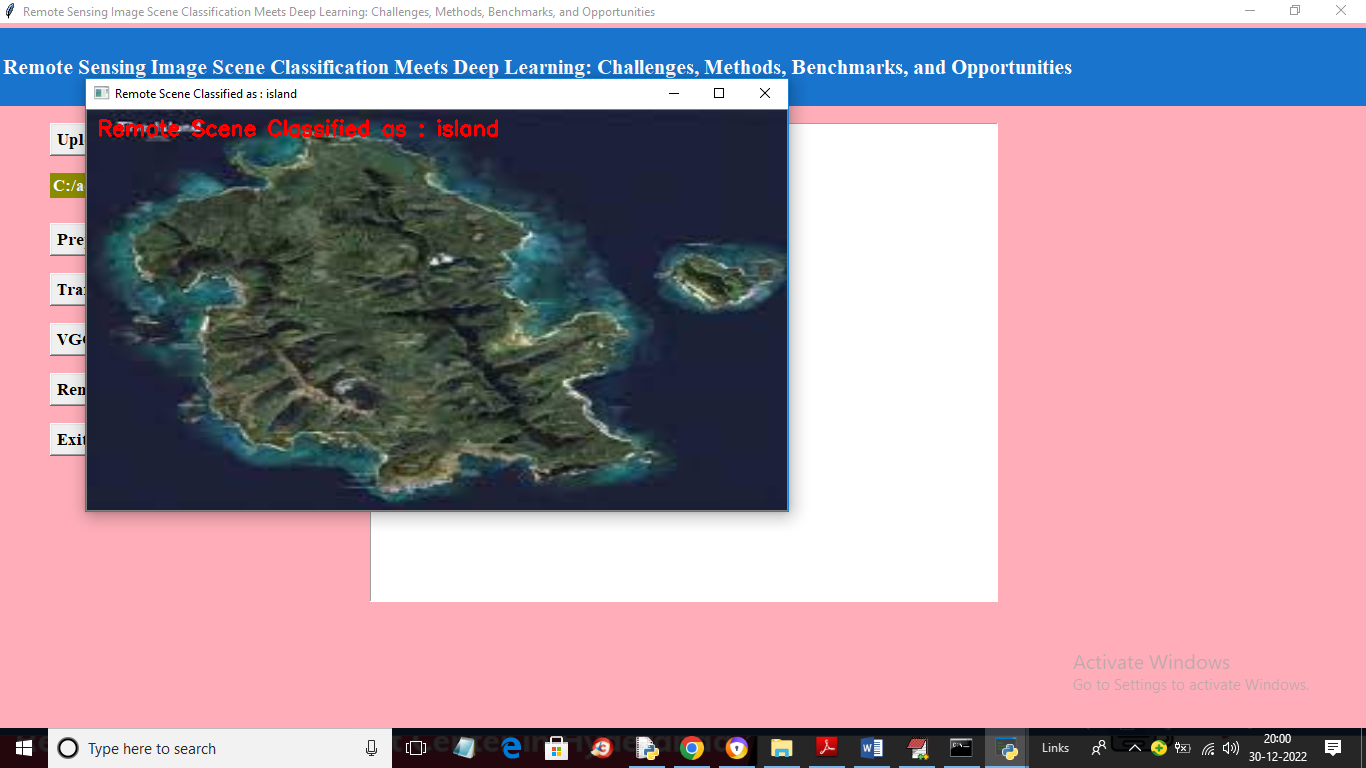
In below screen we are download one image satellite image from Google and then classifying with VGG19



In above screen I am downloading Island scene image and then in below screen uploading same image



After uploading Google ‘download.jpg’ image will get below output



So VGG19 can classify any remote image with an accuracy of 95%

**8. CONCLUSION:**

Scene classification of remote sensing images aspires to annotate them to a semantic class based on their contents. We present a concise and comprehensive survey of the literature’s feature representation methods in this view. This survey paper establishes that deep learning approaches attain superior accuracy in the scene classification of remote sensing images. Further, this paper covers the deep learning methods that have been applied, their strengths, and their shortcoming. Additionally, it covers the various deep learning architectures and available software implementation frameworks. This paper also presents evaluations from the literature on the effectiveness of deep learning approaches in scene classification with different remote sensing datasets.

In general, this work gives insights into the feature learning methods, deep learning architectures, and software frameworks that can be exploited for implementing deep learning solutions for remote sensing image scene classification. Additionally, this paper gives descriptions of remote sensing datasets that can be utilizable to evaluate the effectiveness of image semantic categorization strategies.

**9. REFERENCES:**

1. Cheng, G.; Xie, X.; Han, J.; Guo, L.; Xia, G.-S. Remote Sensing Image Scene Classification Meets Deep Learning: Challenges, Methods, Benchmarks, and Opportunities. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. **2020**, 13, 3735–3756. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Remote+Sensing+Image+Scene+Classification+Meets+Deep+Learning:+Challenges,+Methods,+Benchmarks,+and+Opportunities&author=Cheng,+G.&author=Xie,+X.&author=Han,+J.&author=Guo,+L.&author=Xia,+G.-S.&publication_year=2020&journal=IEEE+J.+Sel.+Top.+Appl.+Earth+Obs.+Remote+Sens.&volume=13&pages=3735%E2%80%933756&doi=10.1109/JSTARS.2020.3005403)] [**[CrossRef](https://doi.org/10.1109/JSTARS.2020.3005403" \t "_blank)**]
2. Cheng, G.; Han, J.; Lu, X. Remote Sensing Image Classification: Benchmark and State of the Art. Proc. IEEE **2017**, 105, 1865–1883. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Remote+Sensing+Image+Classification:+Benchmark+and+State+of+the+Art&author=Cheng,+G.&author=Han,+J.&author=Lu,+X.&publication_year=2017&journal=Proc.+IEEE&volume=105&pages=1865%E2%80%931883&doi=10.1109/JPROC.2017.2675998)] [**[CrossRef](https://doi.org/10.1109/JPROC.2017.2675998" \t "_blank)**]
3. Xia, G.-S.; Hu, J.; Hu, F.; Shi, B.; Bai, X.; Zhong, Y.; Zhang, L.; Lu, X. AID: A Benchmark Data Set for Performance Evaluation of Aerial Scene Classification. IEEE Trans. Geosci. Remote Sens. **2017**, 55, 3965–3981. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=AID:+A+Benchmark+Data+Set+for+Performance+Evaluation+of+Aerial+Scene+Classification&author=Xia,+G.-S.&author=Hu,+J.&author=Hu,+F.&author=Shi,+B.&author=Bai,+X.&author=Zhong,+Y.&author=Zhang,+L.&author=Lu,+X.&publication_year=2017&journal=IEEE+Trans.+Geosci.+Remote+Sens.&volume=55&pages=3965%E2%80%933981&doi=10.1109/TGRS.2017.2685945)] [**[CrossRef](https://doi.org/10.1109/TGRS.2017.2685945" \t "_blank)**]
4. Zhou, Q.; Zheng, B.; Zhu, W.; Latecki, L.J. Multi-scale context for scene labeling via flexible segmentation graph. Pattern Recognit. **2016**, 59, 312–324. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Multi-scale+context+for+scene+labeling+via+flexible+segmentation+graph&author=Zhou,+Q.&author=Zheng,+B.&author=Zhu,+W.&author=Latecki,+L.J.&publication_year=2016&journal=Pattern+Recognit.&volume=59&pages=312%E2%80%93324&doi=10.1016/j.patcog.2016.03.023)] [**[CrossRef](https://doi.org/10.1016/j.patcog.2016.03.023" \t "_blank)**]
5. Zhou, B.; Khosla, A.; Lapedriza, A.; Oliva, A.; Torralba, A. Learning deep features for discriminative localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2016; pp. 2921–2929. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Learning+deep+features+for+discriminative+localization&conference=Proceedings+of+the+IEEE+Conference+on+Computer+Vision+and+Pattern+Recognition&author=Zhou,+B.&author=Khosla,+A.&author=Lapedriza,+A.&author=Oliva,+A.&author=Torralba,+A.&publication_year=2016&pages=2921%E2%80%932929)]
6. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2016; pp. 770–778. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Deep+residual+learning+for+image+recognition&conference=Proceedings+of+the+IEEE+Conference+on+Computer+Vision+and+Pattern+Recognition&author=He,+K.&author=Zhang,+X.&author=Ren,+S.&author=Sun,+J.&publication_year=2016&pages=770%E2%80%93778)]
7. Bu, S.; Han, P.; Liu, Z.; Han, J. Scene parsing using inference embedded deep networks. Pattern Recognit. **2016**, 59, 188–198. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Scene+parsing+using+inference+embedded+deep+networks&author=Bu,+S.&author=Han,+P.&author=Liu,+Z.&author=Han,+J.&publication_year=2016&journal=Pattern+Recognit.&volume=59&pages=188%E2%80%93198&doi=10.1016/j.patcog.2016.01.027)] [**[CrossRef](https://doi.org/10.1016/j.patcog.2016.01.027" \t "_blank)**]
8. Pohlen, T.; Hermans, A.; Mathias, M.; Leibe, B. Full-resolution residual networks for semantic segmentation in street scenes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4151–4160. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Full-resolution+residual+networks+for+semantic+segmentation+in+street+scenes&conference=Proceedings+of+the+IEEE+Conference+on+Computer+Vision+and+Pattern+Recognition&author=Pohlen,+T.&author=Hermans,+A.&author=Mathias,+M.&author=Leibe,+B.&publication_year=2017&pages=4151%E2%80%934160)]
9. Tombe, R.; Viriri, S. Adaptive Deep Co-Occurrence Feature Learning Based on Classifier-Fusion for Remote Sensing Scene Classification. IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens. **2020**, 14, 155–164. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Adaptive+Deep+Co-Occurrence+Feature+Learning+Based+on+Classifier-Fusion+for+Remote+Sensing+Scene+Classification&author=Tombe,+R.&author=Viriri,+S.&publication_year=2020&journal=IEEE+J.+Sel.+Top.+Appl.+Earth+Obs.+Remote.+Sens.&volume=14&pages=155%E2%80%93164&doi=10.1109/JSTARS.2020.3044264)] [**[CrossRef](https://doi.org/10.1109/JSTARS.2020.3044264" \t "_blank)**]
10. Boualleg, Y.; Farah, M.; Farah, I.R. Remote Sensing Scene Classification Using Convolutional Features and Deep Forest Classifier. IEEE Geosci. Remote Sens. Lett. **2019**, 16, 1944–1948. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Remote+Sensing+Scene+Classification+Using+Convolutional+Features+and+Deep+Forest+Classifier&author=Boualleg,+Y.&author=Farah,+M.&author=Farah,+I.R.&publication_year=2019&journal=IEEE+Geosci.+Remote+Sens.+Lett.&volume=16&pages=1944%E2%80%931948&doi=10.1109/LGRS.2019.2911855)] [**[CrossRef](https://doi.org/10.1109/LGRS.2019.2911855" \t "_blank)**]
11. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. Nature **2015**, 521, 436–444. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Deep+learning&author=LeCun,+Y.&author=Bengio,+Y.&author=Hinton,+G.&publication_year=2015&journal=Nature&volume=521&pages=436%E2%80%93444&doi=10.1038/nature14539)] [**[CrossRef](https://doi.org/10.1038/nature14539" \t "_blank)**]
12. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. arXiv **2014**, arXiv:1409.1556. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Very+deep+convolutional+networks+for+large-scale+image+recognition&author=Simonyan,+K.&author=Zisserman,+A.&publication_year=2014&journal=arXiv)]
13. Hu, F.; Xia, G.-S.; Hu, J.; Zhang, L. Transferring Deep Convolutional Neural Networks for the Scene Classification of High-Resolution Remote Sensing Imagery. Remote Sens. **2015**, 7, 14680–14707. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Transferring+Deep+Convolutional+Neural+Networks+for+the+Scene+Classification+of+High-Resolution+Remote+Sensing+Imagery&author=Hu,+F.&author=Xia,+G.-S.&author=Hu,+J.&author=Zhang,+L.&publication_year=2015&journal=Remote+Sens.&volume=7&pages=14680%E2%80%9314707&doi=10.3390/rs71114680)] [**[CrossRef](https://doi.org/10.3390/rs71114680" \t "_blank)**]
14. Li, E.; Xia, J.; Du, P.; Lin, C.; Samat, A. Integrating Multilayer Features of Convolutional Neural Networks for Remote Sensing Scene Classification. IEEE Trans. Geosci. Remote Sens. **2017**, 55, 5653–5665. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Integrating+Multilayer+Features+of+Convolutional+Neural+Networks+for+Remote+Sensing+Scene+Classification&author=Li,+E.&author=Xia,+J.&author=Du,+P.&author=Lin,+C.&author=Samat,+A.&publication_year=2017&journal=IEEE+Trans.+Geosci.+Remote+Sens.&volume=55&pages=5653%E2%80%935665&doi=10.1109/TGRS.2017.2711275)] [**[CrossRef](https://doi.org/10.1109/TGRS.2017.2711275" \t "_blank)**]
15. Yuan, B.; Han, L.; Gu, X.; Yan, H. Multi-deep features fusion for high-resolution remote sensing image scene classification. Neural Comput. Appl. **2020**, 33, 2047–2063. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Multi-deep+features+fusion+for+high-resolution+remote+sensing+image+scene+classification&author=Yuan,+B.&author=Han,+L.&author=Gu,+X.&author=Yan,+H.&publication_year=2020&journal=Neural+Comput.+Appl.&volume=33&pages=2047%E2%80%932063&doi=10.1007/s00521-020-05071-7)] [**[CrossRef](https://doi.org/10.1007/s00521-020-05071-7" \t "_blank)**]
16. Liu, N.; Wan, L.; Zhang, Y.; Zhou, T.; Huo, H.; Fang, T. Exploiting Convolutional Neural Networks With Deeply Local Description for Remote Sensing Image Classification. IEEE Access **2018**, 6, 11215–11228. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Exploiting+Convolutional+Neural+Networks+With+Deeply+Local+Description+for+Remote+Sensing+Image+Classification&author=Liu,+N.&author=Wan,+L.&author=Zhang,+Y.&author=Zhou,+T.&author=Huo,+H.&author=Fang,+T.&publication_year=2018&journal=IEEE+Access&volume=6&pages=11215%E2%80%9311228&doi=10.1109/ACCESS.2018.2798799)] [**[CrossRef](https://doi.org/10.1109/ACCESS.2018.2798799" \t "_blank)**]
17. Xu, K.; Huang, H.; Deng, P.; Shi, G. Two-stream feature aggregation deep neural network for scene classification of remote sensing images. Inf. Sci. **2020**, 539, 250–268. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Two-stream+feature+aggregation+deep+neural+network+for+scene+classification+of+remote+sensing+images&author=Xu,+K.&author=Huang,+H.&author=Deng,+P.&author=Shi,+G.&publication_year=2020&journal=Inf.+Sci.&volume=539&pages=250%E2%80%93268&doi=10.1016/j.ins.2020.06.011)] [**[CrossRef](https://doi.org/10.1016/j.ins.2020.06.011" \t "_blank)**]
18. Bazi, Y.; Al Rahhal, M.M.; Alhichri, H.; Alajlan, N. Simple Yet Effective Fine-Tuning of Deep CNNs Using an Auxiliary Classification Loss for Remote Sensing Scene Classification. Remote Sens. **2019**, 11, 2908. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Simple+Yet+Effective+Fine-Tuning+of+Deep+CNNs+Using+an+Auxiliary+Classification+Loss+for+Remote+Sensing+Scene+Classification&author=Bazi,+Y.&author=Al+Rahhal,+M.M.&author=Alhichri,+H.&author=Alajlan,+N.&publication_year=2019&journal=Remote+Sens.&volume=11&pages=2908&doi=10.3390/rs11242908)] [**[CrossRef](https://doi.org/10.3390/rs11242908" \t "_blank)**]
19. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems. Available online: [**https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html**](https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html) (accessed on 30 December 2022).
20. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical Image Computing and Computer-Assisted Intervention; Springer: Berlin/Heidelberg, Germany, 2015; pp. 234–241. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=U-net:+Convolutional+networks+for+biomedical+image+segmentation&author=Ronneberger,+O.&author=Fischer,+P.&author=Brox,+T.&publication_year=2015&pages=234%E2%80%93241)]
21. Jing, W.; Zhang, M.; Tian, D. Improved U-Net model for remote sensing image classification method based on distributed storage. J. Real-Time Image Process. **2020**, 18, 1607–1619. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Improved+U-Net+model+for+remote+sensing+image+classification+method+based+on+distributed+storage&author=Jing,+W.&author=Zhang,+M.&author=Tian,+D.&publication_year=2020&journal=J.+Real-Time+Image+Process.&volume=18&pages=1607%E2%80%931619&doi=10.1007/s11554-020-01028-0)] [**[CrossRef](https://doi.org/10.1007/s11554-020-01028-0" \t "_blank)**]
22. Badrinarayanan, V.; Kendall, A.; Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **2017**, 39, 2481–2495. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Segnet:+A+deep+convolutional+encoder-decoder+architecture+for+image+segmentation&author=Badrinarayanan,+V.&author=Kendall,+A.&author=Cipolla,+R.&publication_year=2017&journal=IEEE+Trans.+Pattern+Anal.+Mach.+Intell.&volume=39&pages=2481%E2%80%932495&doi=10.1109/TPAMI.2016.2644615)] [**[CrossRef](https://doi.org/10.1109/TPAMI.2016.2644615" \t "_blank)**]
23. Yi, Y.; Zhang, Z.; Zhang, W.; Zhang, C.; Li, W.; Zhao, T. Semantic Segmentation of Urban Buildings from VHR Remote Sensing Imagery Using a Deep Convolutional Neural Network. Remote Sens. **2019**, 11, 1774. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Semantic+Segmentation+of+Urban+Buildings+from+VHR+Remote+Sensing+Imagery+Using+a+Deep+Convolutional+Neural+Network&author=Yi,+Y.&author=Zhang,+Z.&author=Zhang,+W.&author=Zhang,+C.&author=Li,+W.&author=Zhao,+T.&publication_year=2019&journal=Remote+Sens.&volume=11&pages=1774&doi=10.3390/rs11151774)] [**[CrossRef](https://doi.org/10.3390/rs11151774" \t "_blank)**]
24. Xu, K.; Deng, P.; Huang, H. Vision Transformer: An Excellent Teacher for Guiding Small Networks in Remote Sensing Image Scene Classification. IEEE Trans. Geosci. Remote Sens. **2022**, 60, 1–15. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Vision+Transformer:+An+Excellent+Teacher+for+Guiding+Small+Networks+in+Remote+Sensing+Image+Scene+Classification&author=Xu,+K.&author=Deng,+P.&author=Huang,+H.&publication_year=2022&journal=IEEE+Trans.+Geosci.+Remote+Sens.&volume=60&pages=1%E2%80%9315&doi=10.1109/TGRS.2022.3152566)] [**[CrossRef](https://doi.org/10.1109/TGRS.2022.3152566" \t "_blank)**]
25. Wang, X.; Zhu, J.; Yan, Z.; Zhang, Z.; Zhang, Y.; Chen, Y.; Li, H. LaST: Label-Free Self-Distillation Contrastive Learning With Transformer Architecture for Remote Sensing Image Scene Classification. IEEE Geosci. Remote Sens. Lett. **2022**, 19, 1–5. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=LaST:+Label-Free+Self-Distillation+Contrastive+Learning+With+Transformer+Architecture+for+Remote+Sensing+Image+Scene+Classification&author=Wang,+X.&author=Zhu,+J.&author=Yan,+Z.&author=Zhang,+Z.&author=Zhang,+Y.&author=Chen,+Y.&author=Li,+H.&publication_year=2022&journal=IEEE+Geosci.+Remote+Sens.+Lett.&volume=19&pages=1%E2%80%935&doi=10.1109/LGRS.2022.3185088)] [**[CrossRef](https://doi.org/10.1109/LGRS.2022.3185088" \t "_blank)**]
26. Deng, P.; Xu, K.; Huang, H. When CNNs Meet Vision Transformer: A Joint Framework for Remote Sensing Scene Classification. IEEE Geosci. Remote Sens. Lett. **2021**, 19, 1–5. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=When+CNNs+Meet+Vision+Transformer:+A+Joint+Framework+for+Remote+Sensing+Scene+Classification&author=Deng,+P.&author=Xu,+K.&author=Huang,+H.&publication_year=2021&journal=IEEE+Geosci.+Remote+Sens.+Lett.&volume=19&pages=1%E2%80%935&doi=10.1109/LGRS.2021.3109061)] [**[CrossRef](https://doi.org/10.1109/LGRS.2021.3109061" \t "_blank)**]
27. Ge, Y.; Zhang, X.; Atkinson, P.M.; Stein, A.; Li, L. Geoscience-aware deep learning: A new paradigm for remote sensing. Sci. Remote Sens. **2022**, 5, 100047. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Geoscience-aware+deep+learning:+A+new+paradigm+for+remote+sensing&author=Ge,+Y.&author=Zhang,+X.&author=Atkinson,+P.M.&author=Stein,+A.&author=Li,+L.&publication_year=2022&journal=Sci.+Remote+Sens.&volume=5&pages=100047&doi=10.1016/j.srs.2022.100047)] [**[CrossRef](https://doi.org/10.1016/j.srs.2022.100047" \t "_blank)**]
28. Ayush, K.; Uzkent, B.; Meng, C.; Tanmay, K.; Burke, M.; Lobell, D.; Ermon, S. Geography-aware self-supervised learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 10181–10190. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Geography-aware+self-supervised+learning&conference=Proceedings+of+the+IEEE/CVF+International+Conference+on+Computer+Vision&author=Ayush,+K.&author=Uzkent,+B.&author=Meng,+C.&author=Tanmay,+K.&author=Burke,+M.&author=Lobell,+D.&author=Ermon,+S.&publication_year=2021&pages=10181%E2%80%9310190)]
29. Ojala, T.; Pietikainen, M.; Maenpaa, T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. **2002**, 24, 971–987. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Multiresolution+gray-scale+and+rotation+invariant+texture+classification+with+local+binary+patterns&author=Ojala,+T.&author=Pietikainen,+M.&author=Maenpaa,+T.&publication_year=2002&journal=IEEE+Trans.+Pattern+Anal.+Mach.+Intell.&volume=24&pages=971%E2%80%93987&doi=10.1109/TPAMI.2002.1017623)] [**[CrossRef](https://doi.org/10.1109/TPAMI.2002.1017623" \t "_blank)**]
30. Chen, C.; Zhang, B.; Su, H.; Li, W.; Wang, L. Land-use scene classification using multi-scale completed local binary patterns. Signal Image Video Process. **2015**, 10, 745–752. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Land-use+scene+classification+using+multi-scale+completed+local+binary+patterns&author=Chen,+C.&author=Zhang,+B.&author=Su,+H.&author=Li,+W.&author=Wang,+L.&publication_year=2015&journal=Signal+Image+Video+Process.&volume=10&pages=745%E2%80%93752&doi=10.1007/s11760-015-0804-2)] [**[CrossRef](https://doi.org/10.1007/s11760-015-0804-2" \t "_blank)**]