LAND COVER AND LAND USE CLASSIFICATION IN REMOTE SENSING DATA USING DEEP LEARNING

A THESIS
SUBMITTED TO THE DELHI TECHNOLOGICAL UNIVERSITY
FOR THE AWARD OF THE DEGREE OF

DOCTOR OF PHILOSOPHY
In
Computer Science & Engineering

By

ABEBAW ALEM
(2K18/PHD/CO/24)

Under the Supervision of

PROF. SHAILENDER KUMAR
Delhi Technological University, India

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Delhi-110042, India

February, 2023

Dedication

To all Ethiopians who have taught me from elementary school until this degree award, while they themselves are melting like a candle.

CANDIDATE DECLARATION

I, Abebaw Alem (2K18/PHD/CO/24), hereby declare that the thesis entitled "**Land Cover and Land Use Classification in Remote Sensing Data Using Deep Learning**," submitted to Delhi Technological University, Delhi, in partial fulfillment of the requirements for the award of the degree of Doctor of Philosophy in the department of Computer Science and Engineering, is my own original work and has been completed under the supervision of Prof. Shailender Kumar (Supervisor), Department of Computer Science and Engineering, Delhi Technological University, Delhi, India.

The explanations presented are based on how I read and comprehended the original texts. I have never before submitted this work to any other institutions for the award of any other degree, diploma, associateship, fellowship, or other title or honor.

Abebaw Alem
Roll No. 2k18/PHD/CO/24)
Department of Computer Science and Engineering
Delhi Technological University
(Formerly Delhi College of Engineering)
Shahbad Daulatpur Main Bawana Road
Delhi-110042, India

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

(Govt. of National Capital Territory of Delhi)

Shahbad Daulatpur, main Bawana Road,

Delhi – 110042, India

Date: _________________

CERTIFICATE

This is to certify that the work embodied in the thesis entitled "**Land Cover and Land Use Classification in Remote Sensing Data Using Deep Learning"** submitted by Mr. Abebaw Alem (Roll No. 2K18/PHD/CO/24) as a full-time PhD scholar in partial fulfillment of the requirements for the award of the degree of Doctor of Philosophy, to the Delhi Technological University, Delhi, India, is carried out by the candidate under my supervision and guidance at the Department of Computer Science and Engineering, Delhi Technological University, Delhi, India.

The results embodied in this thesis have not been presented to any other university or institute for the award of any other degree or diploma.

Prof. Shailender Kumar
Department of Computer Science and Engineering
Delhi Technological University
Delhi-110042, India

ACKNOWLEDGMENTS

This study has been carried out in its entirety with the assistance of the following key players. It's an honor to be able to publicly thank the people and organizations that have been so supportive of me as I've conducted this study.

First of all, I want to begin by praising **God** and honoring His mother, **Saint Mary**, for keeping me from dangers like the COVID-19 pandemic and providing the motivation and energy I needed to complete this task. Nothing I have accomplished in my life has been possible without the grace of God and the intercession of His holy mother, Saint Mary.

Second, I'd like to thank my advisor, **Prof. Shailender Kumar**, a Professor of Computer Science and Engineering at Delhi Technological University, for all of his help throughout this research. Throughout the whole process of developing this research strategy, from primary data analysis to literature evaluation, his wealth of experience, enthusiasm, and insightful remarks have been significant assets. In addition to the research work, I learned a lot about cryptography from him and enjoyed his teaching manner in his "Information and Network Security" course.

Third, I'd like to thank **Debre Tabor University** and the **Ethiopian Ministry of Education** (MOE) for their continued financial assistance with my tuition and living expenses throughout the duration of this study. My gratitude also extends to the Embassy of Ethiopia in Delhi, which supports my financial processes and follows up my progress to complete my work. It was imperative that I give due recognition to these institutions, which I did.

Fourth, I'd like to express my gratitude to my Student Research Committee (SRC) and/or the Department Research Committee (DRC) members, who offered insightful and pertinent comments while this research proposal was being conducted. They have also provided ongoing follow-ups and insightful suggestions for the research progress report on a semesterly basis ever since the SRC was conducted. Those SRC/DRC members are **Prof. Rajni Jindal** (the former head of the department of Computer Science and Engineering (CSE) and the chairman of the SRC/DRC), **Prof. Pravin Chandra** and **Prof. S.K. Dhurandhar** (the experts of the SRC/DRC from outside the university of DTU), **Prof**. **Vinod Kumar** (the HOD-CSE and the expert of the SRC/DRC from inside the department within the DTU), and **Prof. Rajesh Rohilla** (the expert of the SRC/DRC from outside the department within the DTU).

Fifth, I'd like to express my gratitude to the people I've met during my time at **Delhi Technological University**. This includes the faculty members of the Department of CSE for their unwavering support during the course of this study, such as, **Prof. Anil Singh Parihar,** who taught me Natural Language Processing (NLP), **Dr. Akshi Kumar**, who taught me Advanced Web and Internet Technology (AWIT), and all the other lecturers in other departments who taught me Research Methodology and its elective courses; the **Data Mining Lab-mates**, who assisted me with every concern; the **Academic Postgraduate Section staffs**, who manipulate my every academic issues; the **International Affairs staffs**, who assist me by liaising my related issues with the Embassy of Ethiopia; the **Examination Branch staffs**, who communicated this thesis with external and internal examiners; and the **university security guards** who secure me including my properties in PG as well as in the university.
Sixth, I'd like to thank **Google Teams** for making it possible to use the Colaboratory cloud server. This is an alternative way to use graphics processing unit (GPU) processors instead of central processing unit (CPU) processors to implement the classification models used in this thesis.

Seventh, the **journal paper editors** and/or r**eviewers**, and **examiners** of the thesis, who provided pertinent and constructive comments for the thesis quality improvement are duly acknowledged.

Last but not least, I'd like to extend my family's sincere gratitude and appreciation. I'd like to express my gratitude to my beloved wife, **Genet Achenef**, for being the patient rock upon whom I could lean and for her unwavering support as I worked toward my study. This wouldn't have been possible without her unwavering support and watchful eye over our family. More than that, I'd like to express my gratitude to my **parents** and **parents-in-law** for always being there for me and providing countless supports throughout my life.

Abebaw Alem

Roll No.: 2k18/PHD/CO/24)
Department of Computer Science and Engineering
Delhi Technological University
(Formerly Delhi College of Engineering)
Shahbad Daulatpur Main Bawana Road
Delhi-110042, India

ABSTRACT

The classification of land cover and land use (LCLU) from remotely sensed imaging data has been a motivating area of study. LCLU data includes dynamic remote sensed images that exhibit inconsistencies due to limitations in sensor technology, seasonal changes, and spatial distance. The LCLU classification problem has recently been solved with the help of deep learning, an advancement in machine learning and artificial intelligence (AI). Recently, deep learning classification systems have been recognized as a powerful and popular modeling tool for extracting hidden information from remote sensing data for LCLU classification in the observed earth environment. Since the LCLU classification system is based on deep learning, it is important to look into these methods for environmental control, environmental management, agricultural decisions, and urban development.

For the LCLU classification issue, we collected data from publicly available sources and developed deep learning strategies employing convolutional neural networks, transfer learning, and pretrained networks. Many recent studies have examined deep convolutional neural networks in remote sensing categorization, with the networks having been trained using pretrained networks. However, this has not been extensively explored because of the time and processing power required to train convolutional neural networks for remotely sensed images. In order to classify LCLUs in the University of California Merced (UCM) dataset, we used hyperparameters, regularization, and early stopping in a convolutional neural network feature extractor (CNN-FE) deep learning approach. In order to ensure cross-domain generalizability, we retrained the CNN-FE model using the SIRI-WHU dataset and used the same hyperparameters to build the VGG19 pretrained feature extractor model. However, the training period for CNN model is very long. Deep transfer learning (TL) modeling, which makes use of pretrained models to quickly build TL models, could be used to solve this issue. The use of DL and machine learning techniques for image classification has recently shifted its focus to transfer learning. And the pretrained networks are efficient. Using various remote sensed hyperspectral images, we developed convolutional neural network–based pre–trained models for LCLU classification, such as EfficientNetB7, InceptionV3, and MobileNet deep learning models.

We compared the results from the state-of-the-art studies and other built models with those from the UCM, SIRI-WHU, and RSSCN7 datasets after building and training the DL models on those datasets. The first experiment was designing CNN-FE. Results in this experiment showed performance improvements for the CNN-FE model when compared to state-of-the-art baseline studies and the VGG-19 pretrained model. Additionally, the CNN-FE model's performance was better when trained on the UCM dataset compared to when trained on the SIRI-WHU dataset. The second experiment was building the TL model. We used the InceptionV3, Resnet50V2, and VGG19 pretrained models for LCLU classification in the UCM dataset, with the TL model trained with bottleneck feature extraction. Based on these experiments, the TL model was developed, with improved results of 92.46, 94.38, and 99.64 in Resnet50V2, InceptionV3, and VGG19, respectively. The third experiment was the comparative evaluation of the CNN-FE and TL with fine-tuning. In this experiment, the fine-tuning model has outperformed the CNN-FE and TL in both the UCM and SIRI-WHU datasets. The fourth experiment was building the DL models, such as EfficientNetB7, InceptionV3, and MobileNet, for different datasets of UCM, SIRI-WHU, and RSSCN7 that have distinct parameters. In accuracy performance, the MobileNet outperformed the competition on the UCM and SIRI-WHU datasets, while EfficientNetB7 performed better on the RSSCN7 dataset. We also found that the dataset had an effect on the model's efficiency, with the UCM dataset outperforming the SIRI-WHU and RSSCN7 datasets across the board in terms of most measurement measures. The findings of this study indicated that it could provide significant benefits to remote sensing communities and decision-makers. The need for a powerful processing unit and the limited time frames caused by COVID-19 were the major challenges and limitations of this research. Based on these challenges, we have come up with some recommendations for the future, such as using a more powerful processor to improve the performance of DL models and applying DL hyperparameters to the domain area.

Keywords: convolutional neural network, deep learning, end-to-end learning, land cover and use classification, performance evaluations, pretrained model, remote sensed image, transfer learning.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

AdaDelta: Adaptive Delta Algorithm

AdaGrad: Adaptive Gradient Algorithm

Adam: Adaptive Moment Estimation

AI: Artificial Intelligence

aka: also known as

ANNs: Artificial Neural Networks

CM: Confusion Matrix

CNN-FE: Convolutional Neural Network Feature Extractor

CNNs: Convolutional Neural Networks

CNTK: Microsoft Cognitive Toolkit

Colab: Colaboratory

Conv2D: Convolution with two Dimensions

COVID-19: Coronavirus Disease founded in 2019

CPU: Central Processing Unit

CV: Computer Vision

DBN: Deep Belief Network

DL: Deep Learning

ES: Expert Systems

FCN: Fully Connected layers/ Networks

GAN: Generative Adversarial Network

GIS: Geographic Information Systems

GPU: Graphics Processing Unit

IEEE: Institute of Electrical and Electronics Engineers

LCLU: Land Cover and Land Use

LR: Learning Rate

MDPI: Multidisciplinary Digital Publishing Institute

ML: Machine Learning

NLP: Natural Language Processing

Relu: Rectified Linear Unit

ResNet: Residual Neural Network

RGB: Red, Green and Blue channels

RMSProp: Root Mean Square Probability

RNN: Recurrent Neural Network

RO: Research Objective

RS: Remote Sensed or Remote Sensing

RSHIs: Remote Sensed Hyperspectral Images

RSSCN: Remote Sensing Scene Classification image

SGD: Stochastic Gradient Descent

SVM: Support Vector Machines

Tanh: Tangent Hyperbolic Function

TL: Transfer Learning

VGG19: Visual Geometry Group with 19 convolutional layers

VGGNet: Visual Geometry Group Network

VPU: Vision Processing Unit

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Land Cover and Land Use (LCLU) Classification

The land itself is a natural resource because it is the medium of existence for all earthly things, both living and nonliving. Potential components are land cover and land use (LCLU). While land cover is a naturally occurring occurrence on Earth, land use is the result of human intervention in order to use the land cover to the best possible advantage. The classification of LCLU is a difficult phenomenon in RS that hinders effective management of the Earth's ecosystems. Features on Earth are described and extracted from satellite data using LCLU. Land cover characterizes the earth's surface in terms of its physical features, while land use describes the socio-economic function of a certain area of land [1].

Satellite data is earth geospatial data with metadata. The metadata of satellite images describes the location of objects on Earth. Spatial data, often known as geospatial data, is information about locations and their features. It keeps tabs on data pertaining to the natural world. Because land cover is a continuous representation of spatial data, it is useful for making decisions based on the LCLU classification system when it is well organized and interpreted. Objects inside a geographic system include the earth and everything on it, including trees, buildings, rivers, grasslands, and people, which are all examples of objects for RS data collected from the earth’s environment. LCLU is the extraction of features in the land from RS data, such as residential areas, rivers, grasslands, forests, and the like. The extraction of land features from RS data is a crucial part of urban and agricultural planning and decision-making.

In general, RS is the acquisition of information about any phenomenon without making physical contact with the object; simply put, it is an onsite observation on the earth. It's the practice of identifying and categorizing terrestrial features using data collected by sensors deployed from spacecraft or airplanes. Geography, geoinformatics, AI, planning, and humanitarians are some of its subject areas, while agriculture, healthcare, LCLU categorization, environmental monitoring, climate change, geographic information systems (GIS), and urban planning are just a few of the domain areas that can benefit from this field of study. It is possible to apply DL techniques to create LCLU classification intelligent systems. There are publicly available RS datasets used to design this LCLU classification system.

Agricultural spots, healthcare, language analysis, and image processing are all possible application areas for DL techniques. Based on our examination of the existing literature, we have identified three research gaps where more investigation is needed. Therefore, we plan to use DL techniques for LCLU classification in RS image datasets.

1.2 Deep Learning Techniques

Artificial intelligence (AI), machine learning (ML), and deep learning (DL) have emerged as major fields of study applied in many domain areas in recent years. AI is a broad area of research into the development of intelligent software. The term "ML" is commonly used to describe the evolution of AI-related system capabilities. Thus, ML approaches, where a computer learns from incoming data and increases its efficiency and effectiveness, have made AI a prominent research subject. AI has the potential to significantly alter many industries, including agriculture, the earth's environment, transportation, healthcare, language analysis, and the media.

Therefore, "ML" refers to the theory, practice, and method of modeling and analyzing the data-driven computer learning process. When it comes to data management, ML approaches can be broken down into two distinct camps: the classical or traditional ones and the recent or DL data-driven solutions. Supervised learning, unsupervised learning, and reinforcement learning are three areas that have recently attracted a lot of attention in ML research. Most research in both traditional and DL methods is done in the supervised ML area, which can be used in a wide range of situations. Multiple neural networks, some of which might more accurately be described as "artificial neural networks" (ANNs), make up AI (ANNs). One AI method that has been applied to the task of modeling and evaluating satellite imagery and data is the usage of ANNs. The more ANNs there are, the more DL is created, and DL is the newest approach of advancing ML. Recently, DL has risen to prominence as a key research area in the field of ML.

Different researchers have described DL as a subset of a new ML technique used for interdisciplinary data analysis. The study of remote sensing (RS) image data might have benefited from the deployment of DL algorithms, a promising field of ML. In circumstances where the agent has no prior information, as may be the case, DL becomes crucial for unknown environments. The geoinformatics and RS communities lack access to RS data. Therefore, DL allows for the structuring of newly acquired information into generic, efficient representations for the purposes of decision-making.

A variety of DL methods can be used to analyze and model RS data. Deep ANNs are ideal for processing large amounts of data quickly in a specific real-world domain. Among these ANNs are convolutional neural networks (CNNs), recurrent neural networks (RNNs), and generative adversarial networks (GANs). The most widely used supervised classification algorithms for extracting image features are CNNs [2]. Thus, CNNs are the most prominent modeling technique used by researchers in LCLU classifications. CNNs are available with dependable modeling architectures, including AlexNet, GoogleNet (InceptionNet), VGGNet, ResNet, EfficientNet, and MobileNet.

1.3 Motivations of the Study

With its performance improvements in large dataset processing skills, particularly in RS data, DL has become a flourishing field of study and a recent active research area in many challenging practical applications, which motivated us to begin this investigation. We were also inspired to begin this research in part because of the suggestions or recommendations made in past investigations.

The forwarded recommendations by the earlier researchers, such as using pretrained networks and fine-tuning for satellite data classifications [3], [4], utilizing appropriate DL methods to better scene classification tasks [5], analyzing the relation between the number of classes in the dataset and the number of parameters in the ConvNet and their impact on the discrepancy between fine tuning and full training processes [6], and evaluating the efficiency of CNN with very large scale images [7], are also our motivations to do this study. Such prior research experience influenced and motivated us to pursue this research problem that allows us to understand the challenges, opportunities and significances and to identify the gaps of the domain area.

1.4 Research Objectives

The main goal of this study is to come up with DL methods for LCLU classification in Earth Observation RS imagery data. To achieve this aim, the following specific objectives are proposed.

- To analyze state-of-the-art of deep learning methods for LCLU classification in RS data.
- To design a deep learning convolutional neural network model for LCLU classification.
- To design transfer learning for LCLU classification.
- To compare the performance of convolutional neural network and transfer learning for LCLU classification.
- To evaluate deep learning methods for LCLU classification using different data sets.

After the aforementioned research objectives (ROs) have been set, we have performed each objective using different DL methods and RS datasets. Therefore, the findings and achievements for the ROs have been summarized in the following ways.

RO 1: In this objective, we have reviewed and analyzed the existing studies by retrieving them from different databases to dig out the DL methods applied to the domain area. Reviewing and surveying the earlier works is not an easy task. We have performed it from the start to the end of our study. This RO enabled us to identify the research gaps that existed at the start of the study. Therefore, we set it as the first RO, and we have achieved it by surveying the papers and identifying the gaps.

RO 2: For this objective, we designed the deep CNN model for LCLU classification in RS images. The model's ability to work in the domain was checked by comparing it to the VGG19 network that had already been trained and retraining it on the other publicly available RS dataset.

RO 3: In this RO, the TL DL method has been designed for the classification problem. The TL model was designed using different pretrained models for LCLU classification in RS images. This model has been effective in terms of time and resource consumption.

RO 4: We designed and compared the CNN and TL for LCLU classification in the RS image. In addition to these two DL models, we applied the fine-tuning technique. Each of the three models has distinct advantages. In terms of efficient time consumption and resource requirements, the TL and fine-tuning models are more significant than the developed CNN model. Whereas, in the case of individual class feature analysis, the CNN model is more significant than the TL and the fine-tuning models. Finally, the fine-tuning model outperformed the performances in a short period of time.

RO 5: This RO described the performance evaluation of various DL models using various datasets. This objective enabled us to identify that the properties of the dataset have significant influences on the classification system's performance.

In general, we have achieved each research objective positively. The achievements of each objective lined up with publications are summarized in Table 1.1.

Table 1.1. Achievements of ROs aligned with publications

ROs	Publication per Ros
RO 1	A. Alem and S. Kumar, "Deep Learning Methods for Land Cover and Land Use Classification in Remote Sensing: A Review," 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, June 4-5, 2020, pp. 903-908, doi: 10.1109/ICRITO48877.2020.9197824. (**In IEEE Explore Scopus Indexed**)
RO 2	A. Alem and S. Kumar (2022), End-to-end Convolutional Neural Network Feature Extraction for Remote Sensed Images Classification, Applied Artificial Intelligence, vol. 36 (1), art. no. 2137650, DOI: 10.1080/08839514.2022.2137650 (**SCIE, IF: 2.777, best quartile- Q2**)
RO 3	1. A. Alem and S. Kumar (2021), Transfer Learning Models for Land Cover and Land Use Classification in Remote Sensing Image, Applied Artificial Intelligence, vol. 36 (1), art. no. 2014192, DOI: 10.1080/08839514.2021.2014192 (**SCIE, IF: 2.777, best quartile- Q2**) 2. Abebaw Alem, Shailender Kumar, Transfer Learning for Land Use Classification in Remote Sensing. In Jamia Teachers' Association Multidisciplinary International Conference (JTACON-2020), New Delhi, India, February 16-18, 2020. (**Presented**)
RO 4	A. Alem and S. Kumar (2022) Deep Learning Models Performance Evaluations for Remote Sensed Image Classification, IEEE Access, vol. 10, pp. 111784-111793, DOI: 10.1109/ACCESS.2022.3215264 (**SCIE, IF: 3.476, best quartile- Q1**)
RO 5	A. Alem and S. Kumar (2022), "Deep Learning Models for Remote Sensed Hyperspectral Image Classification," 2022 13th International Conference on Computing, Communication and network Technologies (ICCCNT), Kharagpur, India, Oct 3-5, 2022, pp. 1-7, doi: 10.1109/ICCCNT54827.2022.9984282. (**In IEEE Explore Scopus Indexed**).

1.5 Limitations of the Study

During working on this study, we faced limitations or constraints that could affect the models' performances. These limitations include the lack of powerful computational resources, i.e., robust processor needs and time constraints resulting from the COVID-19 epidemic. Since we were working with a CPU processor and Google Colab instead of a GPU processor, we had to work with relatively small RS datasets. In addition, both the TL and fine-tuning modes work with small data sets. Developing DL models from scratch is time-consuming and difficult, so the size of the dataset may have an effect on the DL performance. We didn't have enough time to compare typical ML methods with DL methods or test the effects of important DL hyperparameters that can't be learned, like the deeper number of layers, iteration, and batch normalization (mean and variance).

1.6 Significance of the Study

The use of DL algorithms for LCLU classification employing RS imaging data has the potential to bring about a variety of advantages for the communities located within the determined scope of application. The findings of this research may have direct and indirect significances for the community. The researchers are the primary direct beneficiaries of getting the academic promotion awards by fulfilling requirements in the academic program and publishing the findings in peer-reviewed reputable journals. Moreover, this study has the following direct or indirect significances for its beneficiaries:

- To address the needs of the RS communities, an intelligent LCLU classification system based on DL might be used to address the issue of scene classification. So, this research could be the first step toward making a fully autonomous LCLU classification system that could be used to classify, control, and manage Earth's natural resources more effectively.
- The study may also be useful for the decision-makers. Using the LCLU classification system will considerably improve the likelihood of making decisions that contribute to sustainable development in the areas of agricultural and urban planning, environmental protection, and natural resource management.
- We also looked at how the DL models worked and improved their performance, showing that they could be used to classify LCLUs in RS images. However, as mentioned earlier, there are limitations to the study, and we suggest future research directions based on these limitations. Therefore, future researchers will indirectly benefit from this work because they can use its limitations as a starting point for their own investigations.

1.7 Organization of the Thesis

The thesis is organized into eight consecutive chapters to describe the thesis from the beginning to the end. Each chapter has been described in the following manner:

Chapter 1: This chapter deals with the introduction of the thesis, which includes the general concepts of the terms, the methods, the motivations, the objectives, the limitations, and the significance of the thesis.

Chapter 2: This chapter states the literature review and the literature survey analysis that enable us to retrieve related ideas and methods to identify the gaps. We have retrieved and analyzed the related background details of the DL methods used for the LCLU classification domain. The chapter includes the theoretical understanding, the survey analysis, the related works, and the research gaps.

Chapter 3: The chapter deals with the research materials and the DL methods used in this study. The chapter states the publicly available RS datasets, the reliable DL methods for the LCLU classification problem, and the tools and DL frameworks used for implementing the identified classification problem.

Chapter 4: The chapter describes the design of the CNN DL method for LCLU classification in RS images. The model has been checked its applicability on the domain by comparing with the pretrained VGG19 network and retraining it on the other publicly available RS dataset. We checked that the CNN model was applicable in the domain area.

Chapter 5: In this chapter, the TL DL method has been designed for the classification problem. The TL model was designed using different pretrained models for LCLU classification in RS images. This model has been effective in terms of time and resource consumption.

Chapter 6: The sixth chapter deals with the comparisons of the CNN, TL, and fine-tuning for LCLU classification in the RS image. Each of the three algorithms has distinct advantages. In the case of less time consumption and resource requirement, the TL and fine-tuning models are more significant than the CNN-FE model. Whereas, in the case of individual class feature analysis, the CNN model is more significant than the TL and the fine-tuning models.

Chapter 7: The seventh chapter describes the performance evaluation of DL using various datasets. In this chapter, we identified that the properties of the dataset have significant influences on the classification system's performance.

Chapter 8: Finally, the last chapter deals with the conclusions and recommendations of the thesis. This chapter describes the problem, objectives, methods, findings, contributions, significance, limitations, and future suggestions based on the identified limitations.

2. LITERATURE REVIEW AND RESEARCH GAPS IDENTIFICATION

2.1 Land Cover and Land Use (LCLU) Classification

Land is an important natural resource that supports all living and nonliving things on the earth. Humans are in charge of managing this pillar resource, which allows for the existence of other things. Fully automated land cover classification is a hard problem that requires ML and computer vision tasks. Classifying land use from RS imagery is also important for monitoring and managing human development [8].

Land cover expresses physical features of the earth's surface, whereas land use describes a piece of land's socioeconomic function [1]. Land cover classification is a popular and thriving research area in RS applications [9], and we would look into using deep CNNs to handle and challenge ML tasks to improve performance when using trained augmentation data [3], [10].

Land cover and land use are two different terms, but we used them as one in our study. Because the land use class (e.g., residential or commercial area) often consists of multiple classes of land cover (e.g., buildings, trees, and roads) with great variance [11]. In this study, land cover and land use are phrased as one and used interchangeably. LCLU is a description of the earth’s features and an extraction of these features from satellite data. These include the extraction of various land features such as roads, urban areas, buildings, agricultural lands, waterways, sports fields, grassland, forests, and the like. LCLU is a description of the earth's features as well as human interaction with it. This LCLU classification is an important task [12], and it contributes to decision-making and planning in the earth observation environment, which is a domain of satellite imagery classification.

Thus, in the earth observation environment, such as the rural and urban sectors, the LCLU classification enables humans to make decisions and plan [13]–[15]. LCLU is an important source of earth information and a research problem for DL systems that use automatic classification. To utilize the classification system, many scholars have been investigating the classification of land features with DL methods using satellite data. Therefore, LCLU classification is a recent popular and important research field to utilize RS applications in various challenging tasks such as urban planning, land resources management, environmental monitoring and detection, and governmental management in RS technology [14], [16]–[18]. RS is becoming a dominant source for acquiring images and performing LCLU classification tasks [17], and it plays a significant role in the field of earth observation [19].

2.2 Machine Learning Techniques

AI is one of the newest multidisciplinary sciences and research areas in the recent "big data era." It is a general field that encompasses ML and DL. It also includes many more approaches that don't involve any other learning tasks other than ML, as observed in Figure 2.2. Now days, researchers have been investigating findings on AI domain areas such as image classification, object detection, speech recognition, and language analysis by using recent ML approaches.

ML is the 21st century's hot research area in AI; it is an approach, a method, and a science of modeling and analyzing computer learning processes using data, observations, examples, and autonomously experiences. ML is an iterative process of training models and usually refers to the changes in systems that perform tasks associated with AI. These changes in iterations of ML systems could occur due to data preprocessing and feature selection, the algorithm selection and parameters used, and the training data. The ML tasks in LCLU using RS imagery data involve environmental classifications, monitoring, and detection (change and object); agricultural and urbanization planning; land resource management; and government management (decision making). ML systems use generic algorithms that can extract hidden information from a set of data.

It is stated by [20] that the field of ML usually distinguishes three learning capability areas: supervised, unsupervised, and reinforcement learning. Supervised learning is the ability of the model to learn from the input and output data. It focuses on the classifications or predictions and regression problems according to their categories. Unsupervised learning is the capability of the model that is expected to learn features on its own without guidance. It aims to group or cluster objects according to their similarities. Reinforcement learning focuses on the motivational adaptation for successful performance by learning from its mistakes, as seen in Figure 2.1. The reinforcement learning model could provide positive or negative feedback, which enabled a better decision.

Supervised classification is the most active research area in satellite data analysis using both conventional/classical and recent ML models. Very high spatial resolution (VHSR) images used for supervised classification are still an open area of research in the RS [5].

Supervised learning is used to analyze structured (such as databases) and unstructured data (such as images and audio). In this study, we are focusing on supervised classification using recent ML methods called DL methods to analyze RS imagery data. Training data in DL models is a key component of supervised learning, and most ML algorithms require a large number of training data samples [21].

Data is the engine, coal, and input process for ML algorithms. For this reason, ML is considered a data governance method. There are various datasets in various application areas. RS satellite data is focused forward for LCLU classification. In this "big data era," there will be a paradigm shift towards data-intensive science [22], and ML techniques are keys for analyzing big data in this era.

There are different approaches of ML in AI that could analyze and model satellite image data. These ML algorithms could be categorized into two groups based on their depth. The first approach is the classical/traditional algorithm that includes support vector machines (SVM), classical ANNs (ANNs with one layer only), and maximum likelihood estimators [23]. The second approach is the **recent one** (DL), which is the advancement of classical neural networks. DL methods, such as CNNs [4], [24]–[26], RNNs [27], and DNNs classifiers [28], outperform traditional ML methods in terms of performance and power for RS imagery classification. Therefore, we are motivated to focus on DL methods to extract RS features.

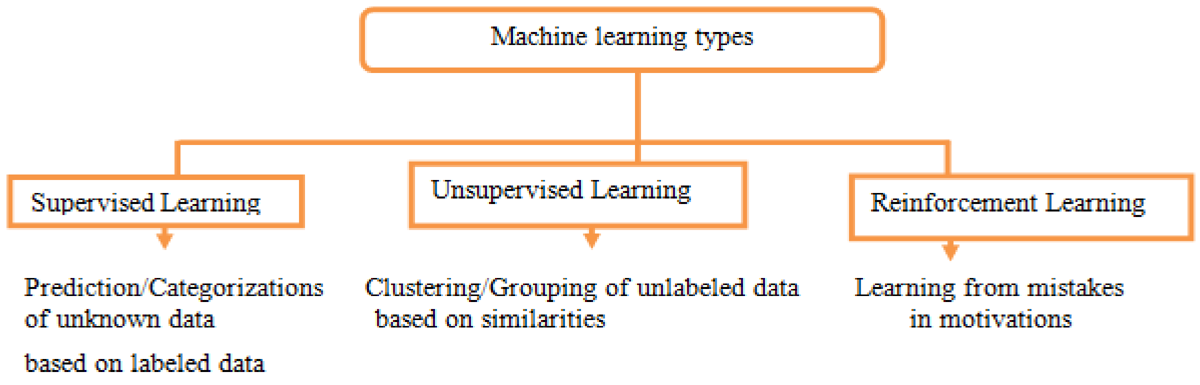

Machine learning types
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Prediction/Categorizations of unknown data based on labeled data
Clustering/Grouping of unlabeled data based on similarities
Learning from mistakes in motivations

Figure 2.1. Machine Learning Techniques

2.3 Deep Learning Methods

DL is a subdivision of the ML method that uses deep architectures to learn high-level image feature representations. It is a recent technology that various researchers have focused on for its reliable performance. More ANNs create DL, which is the capability to train more neural networks on a given dataset. ML techniques are becoming increasingly important, and DL has proven to be an extremely powerful method in many fields and one of the fastest-growing trends in big data analysis [29]. Satellite image data sets have the characteristics of a large data volume and complex image classification [30]. This complex image would be analyzed in DL. DL is a subset of ML that focuses on learning successive layers of increasingly in depth and meaningful representations of neurons. DL is an extension of classical ML methods, i.e., the depth of more ANNs, and has become powerful in recent research focus areas. DL is an end-to-end learning (feature learning and abstraction, model learning) method that consists of more than five processing layers, mostly in supervised classification [31], [32]. Supervised DL, especially CNN, is the most fitting modeling method for RS image classification [33]. The interconnection between AI, ML, and DL is summarized in Figure 2.2.

Therefore, DL is a hotspot in the deep ML area and would have been used in RS image data analysis. DL is essential for unknown environments, i.e., when the agent lacks knowledge. RS data are not known by the geoinformatics and RS communities as they are hunted by RS technologies. Therefore, DL enables the organization of new knowledge into general, effective representations for decisions and planning.

Research results have found more accurate performance when performed with DL methods. The deeper the network is the better the performance of the model. The performance of the LCLU classification could be increased by adding more neural networks [15] and [34].

Recent studies used DL methods for LCLU classification in RS datasets with Deep Neural Networks (DNNs) [28]. DL methods with DNNs include various modeling methods, such as CNNs [4]–[8], [15], [24], [35]–[37], RNNs [38], GANs [39], [40], and deep belief networks (DBN) [41], [42].

CNN is superior to other deep network algorithms due to its abilities [33], [43]. CNN has achieved remarkable results in image classification, recognition, and other computer vision tasks [44].

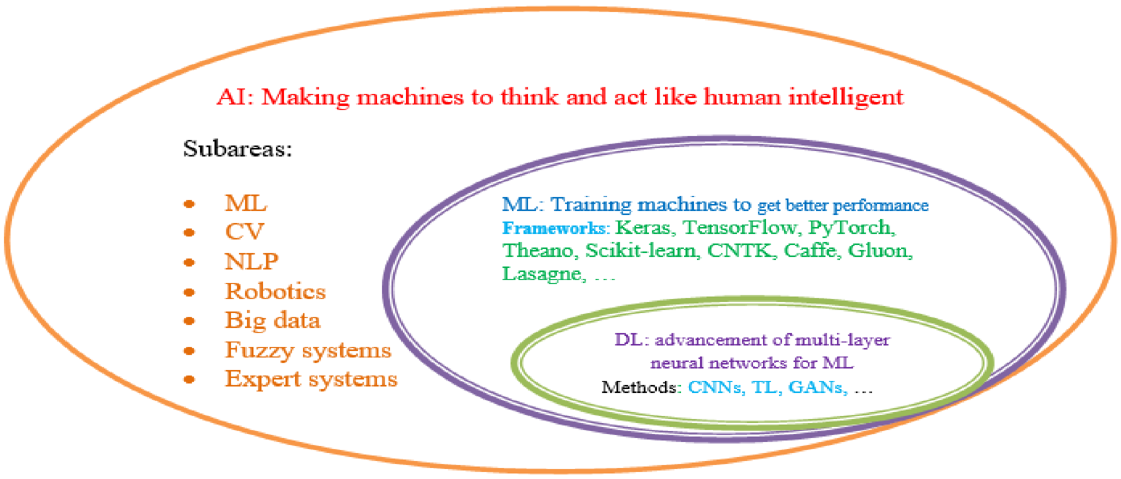

AI: Making machines to think and act like human intelligent
Subareas:
ML
CV
NLP
Robotics
Big data
Fuzzy systems
Expert systems
ML: Training machines to get better performance
Frameworks: Keras, TensorFlow, PyTorch, Theano, Scikit-learn, CNTK, Caffe, Gluon, Lasagne, …
DL: advancement of multi-layer neural networks for ML
Methods: CNNs, TL, GANs, …

Figure 2.2. Relation between AI, ML and DL and frameworks and methods of ML

2.3.1 Convolutional Neural Networks (CNNs) for Classification

In recent times, the most researched areas of AI could be categorized as symbolic and connectionist AI. The symbolic research focuses on the symbolic AI that includes predicates and fuzzy logic (rule-based systems), while the connectionist AI focuses on interconnected entities like nodes. The connectionist AI is investigated in our research that dealt with the interconnected neurons, which could be named neural networks. This focused research domain could be implemented in traditional and advanced (DL) ML AI approaches. We focused on the advanced ML approaches that included CNN in our study.

The CNNs are multi-layer neural networks that are used to extract image features, or pixels. CNNs are one of the DL methods that are particularly designed for RS image classification based on multi-layer ANNs, and they have been mostly used in recent research. DL CNN is a popular and widely used method for LCLU classification using RS data [45]. CNNs are the most well-known DL algorithms and have gained interest from researchers for RS image processing in recent years [29]. According to many researchers, the interest in CNNs is growing rapidly due to their impressive results. DCNNs have recently emerged as a dominant paradigm for ML in a variety of domains, such as the RS domain for land cover classifications [8].

The exploration of the potentials of deep CNNs can be a complex task because of several challenges, which are over-parameterized [6]. Application cases of CNN-based RS image classification are classified into scene classification (based on RS contents), object detection (labeling locations and types of the targets with bounding boxes), and object extraction (accurate boundaries of the objects to be extracted in RS) [46], [47]. CNNs have shown powerful feature representation capability to improve scene classification of RS imagery [48].

CNNs are built with a series of layers, including convolutional layers, pooling layers, fully connected layers, and an output layer [32], [46], [47], [49], [50], as shown in Figure *2.3*.

There are various architectures of CNNs that are used by researchers to build CNN models for RS image classification. The most commonly used architectures are AlexNet, VGGNet, ResNet, and GoogleNet [7], [50]–[53]. These architectures of CNNs have their own characteristics for RS image classifications.

CNNs could be shallow in recurrent and feedback connections [54], or have a forward and backward stream in a recurrent network in a fully convolutional network architecture [55].

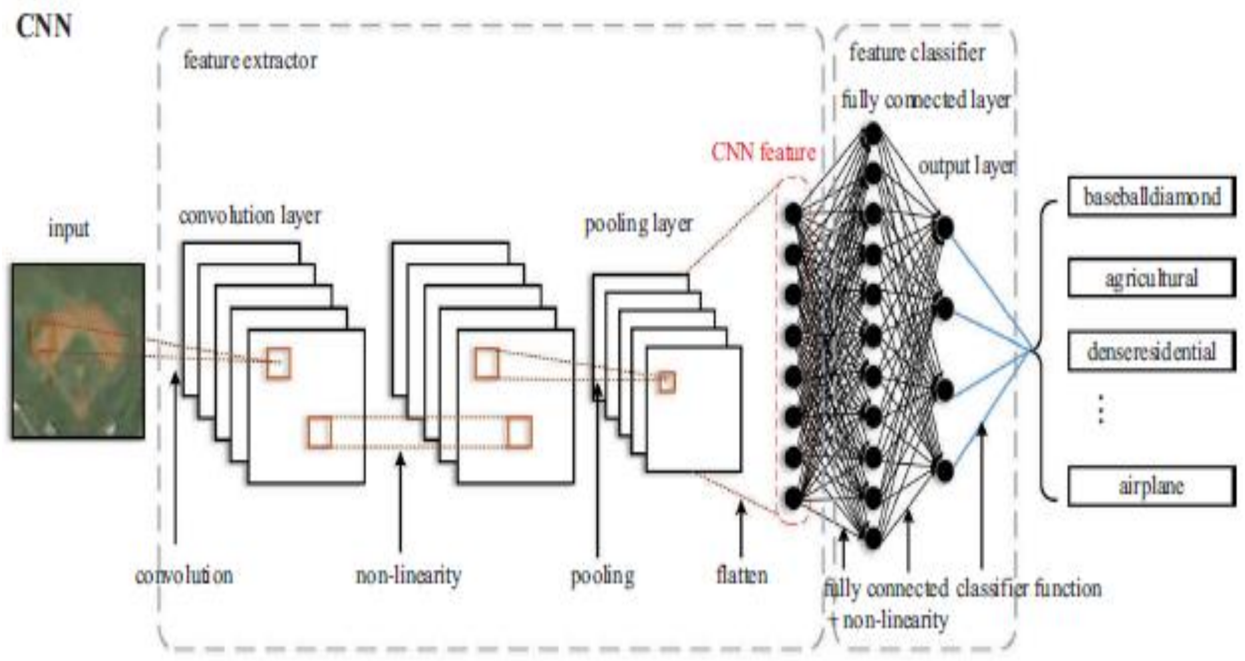


Figure 2.3. Structure of CNNs adopted from [45]

2.3.2 Transfer Learning

Transfer learning (TL), gets more attention for reducing the training time and the dependence on large amounts of training datasets [8], [56], [57]. The TL algorithm reuses the pretrained models such as AlexNets, ResNets, InceptionNets, and VGGNets to build new models. We have been trying to investigate the recommended work suggested by [3]. The TL is widely used for RS image classification [5], [11], [58], [59] because RS images are essential for LCLU classification in DL approaches [60].

2.4 RS Dataset Descriptions

Data are fuels and new oils for ML as well as DL. The LCLU classification problem could be investigated using earth observation data called satellite data. Satellite data are earth observation data, records of environmental information, and are used to make LCLU decisions when organized because land cover is a continuous spatial data representative. A satellite image is an image of the whole or part of the earth taken using artificial satellites [7].

RS is used to obtain the information from the earth's surface by using a satellite imaging system [23]. The term "remote" means an agent without physical contact, and the term "sensing" means a sense of observing the environment for measurement of information or data. Thus, RS is the acquisition of information about any phenomenon without making physical contact with the object; simply, it is an onsite observation on the earth. It is the use of satellite- or aircraft-based sensor technologies to detect and classify objects on the earth.

RS image classification is still facing unprecedented and significant challenges and has been an active research topic [36], [61]–[67] for DL applications in LCLU classification. Because RS image data has high resolutions and is frequently multimodal, geolocated, and time-variable, it is frequently used for object detection or classification [22].

Satellite RS data could be applied in both subject and domain areas to conduct research. RS has applications in different disciplines such as geography, geoinformatics, AI, planning, and humanitarian aid. Environmental management, agricultural planning, health studies, climate and biodiversity monitoring, LCLU mapping, land change detection, spatial data analysis, water resources, forestry, and GIS are some of the application domains for RS. Therefore, land use (LU) planning based on the land cover is vital for development. For instance, urban planning benefits from keeping track of the evolution of city centers or knowing how the land is used, such as for public facilities, residential areas, or commercial areas [13].

RS data for those domain areas is open for any investigation and can be found mostly on USGS, Earth Explorer, and other sites. Considering future research with big RS data to validate RS systems in more urban areas of the world is important [68]. Satellite data with greater spectra resolution and geographical variations could also be considered [5].

The role of RS image-based scene classification in LCLU classifications is significant [24], [69]. To design and evaluate DL methods for LCLU classifications, various RS datasets could be collected from their sources. The United States Geological Survey (USGS), European Space Agency (ESA), and Google Earth are the major sources of RS datasets. From our extensive analysis of the literature review, we collected, described, and summarized the data in Table 2.1.

UCM (University of California Merced) data set: To address the problems of LCLU classification, a number of researchers have been using the UCM Land Use data set. This dataset was manually collected and introduced by [70] from the USGS National Map Urban Area Imagery. This dataset consists of 21 land use and land cover classes that contain 100 images each with RGB color bands, measuring 256 × 256 pixels with a spatial resolution of about 30 cm/pixel. This data has been using by most researchers for LCLU classification. This data set has been used mostly by many researchers and it is/was publicly available at:
http:// vision.ucmerced.edu/datasets/landuse.html.

AID (Aerial Image Dataset): this data set consists of 30 classes with 10000 images each in a size of 600 × 600 pixels with 0.5-m to 8-m /pixel spatial resolutions. It was introduced by [71].

NWPU-RESISC45 (Northwestern Poly technical University-Remote Sensing Image Scene Classification) dataset: this is a larger dataset that consists of 45 scene classes with 700 images each and contains a total of 31,500 images with a size of 256 × 256 pixels in the RGB color space. This large-scale data set was created by [69] at NWPU. The UC Merced, AID, and NWPU-RESISC45 datasets were used by [72] to classify RS images using descriptive CNNs (D-CNNs) and compare the results of each dataset with ML methods.

RSI-CB (RS Image Classification Benchmark) data set: is a worldwide large-scale benchmark dataset with 0.22 to 3 m/pixel spatial resolution for RS image classification via crowdsourced data. It was built with two versions of pixels: RSI-CB128 (128 × 128 pixels) and RSI-CB256 (256 × 256 pixels), and constructed by collecting sample images from Google Earth imagery and being mapped by [73]. The UCM, AID, and RSI-CB256 RS image datasets were used by [74] for cross-domain semi-supervised learning classification using Classifier-Constrained Deep Adversarial Domain Adaptation method.

EuroSAT- Sentile-2A: Sentile data sets have different versions and levels: Sentile 1, Sentile 2, and Sentile 3 and above. However, Sentile 1 and 2 are mostly used, while Sentile 3 and above have not been used yet for research, even if they are released by ESA. As stated and used by [35], the satellite Sentinels data have been operated by ESA within its Copernicus program to improve earth observation, and two satellite Sentinel datasets were successfully launched in June 2015 (Sentinel-2A) and March 2017 (Sentinel-2B). The Sentile-2A EuroSAT is made up of ten classes with a total of 27000 images in 64×64 pixels and a resolution of 10 meters per pixel that cover 34 European urban atlases. [35] first mentioned this dataset.

RSSCN7 dataset: It was collected from Google Earth by [41]. It includes 2800 RS scene images with 400 x 400 pixels resolution in each image and seven classes. The dataset is or was available at https://sites.google.com/site/qinzoucn/documents.

The SIRI-WHU dataset was collected from Google Earth and covered urban areas in China [75]. The dataset contains twelve categories with 200 images per category with 200 x 200 pixels. The dataset is available at: https://figshare.com/articles/dataset/SIRI_WHU_Dataset/8796980.

LandSat is also one of the major data sets used in LCLU classifications. LandSat data has been used by researchers. Landsat 1 through Landsat 9 data sets are available on the USGS website: https://earthexplorer.usgs.gov. SAT4 and SAT6 data sets were used for LC classification in DL architecture [30], [51], [52], [76]. This dataset was available at http://csc.lsu.edu/*saikat/deepsat. Some RS datasets are summarized in Table 2.1. In general, we extracted the data from the selected primary studies and summarized it in Table 2.2.

Table 2.1. Most commonly used publicly available RS datasets

Dataset	Total Images	No. of classes	Av. Images/ classes	Resolutions (m/pixel)	Size (in pixels)	Introduced year	Contributed by
UCM	2100	21	100	0.3	256×256	2010	[70]
RSSCN7	2800	7	400	-	400×400	2015	[41]
SAT4	500000	4	83333	1	28×28	2015	-
SAT6	405000	6	67500	1	28×28	2015	-
SIRI-WHU	2400	12	200	2	200*200	2016	[75]
AID	10000	30	333	0.5-8	600×600	2017	[71]
RSI-CB256	24747	35	690	0.22-3	256×256	2017	[73]
RSI-CB128	36707	45	800	0.22-3	128×128	2017	[73]
NWPU-RESISC45	31500	45	700	0.2-30	256×256	2017	[69]
EuroSAT	27000	10	2700	10-60	64×64	2019	[35]

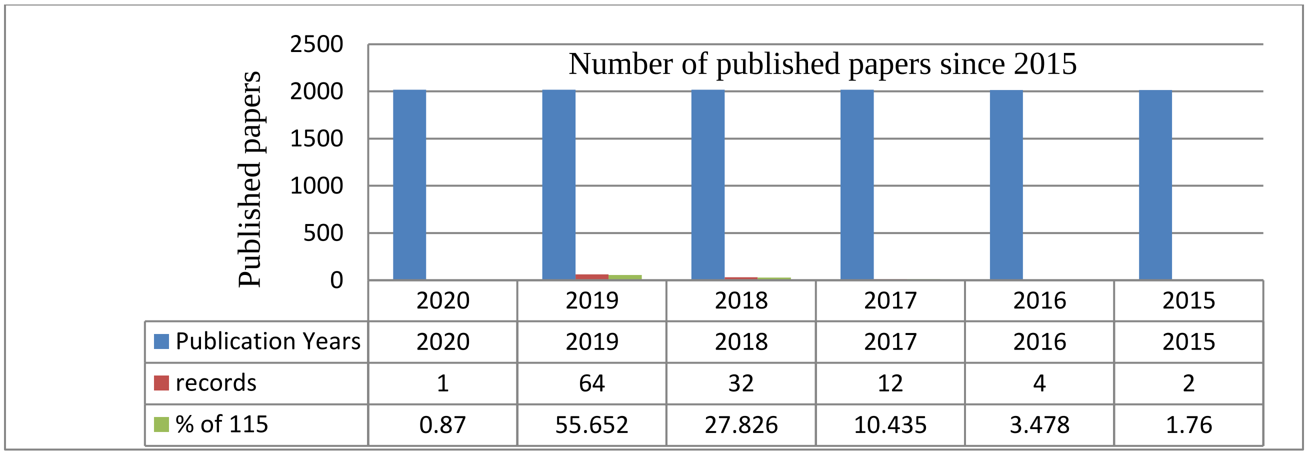
Table 2.2. Some recent primary studies in LCLU with DL in which most of our data were extracted

No.	Author(s)	Application Domains	RS Data Type	ML Methods used	Overall Result (in %)	Future work Recommendations
1	[31]	LU classification	MCM	Multiview DL	93.48	Performance improvement with combination of DNN cascading with other neural networks and use one view scale per network
2	[15]	Urban LCC	ISPRS (Vaihingen +Potsdam)	CNN	-	Performance with more networks
3	[35]	LCLU	Sentile 2A-EuroSAT	CNNs	98.57	LCLU change detection and improvement of geographical maps
4	[72]	RS image Scene	UCM, AID, NWPU-RESISC45	Descriptive CNNs	96.67,97.07,98.93 for each dataset	not clearly mentioned but pointed to more investigations on the AID and NWPU-RESISC45 datasets
5	[3]	Multi-label LC	UCM	CNNS with dynamic data augmentation	82.29	using pretrained models and fine-tuning the architectures for multilabel classification
6	[61]	LU	UCM and RSSCN7	Deep filter banks + Fisher vector	92.7 and 90.4 for both data	-
7	[5]	Earth observation: LU classifications	UCM	CNN: feature fusion	92.4	Pretrained networks on a larger scale experiment in satellite data
8	[36]	LU	UCM	Extream learning machine (ELM) in CNN	95.62	Accuracy improvement with integration operations such as overlapping maxpooling and cross-channel and reduce training time using GPU
9	[9]	LC	ISPRS, GID	Feature ensemble-FE-Net	68.08,65.16	-
10	[7]	Satellite image	UCM	AlexNet CNN	94	Utilizing appropriate DL methods to better features for target detection and scene classification tasks
11	[6]	Scene Classifications	UCM, RS19, and Brazilian Coffee Scenes	ConvNets	97.78, 91.0 and 94.45 respectively	Analyzing the relation between the number of classes in the dataset, the number of parameters in the ConvNet, and their impact in the discrepancy between fine tuning and full training processes.
12	[63]	Scene Classifications	UCM, Sydney	Gradient Boosting Random ConvNet	Varying for d/t learners	application of GBRCN on hyperspectral applications
13	[8]	Land cover	UCM	DNNS: CaffeNet, GoogLeNet, ResNet	97.6,97.1, 98.5	-
14	[4]	LULC	Indian Pines, San Francisco, Pavia, Flevoland	CNN	94.64, 98.70, 83.43, 98.51 respectively	Deeper architecture and parameters of the network, evaluating the efficiency of CNN with very large-scale images
15	[24]	RS Scene	AID, UCM, PatternNet	CNN: TL (InceptionV3 and VGG19 models)	Varying per model and datasets	Begin from deep models and then try to reduce model's size
	Proposed	**LCLU**	**UCM, SIRI-WHU, RSCNN7**	**CNN-FE, TL, Fine-tuning**	Various in hyperparameters	**Training DL models with GPU by considering hyperparameters on other larger datasets for performance improvements.**

2.5 Research Survey Analysis

We conducted our literature review using a systematic approach, which involves three stages: pre-review (planning), review (conducting), and reporting [77]. The first phase of planning includes identifying the needs of the review, formulating research questions, and developing and evaluating a review protocol. The second phase of the review is concerned with analyzing the state of the art. It could be used to search and identify the existing state-of-the-art research from online digital libraries and journals relevant to the area. Then we selected primary studies related to our area and extracted and synthesized the relevant data and parameters. The online digital archives used for searching primary studies are IEEE Xplore, Web of Science, MDPI, SpringerLink, Google Scholar, Taylor & Francis, and Wiley InterScience, with their corresponding journals. We also analyzed the number of publications in line with the study area and year.

According to the literature survey, LCLU in the RS imagery domain has been identified as the most recent study area, but AI has been identified with lower primary studies even though it is the most recent attention area, as observed in Figure 2.4 and Figure 2.5. Different scholars have been investigating DL algorithms using RS data for LCLU classification. Some of these are described in Table 2.2. Thus, this research problem has been proposed for investigating DL methods for LCLU classifications using RS data based on the literature survey.



	2020	2019	2018	2017	2016	2015
🟦 Publication Years	2020	2019	2018	2017	2016	2015
🟥 records	1	64	32	12	4	2
🟩 % of 115	0.87	55.652	27.826	10.435	3.478	1.76

Figure 2.4. Number of articles published in Web of Science Database since 2015 in Year wise

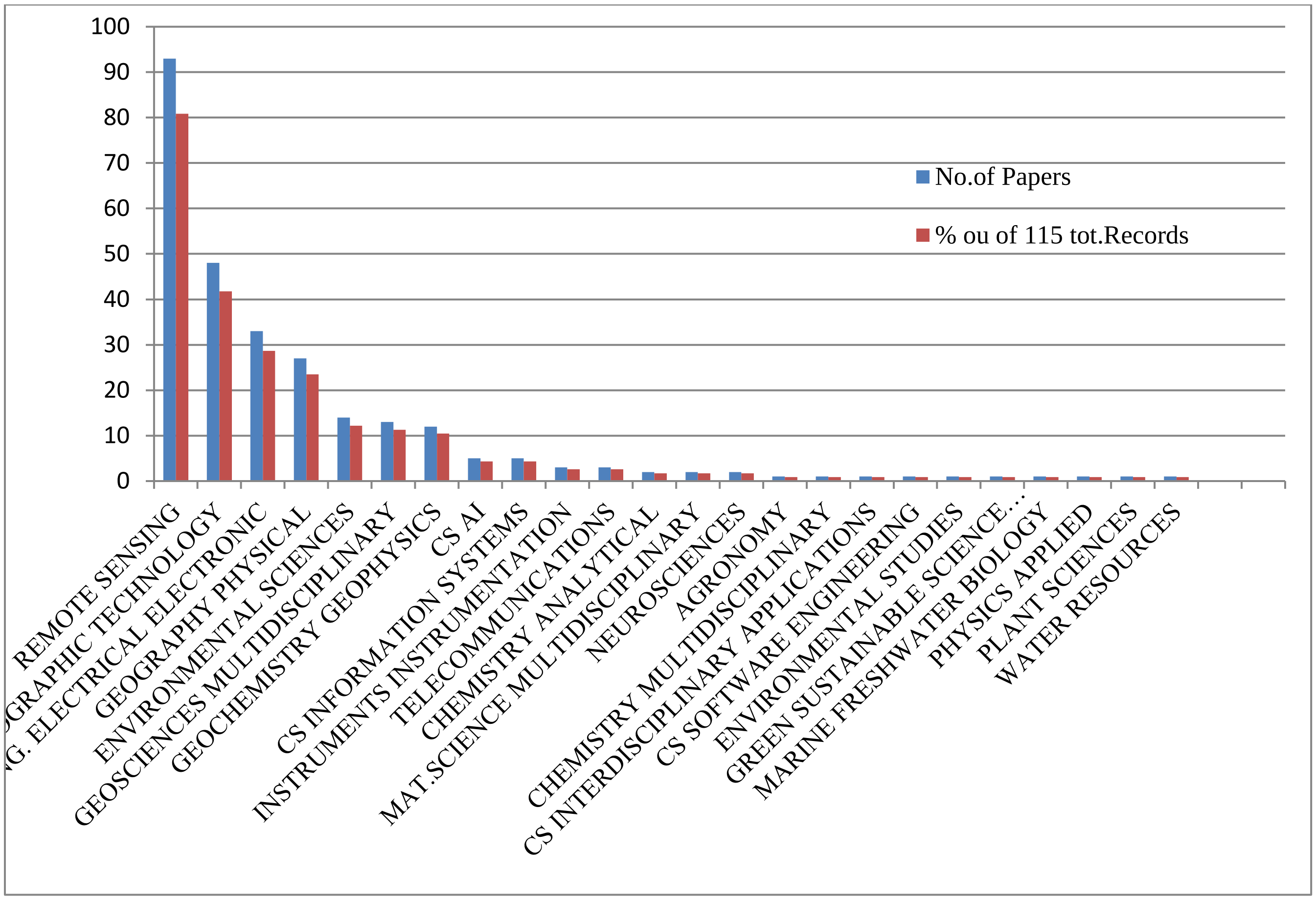


Figure 2.5. Number of articles published in the web of science database in our study area for a general search on [“deep learning” AND “land cover” AND “land use” AND “remote sensing”].

2.6 Research Gaps

In this digital age, a large amount of RS satellite imagery data is recorded on the earth by remotely sensed technology. Satellite images are significant information sources for the earth's environment, and the automatic classification of these images has always been an important research topic [78]. The LCLU classification problem takes into account RS satellite data with a higher level of spectral resolution and differences in location [5].

Classification is a fundamental task for RS imagery analysis [44]. Because of its high cost and labor-intensive nature, LCLU classification is a recent challenging task [3], [10].

DL is a hot research area in various domains. Validation of RS systems in DL by considering more urban areas in the world is important [68]. However, DL models are still facing several challenges for wide application, such as the fact that training samples and hyperparameter selection have large influences on classification performances. Lack of sufficient training samples or a small number of training samples, for instance, could be identified as the major limiting factor [1], [79][65], [66], [78], [1], [79], whereas training a network with a large number of samples could improve DL performance [67], [79]. Parameters selection such as depth of the features, number of hidden layer, size of learning rate, selection of activation and lose functions are also challenges for DL methods [42]. To increase the performance of DL models for LCLU classification, training deep network models to fit different image data sources would be considered [1], [34].
Therefore, from the primary study analysis, we identified the following research gaps.

1. Designing DL models using deeper architecture and hyperparameter optimizations of the network needs more investigation.
2. There aren't enough studies that compare the performance of DL models on different RS datasets.
3. The performance evaluation of DL methods for LCLU classification with different scale data sets in deep AI is still needed.

2.7 Chapter Summarization

In this chapter, we analyze the current primary studies using DL methods for LCLU classification in RS. From the analysis, we identified that DL methods for LCLU classification using RS are recent hot research areas in the field of ML and AI. RS data also have their own new challenges for DL due to the nature of multi-modal, geo-located, geodetic measurements with controlled quality that are time-dependent and face the big data challenge [29], [45]. RS imagery data classification [46] is facing exceptional and significant challenges. LCLU monitoring and management is also another challenge for making decisions. LCLU classification is a major challenge for RS analysis, with tremendous needs for working solutions and many potential applications [19], [23], [42], [44], [47]–[49].

To handle these challenges, DL models should be applied to satellite data analysis. The appearance of DL has provided a chance for analyzing big RS data [30]. DL methods are preferred over traditional ML methods as recent models for improving DL scene classification systems. Thus, DL methods get significant attention in LCLU classification in the RS field and obtain better improvements [50]. Improvements in DL for LCLU classification in RS data have been observed [18].

From our point of view, some of the challenges of DL in satellite imagery data classification research papers are listed in Table 2.2 with their corresponding suggestions. This literature review would help us gain insights into the advancement of DL methods for LCLU classification in RS and enable us to identify further investigations on more DL methods in RS image classification. Thus, we identified the research gaps accordingly, as we stated in Section 2.6.

RS meets DL [11]. DL methods are recommended by many researchers for solving RS challenges. Among DL methods, CNNs are the most convenient approaches for solving classification problems. To make CNN effective, further research will be necessary to help the public's approval and diffusion of CNNs [23] for RS classification. As a result, designing DL methods with CNN models for LCLU classification in RS imagery is our next research topic, with a variety of hyperparameters.

3. RESEARCH METHODS AND MATERIALS

In this section, we will describe the methods and materials, such as the datasets, development tools, and software packages, that are used to accomplish the research objectives.

3.1 Deep Learning Methods and their Parameters

DL is a ML algorithm used to address AI challenges in areas such as natural language processing (NLP), computer vision (CV), and expert systems (ES). The DL algorithms are implemented using Python and its frameworks. Python is a high-level object-oriented programming language that is used in DL frameworks like Keras, TensorFlow, PyTorch, and Caffe.

Why is DL used for LCLU classification in RS imagery? Or, why does RS imagery data use DL for LCLU classification? There are probably numerous reasons and answers, but two of the most crucial are its powerful ability to improve training performance and automatically extract features from large datasets. The research methods will show how to use tools and techniques for accomplishing the study. DL methods for LCLU classification in RS data have been successfully proven by researchers, as observed in the state-of-the-art, and the improvement of their performance has been profound [80]. DL approaches include CNN, RNN, and GANs. We would use CNN, CNN-based transfer learning and its fine tuning, and CNN-based pretrained networks in this study due to their capabilities for RS image classification.

3.1.1 Convolution Neural Networks (CNNs)

CNNs are feed-forward ANNs composed of interconnected neurons or nodes with learnable weights and biases for image classification. CNNs with deep layers have achieved unprecedented improvements in patch-based medium-resolution RS image classification [81]. The CNN method is used for classifying the hyper-spectral RS imagery data based on pixels based on dimensional windows. Image dimension could be h×w×d where h is the height, w is the width, and d is the depth (number of filters) of the RS image. CNNs consist of sequential layers in which the output of one layer is the input for other layers with various dimensions of computation. The CNN's sequence consists of input, convolution, pooling, normalization, and fully connected layers integrated with other DL hyperparameters. In this sequence, CNNs are convolved with filters (kernels) and pooled with a pooling (downsampling) layer.

1. The input layer: it is the entire input image layer with h×w pixels shape.
2. The convolution (Conv2D) layer

CNNs are a specialized kind of neural network with a linear operation for processing grid-like (2D) topology or pixel data. The CNN is built by the convolution layer using the basic units of the series learnable filter, or "kernel," and the input volume, or "matrix." The convolution layer receives the h×w image pixels and computes the perceptron with given f×f filters or kernels. The input volume is then convolved with the filter provided to produce a feature map or output volume. The size of the feature map is determined by the depth (number of filters or kernels), stride (number of pixels shifted over the input weight matrix), and zero-padding, as equated in (3.1).

The CNNs use the filter or kernel to extract the feature maps from the input image by using the convolution operation. Kernels are weights of the input images that are used to reduce the shape of the input images for hidden layer processing. Then the CNNs learn from the filters automatically and capture the spatial features (the arrangement of pixels) from an image. The spatial features enable us to identify the object by looking at the specific feature of the image, such as a forest with its specific feature of trees.

Convolution could be performed with valid convolution (no padding), same convolution (with zero padding at the edges), and stride (slide or shift) convolution. The mathematical computation of the output volume of the image in each layer could be calculated using the input volume (h×w), stride (S), padding (p) parameters, and filter size (f×f). The stride (S) of the filter (f×f) is the interval at which the filter jumps or shifts **S** number of shifts from the first elements in a pixel or in each spatial dimension, while padding (P) is the number of pixels added at the outer edges of the input image volumes (h×w). Filter is usually odd and smaller in size, that is 3×3, 5×5 and 7×7 with 1, 2, and 3 padding, respectively. However, using a very large filter size, such 11×11 and 13×13 is costly in terms of the learnable weights of the networks, and it is not recommended to use it in more modern DL architectures. Therefore, the cheaper filter sizes, such as 3×3 and 5×5 are best suited for learning weights or parameters, and we used 3×3 filters applied in sequence of the layers in this study.

In Keras DL tool, no padding for image border to valid convolution but **P** number of zeros padding for image border to same convolution. Thus, the mathematical computation of the output volume of the image, which is the input for the next layer with computational results, in each layer could be calculated using the input volume, stride, and padding parameters. Thus, the output volume ($n_{new} \times w_{new}$) of a layer could be computed with the mathematical computation in equation (3.1), and the number of zeros padding or same convolution could be computed using equation (3.2) when S = 1. When S>1, we could calculate the number of zeros padding using (3.3) to keep the input image size same as of the output image size. However, the value of p could not be in fraction and rounded it to the next higher integer when calculated in same padding. Moreover, if f >= S, take the maximum value of max(f, S) instead of f to calculate the output feature. The default values of the padding and stride is 0 or valid (no padding) and 1(the weight/filter matrix moves/shifts 1 pixel at a time only in horizontal and vertical edges), respectively.

$$\mathrm{h\_new} = \left\lceil \frac{(\mathrm{h} - \mathrm{f} + 2\mathrm{p})}{\mathrm{s}} \right\rceil + 1 \tag{3.1}$$

$$p = \frac{f - 1}{2} \tag{3.2}$$

$$p = \frac{((h - 1)s - h + f)}{2} \tag{3.3}$$

Suppose we have an input image of 6*6 and an initial weight/filter matrix of 3*3 that is used to extract some features from the input image. As a result, as shown in Figure 3.1, we can calculate the output of the new feature map (output image size) by adding the values of the element-wise multiplication of the weight/filter matrix and the sample corresponding highlighted 3*3 shift of the input image. So, the weights or filter are learned to pull out features from the original image that help the model make a correct prediction.

If we use a stride of 2 for the same input image and weights with same padding, we could get either more reduced output image size by reducing the input image dimensions (as shown in Figure 3.2) or same output image size as the input image size by adding much zeros padding on each edge of the input image (as shown in Figure 3.3). However, it is better to pad the input image with the required layers of zero padding around each edge of the input image for a higher number of strides (S>1) so that the output image size is not reduced in same padding. Thus, we pad the input image with one layer of zeros padding (single zero padding) at each edge of the pixel to get the same input and output image size. The added layer(s) is/are included by the 3*3 weight shifts in all round and the weight shifts or jumps 1 pixel at a time, as we observed in Figure 3.3. When a single zero of padding is added, a single stride or shift of weights or filters is used to keep the size of the input image. So, we restored more information from the border and got the same input and output image sizes.

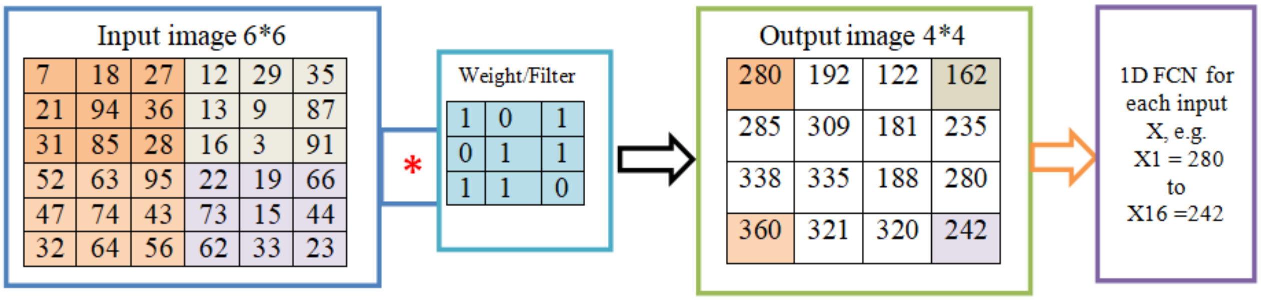


Figure 3.1. Feature maps using weight/filter matrix

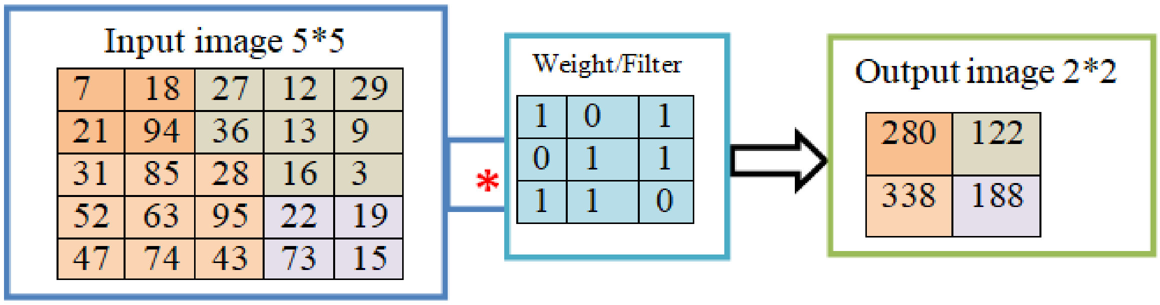
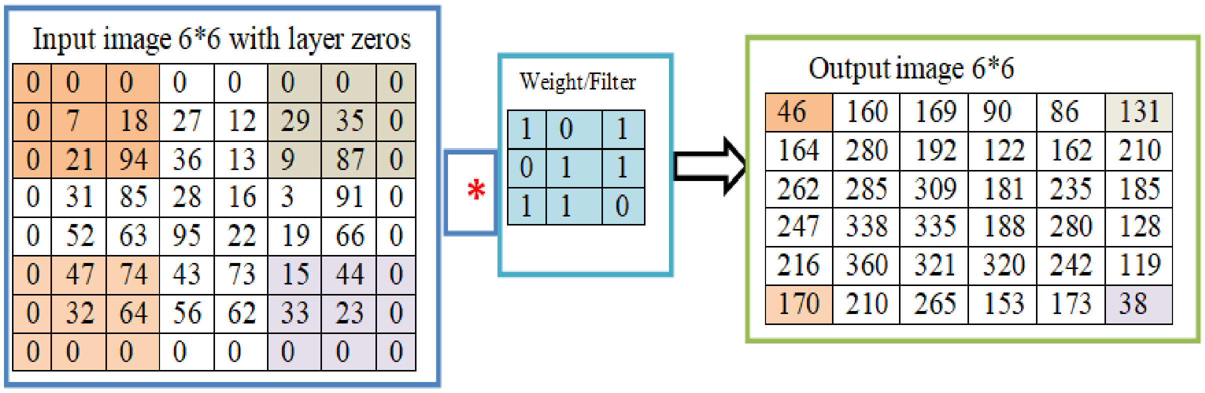


Figure 3.2. Reduced output image (feature map) using weight/filter matrix with stride of 2 and valid padding



0	0	0	0	0	0	0	0
0	7	18	27	12	29	35	0
0	21	94	36	13	9	87	0
0	31	85	28	16	3	91	0
0	52	63	95	22	19	66	0
0	47	74	43	73	15	44	0
0	32	64	56	62	33	23	0
0	0	0	0	0	0	0	0

1	0	1
0	1	1
1	1	0

46	160	169	90	86	131
164	280	192	122	162	210
262	285	309	181	235	185
247	338	335	188	280	128
216	360	321	320	242	119
170	210	265	153	173	38

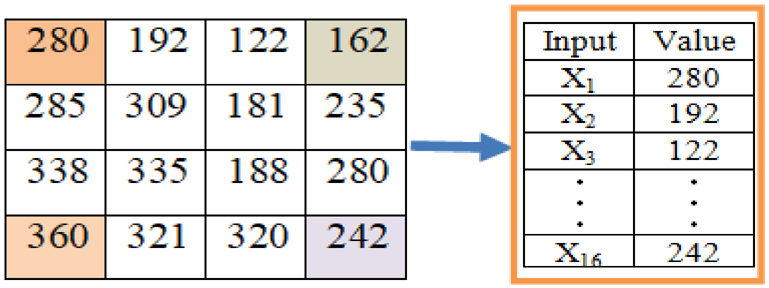
Figure 3.3. Feature maps using 3*3 weight/filter matrix with stride of 2 and same padding (1 layer = 1 padding)

Therefore, convolution is a dot product of a provided filter (or kernel). The symbol "*****" indicates the convolution, and it can be represented in different ways in implementation languages, such as "***conv-forward***" in Python, "***tf.nn.conv2d***" in TensorFlow, and "***Conv2D***" in Keras. The output depth or layer of the image would be the same as the number of filters (or kernels) applied on the network. For instance, suppose we have an input image of size 256*256*3 and we apply 32 convolutions (filters) of size 3*3*3 with stride of 2 in both valid and same paddings. Then according to (3.1), the output volume of the image could 127.5*127.5*32 and 128.5*128.5*32 for valid and a single zero paddings. Nevertheless, the output dimensions could not be in fractions, and they must be converted into integers by truncating towards zero for valid and rounding the next integer for same padding since the image size is halved with a stride of 2. Therefore, the output dimensions of the image would be 128*128*32 for both valid and a single zero padding.

In addition, the output of the convolution layer is a 2D matrix. After the convolution process has been completed, the fully connected layer will follow. However, the fully connected layer accepts a 1D image, i.e., in vector form. Therefore, the linear transformation function could be applied in terms of weights and bias to convert the 2D matrix from the output feature of the convolution layer or the input image (X) of the fully connected layer into the 1D vector form of the fully connected layer, as indicated in Figure 3.1 and Figure 3.4.

The deep neural network models are the learnable algorithms that are able to find the DL parameter values of the weights and bias (W, b) after training them from the input images using optimization techniques. Linear transformation is used to transform matrix form (m, n), where m = h*w is the number of features or inputs (X_1,X_2, … X_m) for this layer and n is the number of neurons (filters or depth) in the layer, with the provided weight and constant bias matrixes to vectorization form. The linear transformation process from 2D to 1D could be calculated using (3.4) in terms of the input image (X), transposed matrix weight (W^T), and constant bias (b), and the feature map has been transformed into 1D, as shown in Figure 3.4. The number of weights depends on the corresponding inputs, while the number of biases depends on the number of neurons. For instance, if we have 3 neurons (aka perceptron or nodes) in the fully connected layer, the resulted feature map depicted in Figure 3.1 matrix shape becomes (16, 3) and the linear transformation equations could be used to compute the transformed result. This transformed result could be used later for any nonlinear activation function. Thus, we could compute a linear function using the 3 biases and 16 weights since there are 3 neurons and 16 input pixels, or columns, respectively.

$$Z = W^T \mathrm{X} + \mathrm{b} \tag{3.4}$$



280	192	122	162
285	309	181	235
338	335	188	280
360	321	320	242

Input	Value
X_1	280
X_2	192
X_3	122
⋮	⋮
X_{16}	242

Figure 3.4. Linear transformation from 2D to 1D i.e., from convolutional layer to fully connected layer

Based on the CNN parameters of the input (X), the randomly initialized weight (W), and the bias, the linear transformation equation could be applied, and a 1D vector of the weighted output has resulted as shown in the following schema. With this result, the non-linear transformation function could also be applied to get the classifier. According to Figure 3.4, we have sixteen input features X with their consecutive weights, and the assumed number of neurons is three, i.e., the number of biases becomes three as provided below.

$$X = \begin{pmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ \cdot \\ X_{16} \end{pmatrix} \quad W = \begin{pmatrix} W_{11}\ W_{12}\ W_{13} \\ W_{21}\ W_{22}\ W_{23} \\ \cdot \\ \cdot \\ \cdot \\ W_{161}\ W_{162}\ W_{163} \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

Now, using the linear transformation equation (3.4), we can calculate the transformed Z result as follow.

$$Z = \begin{pmatrix} W_{11}\ W_{21}\ \ldots W_{161} \\ W_{12}\ W_{22}\ \ldots\ W_{162} \\ W_{13}\ W_{23}\ \ldots\ W_{163} \end{pmatrix} * \begin{pmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ \cdot \\ X_{16} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 + \\ b_3 \end{pmatrix}$$

$$Z = \begin{pmatrix} W_{11}X_1 + W_{21}X_2 + \ldots + W_{161}X_{16} \\ W_{12}X_1 + W_{22}X_2 + \ldots + W_{162}X_{16} \\ W_{13}X_1 + W_{23}X_2 + \ldots + W_{163}X_{16} \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

In the convolution process, the number of parameters (params) could be calculated using (3.5) and (3.6). Parameters are internal variables learned or updated automatically based on the data during the training process that determine the model's behavior. These parameters are used to represent the weights or coefficients that determine how the model maps input data to output prediction and they are adjusted through an optimization algorithm to minimize the difference between predicted and actual values with (3.8). There are convolution (Conv2D) layer and dense layer parameters, which are equated in (3.5) and (3.6), respectively.

$$\text{Conv2Dparam\#} = \text{\#Outchannel} * (\text{\#Inchannel} * \text{Fheight} * \text{Fwidth} + 1) \qquad (3.5)$$

$$\text{Denseparam\#} = \text{\#outputchannel} * (\text{\#inputchannel} + 1) \qquad (3.6)$$

The total parameter numbers of the model are the summations of the calculated results from the Conv2D and dense layers. We designed CNN-FE model with four Conv2D layers that calculate the number of parameters for those layers in the same norm by (3.5) and two dense layers (3.6). However, the calculation formula for dense parameters differs from Conv2D, as equated in (3.6). The number 1 means the bias associated with each filter for learning.

We could calculate the total parameter number using (3.5) and (3.6). The number of parameters for all MaxPooling2D and Flatten layers, on the other hand, is zero because these layers do not learn anything from weights or filters, in addition to the built model.

3. Pooling (Downsampling) Layer: This layer is used to resize and down sample spatial representations, which are then followed by convolution operations. The pooling layer could be used for reducing the number of weights or the number of parameters and controlling overfitting. In DL, there are four pooling operations: maximum, minimum, average, and adaptive pooling. We used the common max pooling technique. Max pooling could pick the most activated feature and could be used to reduce overfitting and reduce the number of parameters.
4. Normalization Layer: This layer normalizes over local input regions to aid in generalization.
5. Fully-connected Layer (FCN): feature classifiers located at the network's final two layers. They include flatten layers, dense layers, and an output layer at the end. Perceptrons in an FCN are fully connected to all previous layer activations. Each layer could include various hyperparameters.

3.1.2 Deep Transfer Learning and Fine-tuning

DL, which is gaining popularity for solving classification problems, was used to extract earth features from remotely sensed imagery data in order to manage the earth's land for proper deep classification system utilization. DL algorithms have grown in popularity due to their ability to automatically learn from large datasets [56], [80], [82]–[86]. They are widely used for their higher performance and accuracy [5], [56], but they are more time-consuming to train and result in overfitting [11], [56].

TL is a domain adaptation DL model that uses the previously developed model to create new DL tasks. The developed model (pretrained) transfers its capabilities from existing experiences to the new one to enhance learning capabilities. In this case, the source domain model (which has already been trained) shares its experiences with the new model, which can learn from and adapt the pretrained model's knowledge. So, the name "transfer learning" comes from this process of sharing knowledge or experience. In this case, we can call TL a domain adaptation model.

TL gets more attentions for reducing the training time and the dependence on large amounts of training datasets [8], [56], [57]. We are impressed by TL's efficient training time, and we are also motivated to design TL based on the recommended work suggested by [3]. The TL algorithm is a pretrained CNN-based DL model with non-trainable weights of the pretrained model, i.e., setting the "***pre_trained_model.trainableor tf.keras.applications.*.trainable = False***" where * is any pretrained network, such as AlexNets, ResNet, InceptionNets, VGGNets, EfficientNet, and MobileNet. While applying TL, we simply allow the last dense layers to be trained for our new model. The tf.keras.applications.* enables the model to have the pixel values in a specific range like [0, 1] from the input images. If we allow all the pretrained layers to be trained i.e. setting the "***pre_trained_model.trainableor tf.keras.applications.*.trainable= True***" the pretrained weights are able to be used for the neural network's initial weights. In this case, we are transforming the DL technique from TL to a new pretrained network called fine-tuning. Therefore, TL uses random initializations of the deeper layers or the weights from the pretrained ImageNet network. Whereas the fine-tuning technique initializes the deeper layers with values from the pretrained ImageNet.

TL adapts the existing pretrained classifier and learns on the top of the fully connected layers of the entire network. TL is widely used for RS image classification [11], [58], [59] because RS images are essential for LCLU classification in DL approaches [60]. TL is used to create a DL model from the existing problem pretrained models. We have used ResNet50V2, VGG19, and InceptionV3 pretrained models.

Fine tuning is a specific technique with in TL where the pretrained model’s weights are further adjusted or fine tuned on the new dataset (task), whereas TL is used to train the final layers of the pretrained model’s weights are usually kept fixed. Unlike TL, fine tuning is modifying not only the final layers but also some other earlier layers of the pretrained model as we described it earlier as the trainable layer’s weight is ‘**True’**.

3.1.3 Deep Pretrained Networks

Pretrained networks are DL approaches that have already been trained on large datasets. The ImageNet large scale visual recognition challenge (ILSVRC) or simply "ImageNet" benchmark dataset is one of the large datasets on which most pretrained networks have been trained. Most of the pretrained networks have been trained on the large ImageNet dataset. The "ImageNet" dataset, which consists of over 14 million images and 1000 categories, was introduced and contributed by [87]. Pretrained networks consume less computational resources and training time (enabling faster training), improved performance and enhanced generalization capabilities

Most pretrained networks are available in the Keras DL development tool, and they can be loaded using the tensorflow "tf" package as "tf.keras.applications.*," where "*" is any pretrained network. The pretrained networks serve as the foundation for TL and fine-tuning DL networks. Currently, various modern DL architectures are being developed for DL applications in various domains. Most of these pretrained DL architectures were introduced by the Google research team, as shown in Table 3.1. Each pretrained network releases different versions from time to time. For instance, for VGGNet, versions VGG16 and VGG19; for GoogleNet (InceptionV1), versions InceptionV2 to InceptionV4; and for MobileNet, versions MobileNetV1 to MobileNetV7, are the released versions by researchers. We used the latest version of each DL network in this study. To sum up, the DL methods used in this study are the deep CNN, CNN-based TL, and pretrained networks.

Table 3.1. Pretrained Network DL architectures

Pretrained network	#Maximum parameter	#Layers	Year	Introduced and contributed by
AlexNet	60 million	8	2012	[88]
VGGNet	20 million	19	2014	[89]
GoogleNet (InceptionNet)	22 million	22	2014	[90](Google team)
ResNet	23 million	152	2015	[91]
MobileNet	4 million	28	2017	[92] (Google team)
EfficientNet	64 million	-	2019	[93] (Google team)

3.1.4 DL Hyperparameters and Optimization Techniques

Most DL algorithms are implemented with numerous hyperparameters that affect the model's performance in terms of computational resources (the time and memory cost for running the training). The DL hyperparameters and optimization techniques are the DL strategies that are used to find the optimal performance or minimum error. Hyperparameters are the external configuration choices or settings that define how the learning algorithm operates and controls the learning algorithm's behavior. The term hyperparameters are differ from parameters. Parameters internal variables learned (updated) automatically based on the data, while hyperparameters are external variables that are not learned (updated) from the data, simply they are set manually and often require setting or adjusting to optimize the model performance.

The optimization technique includes the regularization technique, which limits the values of the hyperparameters for optimization, training, or learning the model. Many DL strategies are being considered in order to reduce test error. DL algorithms include optimization techniques in many settings.

To build DL models, any DL algorithm would have different technical optimal hyperparameter requirements. The optimization techniques enable the model to become better by minimizing errors. The appropriate or optimal hyperparameters can be chosen either manually or automatically (the default). These optimization techniques include optimizer (Adam), learning rate, dropout, early stopping, number of epochs (iteration), back propagation, and the like. Some of the important requirements are going to be described in the following sections.

3.1.4.1 Learning rate (LR)

The learning rate is used to facilitate the ability of the model to learn from the given data. It controls the step size at each iteration or epoch during the optimization process and determines how quickly or slowly the model learns from the data. It has various values such as 0.01, 0.001, and 0.0001. However, if the larger LR is used, the fluctuation of training and learning could happen, and the pretrained weights could be lost. If the LR is smaller, the convergence of the training and validation losses to zero will be too slow. Therefore, the appropriate LR value is advisable to be used in building DL models. So, we used the TR of 0.001 and 0.0001 to optimize our model with the Adam optimizer.

Adam (Adaptive Momentum Estimation), which is the recent optimization technique in DL, is the combination of adaptive delta algorithm (AdaDelta), Root Mean Square Probability (RMSProp), and momentum. This optimization technique was introduced by [94] in 2015.

3.1.4.2 Dropout

Dropout is an optimization technique that is used to drop out randomly selected neurons or nodes with a given percentage probability. This hyperparameter is used to reduce overfitting during the training of the model but is not used in evaluating the model. The percentage values expressed in decimal form are usually expressed as 0.2, 0.3, 0.4, and 0.5. In this study, we used 0.5 (i.e., 50%) to reduce the overfitting of the training.

3.1.4.3 Loss functions (Error function)

The loss function is the difference between the actual and predicted output values in the DL model. The minimized error value makes the model better. To minimize the error function, we could use various DL optimization techniques, such as Adam, epochs, learning rate, dropout, and back propagation. In DL modeling techniques, there are two loss functions: regression losses and classification losses. Regression loss includes mean square error (MSE-L2) loss and mean absolute error (L1) loss, and classification loss includes binary classification loss or binary cross-entropy and multi-class classification loss or multi-class cross-entropy.

Cross-entropy loss (aka log loss) is used to measure the performance of the model that has an error probability value between 0 and 1, which could be calculated using equation (3.7). We used a multi-class cross-entropy loss function since our class is multi-classes of the RS images.

$$E = -\sum_{k=1}^{N} \mathrm{O_k} \log(\mathrm{Y_k}) \tag{3.7}$$

where E is the cost, error, or loss function of the model, N is the number of total classes, k is the number of neurons from 1 to N output neurons, O is the actual output, and Y is the predicted output. However, the error function is a function of the weights and bias. Therefore, the backpropagation algorithm of the minimization of cost function or error function (E(W, b)) could be used to find the difference between the actual and predicted outputs, as equated in (3.8).

$$\mathrm{E}(\mathrm{W}\,,\mathrm{b}) = \frac{1}{2}\sum_{k=1}^{N}(O_k - Y_k)^2 \tag{3.8}$$

3.1.4.4 Activation functions

In the convolution process, the output feature 2D array is converted into a 1D array in the fully connected layer, and each individual pixel value is considered a feature of the image. In the fully connected layer, the linear and non-linear transformation operations are applied. We described the linear transformation operation earlier. In addition to this transformation, the very vital non-linear transformation component known as the activation function is applied at each layer of the neural network depending on its availability. Therefore, we described the most important activation functions that are non-linear transformations or functions in this section to have some know-how idea about them.

Activation functions are used to update the weight values for the learning capability of the model. There are various linear and non-linear activation functions used in DL. For all non-linear functions, we can easily backpropagate the forward propagation process. DL CNNs have various activation functions, which should be non-linear as linear functions have a constant derivative. The most commonly used activation functions are sigmoid or logistic functions, tangent hyperbolic (tanh), rectified linear unit (Relu), and softmax.

We used the Relu at the entire convolutional layer to activate the weights in each convolution process and the Softmax at the output layer since it is reliable for our multiclass classification problem. Relu is used after each convolution layer as they are faster at training the network without considering accuracy. The softmax function is a feature classifier, and it introduces a probability score for each class. The class with the highest probability score is predicted to be our predicted class. This probability score will be used for performance evaluations.

3.1.4.5 Sigmoid

This function precedes any ranged numbers as inputs and generates the output value in the range of 0 to 1 with an "S" shaped curve. We represented the sigmoid function and its derivative (gradient) with the input x mathematically in equations (3.9) and (3.10), respectively, and graphically in Figure 3.5.

$$f(x) = \frac{1}{1 + exp^{(-x)}} = \frac{exp^{x}}{1 + exp^{x}} \tag{3.9}$$

And its derivative (gradient) is:

$$f'(x) = sigmoid(x).\left(1 - sigmoid(x)\right) = f(x).[1 - f(x)] \tag{3.10}$$

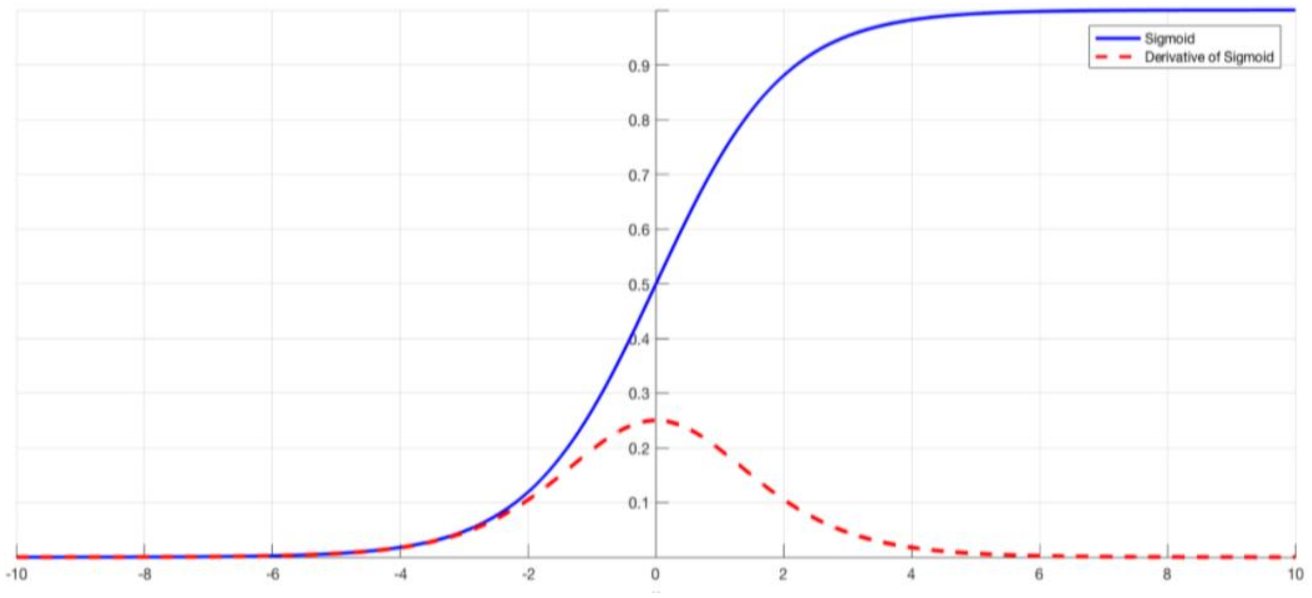


Figure 3.5. Graphical representations of the sigmoid function and its derivative (gradient)

The gradient function of the sigmoid values is approaching 0. Thus, the probability of the neural network's learning ability could be very low. The function has been used in the output layers of

DL architectures as well as the loss function in binary classification problems and logistic regression neural network applications.

3.1.4.6 Tangent Hyperbolic (tanh)

It is somewhat similar to that of the sigmoid function, but while the sigmoid function takes input values between 0 and 1, the tanh function takes input values between -1 and 1, as depicted in Figure 3.6. The mathematical representation of the tanh function and its derivative is quantified in (3.11) and (3.12), respectively.

$$a = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{2}{1 + e^{-2z}} - 1 \qquad (3.11)$$

In Python with numpy (np) library, this activation function formula can be coded as:

a = (np.exp(z)-np.exp(-z))/(np.exp(z)+np.exp(-z)).

And the **derivative of tanh(z) could be:**

$$\tanh'(\mathrm{z}) = \frac{\mathrm{da}}{\mathrm{dz}} = 1 - a^2 \qquad (3.12)$$

In Python, this derivative activation function can also be coded as **da = 1-a**2.** The graphical representation of tanh and its derivative is plotted in Figure 3.6.

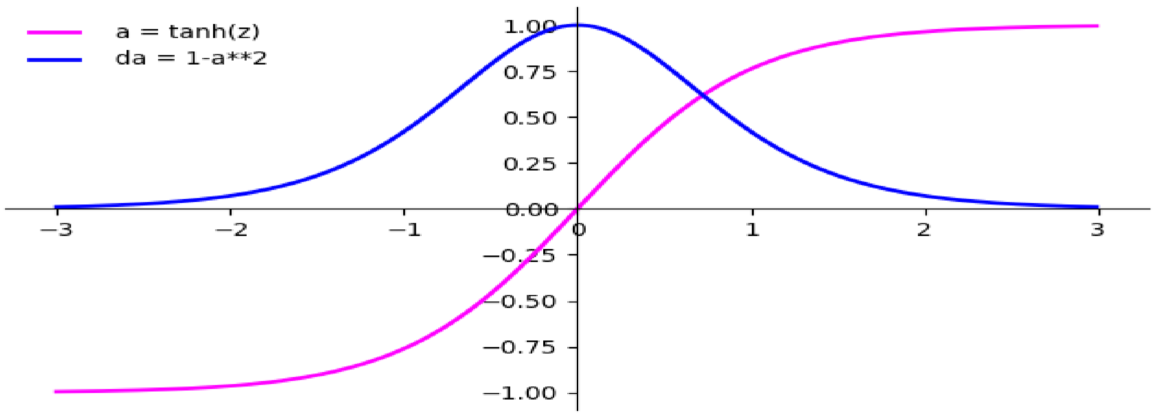


Figure 3.6. Tanh Function and its Derivative

In Figure 3.6, the tanh function produces the zero-centered output, and it is usually applied in the hidden layers since its variety is between -1 to 1, i.e., -1 < variety < 1. Thus, the tanh is used to center the data by approaching the mean of the hidden layer close to 0. In this case, the next ability is much easier to learn and optimize than the sigmoid function. In the derivative function with a blue-colored graph, the output value approaching 0 for the real number range is less than -3 and greater than 3. The variety is between 0 and 1, i.e., 0 < variety < 1.

3.1.4.7 Rectified linear unit (Relu)

Relu has been used in almost all DL models nowadays, and its performance is better than the sigmoid function. As a result, we used this function to classify RS imagery data for our deep TL model. Relu returns x if x > 0 and 0 otherwise for any real number x. The mathematical and graphical representation of Relu is represented in equation (3.13) and Figure 3.7.

$$f(x) = max(0, x) = \begin{cases} x_i\,, if\ x > 0 \\ 0, if\ x < 0 \end{cases} \tag{3.13}$$

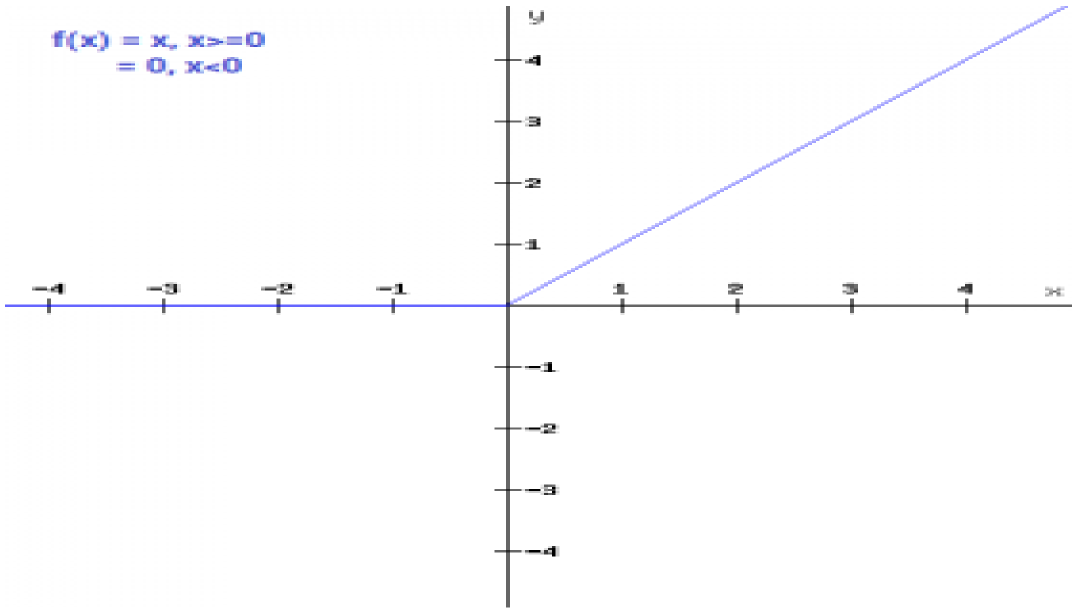


Figure 3.7. Relu function graphical representation

From Figure 3.7, for the negative input value of x, the result becomes zero, which implies that the neurons with the negative values are not activated except the neurons with positive values. The algorithm for Relu could be:

```
def relu_func(x)
if x>0
print(x)
else:
print(0)
```

In this algorithm, if we call the function relu_func(4) and relu_func(-4), the output is 4 and 0, respectively (also Figure 3.7).

The derivative, i.e., the gradients of Relu function, is represented in equation (3.14) as:

$$f'(x) = \begin{cases} 1, if\ x \geq 0 \\ 0, if\ x < 0 \end{cases} \tag{3.14}$$

And the graph of the derivative (gradient) of the relu function has been plotted in Figure 3.8. The gradient's slope is constant i.e. ether 1 ∀x, x >= 0 or 0 ∀x, x<0.

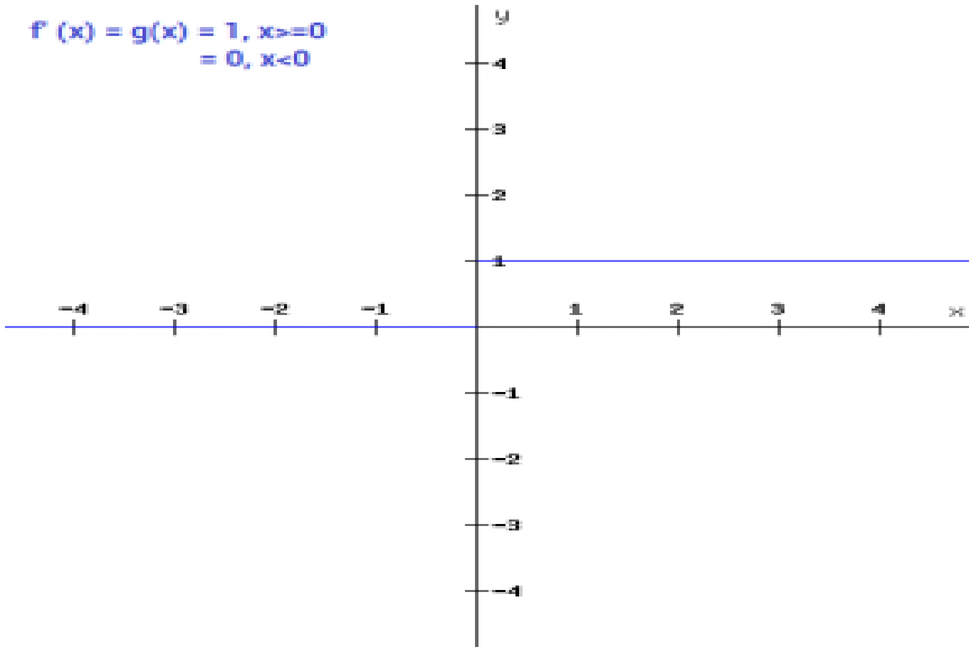


Figure 3.8. Graph of the gradient of relu function

According to Figure 3.8, as stated earlier, for the negative input value of x, the gradient value is always zero. This concept implies that the dead neurons with the negative values that never get activated are created because weights and biases were not updated for some neurons during the backpropagation process. Backpropagation is an algorithm in ML and AI that is used to fine-tune the computational weight functions and to update the accuracy of the model or output in a chain rule. It is used to calculate the gradient descents of the loss function with respect to the given weights in the ANNs. For the positive input value of x in the other cases, the gradient value is always one. In this case, some neurons are activated, and the capability of learning ability is taken place.

In relu, errors could propagate easily, and multiple layers of the neurons have been activated. The mathematical operation is simpler than that of the tanh and sigmoid functions because only a few neurons are activated at a time to make the network competent and stress-free for computation.

3.1.4.8 Softmax (softargmax or normalized exponential function)

It is used to predict the class having the highest probability in multi-class classification problems for the input labels. We used this function in this study since our RS imagery data is a multi-class classification problem.

The softmax output is between 0 and 1, and the sum of each class probability is **1.0**. If some **N** elements of the input vector are **N**<0 or **N**>1, they would be between (0, 1) after using the softmax function. Its equation $\boldsymbol{f}(\mathbf{z}_{i,j})$ over N classes is computed in the equation (3.15) given.

$$s = f(\mathrm{z}_{i,j}) = \frac{\exp(\mathrm{z}_i)}{\sum_{j=1}^{N} exp(\mathrm{z}_j)} = \frac{\mathrm{e}^{\mathrm{z}_i}}{\sum_{j=1}^{N} \mathrm{e}^{\mathrm{z}_j}} \tag{3.15}$$

The softmax function can be written in vector forms as:

$$\mathrm{T(s)}: \quad \begin{pmatrix} \mathbf{s_1} \\ \mathbf{s_2} \\ \cdots \\ \mathbf{s_n} \end{pmatrix} \rightarrow \begin{pmatrix} \mathbf{t_1} \\ \mathbf{t_2} \\ \cdots \\ \mathbf{t_n} \end{pmatrix}$$

Per each element, the softmax function looks like this: $\mathrm{t}_j(\mathrm{s}) = \frac{\exp(\mathrm{s}_j)}{\sum_{k=1}^{N} exp(\mathrm{s}_j)} = \frac{\mathrm{e}^{\mathrm{s}_j}}{\sum_{k=1}^{N} \mathrm{e}^{\mathrm{s}_k}}$,

$$where, \forall k = 1,2, \dots, N$$

The derivatives of softmax can be calculated using matrix forms of equation (3.16).

$$\frac{\boldsymbol{\partial}\mathbf{T}}{\boldsymbol{\partial}\mathbf{S}} = \begin{bmatrix} \frac{\partial \mathrm{t}_1}{\partial \mathrm{s}_1} \frac{\partial \mathrm{t}_1}{\partial \mathrm{s}_2} \cdots \frac{\partial \mathrm{t}_1}{\partial \mathrm{s}_n} \\ \cdots \\ \frac{\partial \mathrm{t_n}}{\partial \mathrm{s}_1} \frac{\partial \mathrm{t_n}}{\partial \mathrm{s}_2} \cdots \frac{\partial \mathrm{t_n}}{\partial \mathrm{s}_n} \end{bmatrix} \tag{3.16}$$

And the derivative of softmax could be used to compute the error for every i and j elements as:

$$\frac{\partial \mathrm{t_i}}{\partial \mathrm{s_j}} = \frac{\partial \frac{e^{\mathrm{s_i}}}{\sum_{k=1}^{N} \mathrm{e}^{s_k}}}{\partial \mathrm{s_j}}$$

Let us apply the quotient rule as:

$$f(x) = \frac{g(x)}{h(x)} \text{and} f'(x) = \frac{g'(x)h(x) - g(x)h\prime(x)}{(h(x))^2}$$

In this equation, $g(x) = e^{\mathrm{s_i}}$and $h(x) = \sum_{k=1}^{N} \mathrm{e}^{s_k}$. Thus, the derivative of g(x) and h(x) is:

$g'(x) = \begin{cases} e^{\mathrm{s_i}}, & if\ i = j \\ 0, & otherwise \end{cases}$, and $h'(x) = e^{\mathrm{s_j}}, \forall k = 1,2 \dots n$, respectively.

$$\text{Therefore, } f'(x) = \frac{g'(x)h(x) - g(x)h'(x)}{\left(h(x)\right)^2}$$

$$= \frac{e^{\mathrm{s_I}} \cdot \sum_{k=1}^{N} \mathrm{e}^{s_k} - e^{\mathrm{s_i}} . e^{\mathrm{s_j}}}{(\sum_{k=1}^{N} \mathrm{e}^{s_k})^2}$$

$$= \frac{e^{\mathrm{s_I}}}{\sum_{k=1}^{N} \mathrm{e}^{s_k}} . \left(\frac{\sum_{k=1}^{N} \mathrm{e}^{s_k} - e^{\mathrm{s_j}}}{\sum_{k=1}^{N} \mathrm{e}^{s_k}} \right)$$

$$= \frac{e^{\mathrm{s_i}}}{\sum_{k=1}^{N} \mathrm{e}^{s_k}} . \left(1 - \frac{e^{\mathrm{s_j}}}{\sum_{k=1}^{N} \mathrm{e}^{s_k}} \right), \forall ij, i = j$$

$$and\ \frac{e^{\mathrm{s_i}}}{\sum_{k=1}^{N} \mathrm{e}^{s_k}} . \frac{e^{\mathrm{s_j}}}{\sum_{k=1}^{N} \mathrm{e}^{s_k}}, \forall ij, \mathrm{i} \neq \mathrm{j}$$

$$\therefore \quad f'(x) = \begin{cases} \mathrm{t_i} . \left(1 - t_{\mathrm{j}}\right), & if\ i = j \\ -\mathrm{t_i} . t_j, & otherwise \end{cases} \tag{3.17}$$

By using the Kronecker delta function,

$\delta_{ij} = \begin{cases} 1\ if\ i = j \\ 0, otherwise \end{cases}$, $\frac{\partial \mathrm{t_i}}{\partial s_j} = \mathrm{t_i}(\delta_{ij} - \mathrm{t_j})$. By applying this equation into equation (3.16), finally we got the derivative of softmax function can be calculated using equation (3.18).

$$\frac{\partial \mathbf{T}}{\partial \mathbf{S}} = \begin{bmatrix} \mathbf{t_1}(\boldsymbol{\delta}_{11} - \mathbf{t_1}) \mathbf{t_1}(\boldsymbol{\delta}_{12} - \mathbf{t_2}) \ \dots \ \mathbf{t_1}\left(\boldsymbol{\delta}_{1j} - \mathbf{t}_j\right) \\ \mathbf{t_2}(\boldsymbol{\delta}_{21} - \mathbf{t_1}) \mathbf{t_2}(\boldsymbol{\delta}_{22} - \mathbf{t_2}) \ \dots \ \mathbf{t_2}(\boldsymbol{\delta}_{2j} - \mathbf{t}_j) \\ \dots \\ \mathbf{t_i}(\boldsymbol{\delta}_{i1} - \mathbf{t_1}) \ \mathbf{t_i}(\boldsymbol{\delta}_{i2} - \mathbf{t_2}) \dots \mathbf{t_i}(\boldsymbol{\delta}_{ij} - \mathbf{t}_j) \end{bmatrix} \tag{3.18}$$

When we substitute $\delta_{ij}$by 1 or 0, the derivative of the softmax function in equation (3.18) can also be simplified as in equation (3.19).

$$\frac{\partial \mathbf{T}}{\partial \mathbf{S}} = \begin{bmatrix} t_1(1-t_1) & -t_1.t_2 & \dots & -t_1.t_{j-1} & -t_1.t_j \\ -t_2.t_1 & t_2(1-t_2) & \dots & -t_2.t_{j-1} & -t_2.t_j \\ & & \dots & & \\ -t_i t_1 & t_i(1-t_j) & \dots & -t_i.t_{j-1} & -t_i.t_j \end{bmatrix} \tag{3.19}$$

The algorithm for simplified derivative matrix (equation (3.19)) could be represented as:

```
matrix = np.diag(t = np.arry([x_ijOriginalValues])
fori in range(len(matrix)):
for j in range(len(matrix)):
ifi == j:
matrix[i][j] = t[i] * (1-t[i])
else:
matrix [i][j] = -t[i]*t[j]
```

3.1.5 Fittings in Deep Learning

Fitting is a DL technique that is used to classify categorical features in different patterns. It has effects on the performance of the DL in terms of bias and variance. There are three categories of fittings observed in the ML and DL model performances, namely under-fitting, optimal fit, and overfitting. Underfitting and overfitting are the two DL challenges that affect the performance of the model. Underfitting occurs when high bias or low variance errors exist, the model does not fit the training set correctly, and the model is not able to generalize to a new sample image. It also occurs when the model generates high training errors on the training set. Overfitting occurs when there are high variance or low bias errors and the model perfectly fits the training set, and the model may be unable to classify a new unknown sample image. It also occurs when the gap between the training error and the validation or test error is too large. Therefore, overfitting occurs on the validation or test error with its high variance, while underfitting occurs on the training error with its high bias.

Bias and variance in any ML model are unavoidable, but it is possible to make them optimal. When a model has low bias and variance and performs well on new sample images, it is said to be “optimally fitting.” However, the model could be doing poorly on the training set because of high bias, and its performance on the validation or test set could be worse because of high variance. In this case, the model becomes overfitting and underfitting simultaneously, and it is hard to apply it to a specific task, such as classification, detection, or recognition.

Bias and variance contribute to errors. Therefore, bias and variance are errors of the classification model, and high errors on the training data may result in under-fitting while nearly zero errors on the training data may result in overfitting. In this idea, a complex model (overfitting) could be built with high variance and low bias, while a simple model (under-fitting) could be built with low variance and high bias. To calculate the total error, we might use various mathematical formulas depending on the error metrics applied. For instance, we equated (3.7) and (3.8) for cross-entropy error metrics and (3.20) for mean squared error metrics.

To overcome such problems, minimizing the error by utilizing optimization techniques is the focus, but not the bias or variance specifically. Overfitting is the main unavoidable problem in DL that could influence the model's performance. Nevertheless, it is possible to reduce it by using optimization techniques such as dropout, early stopping, regularization, and data augmentation. It is also possible to reduce bias and variance by changing the model architecture, even though that is hard to identify and implement.

$$Error = bias^2 + variance \tag{3.20}$$

3.2 Remote Sensing Datasets

The RS data are geospatial earth observation data and environmental records. These data are collected by using RS technologies or sensors. The use of ML, particularly DL, to analyze this imagery data is critical for solving the classification problem. In this imagery RS data, classification has been a prominent research problem. As a result, RS imagery data classification is a significant issue in a variety of domains [83], [84]. LCLU is an important domain area because land cover is continuous spatial data.

To design and evaluate DL methods for LCLU classifications, various RS datasets would be collected from their sources. The United States Geological Survey (USGS), European Space Agency (ESA), and Google Earth are the major sources of RS datasets. The DL algorithms mostly need large set of labeled data to train and classify the RS image, which might be available commercially or publicly. Among the publicly available RS datasets listed in Table 2.1, we used the UCM, SIRI-WHU, and RSSCN7 datasets to design and evaluate the DL algorithms. The sample images in each class of these datasets are depicted in Figure 3.9, Figure 3.10, and Figure 3.11 for UCM, SIRI-WHU, and RSSCN7, respectively.

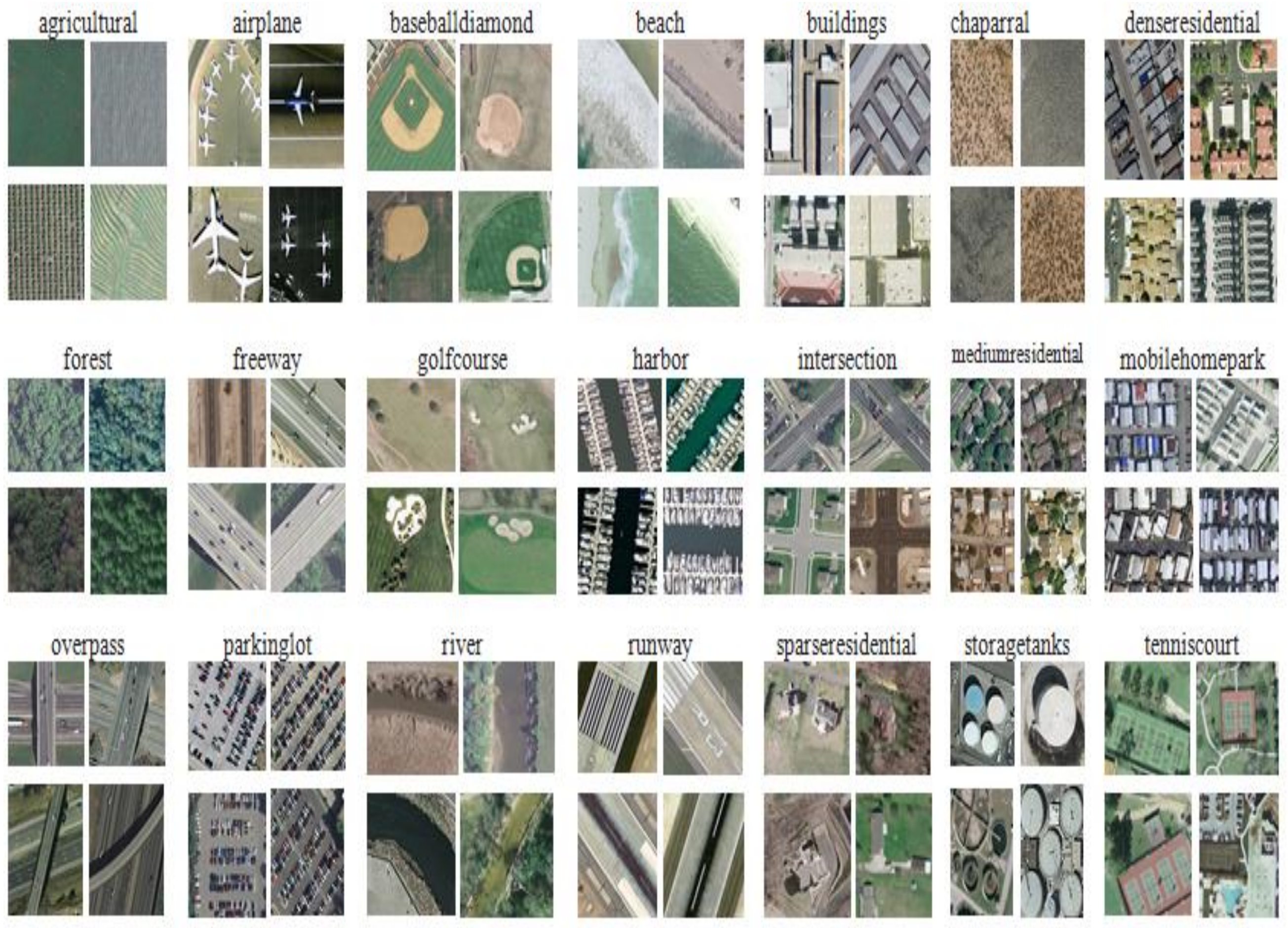


Figure 3.9. Sample images in each class of the UCM dataset

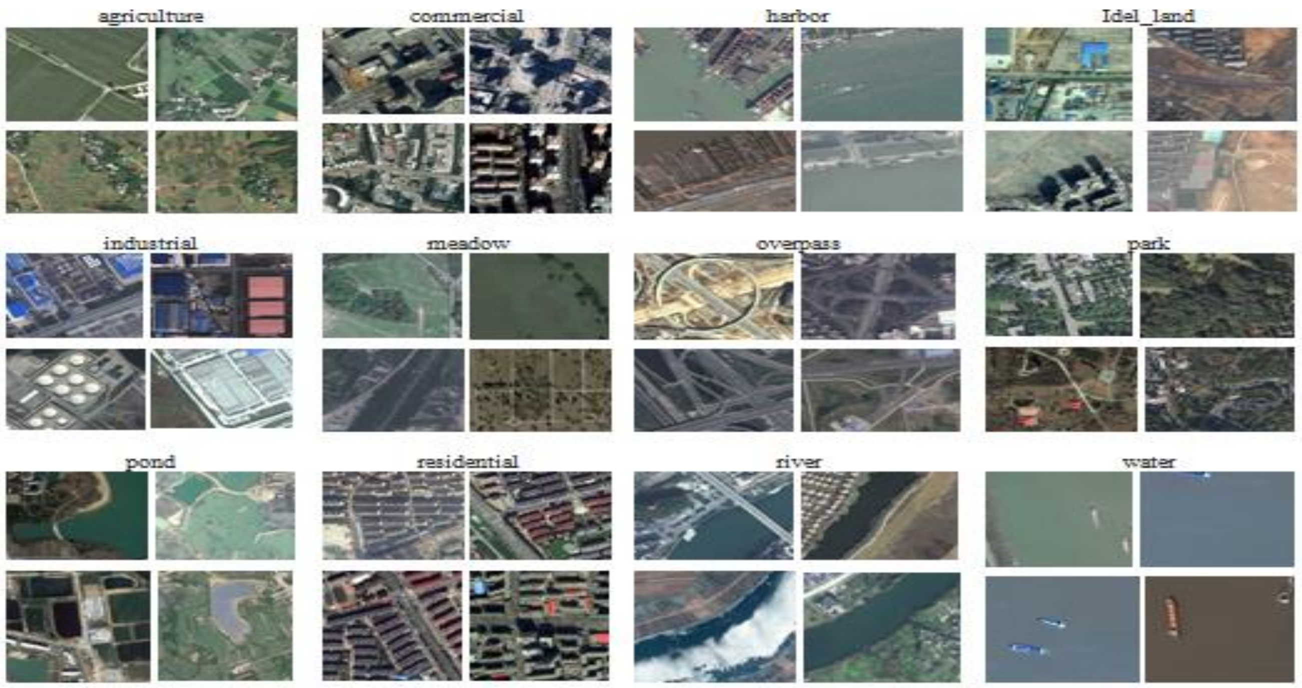


Figure 3.10. Sample images in each class of the SIRI-WHU dataset used for CNN-FE model checking

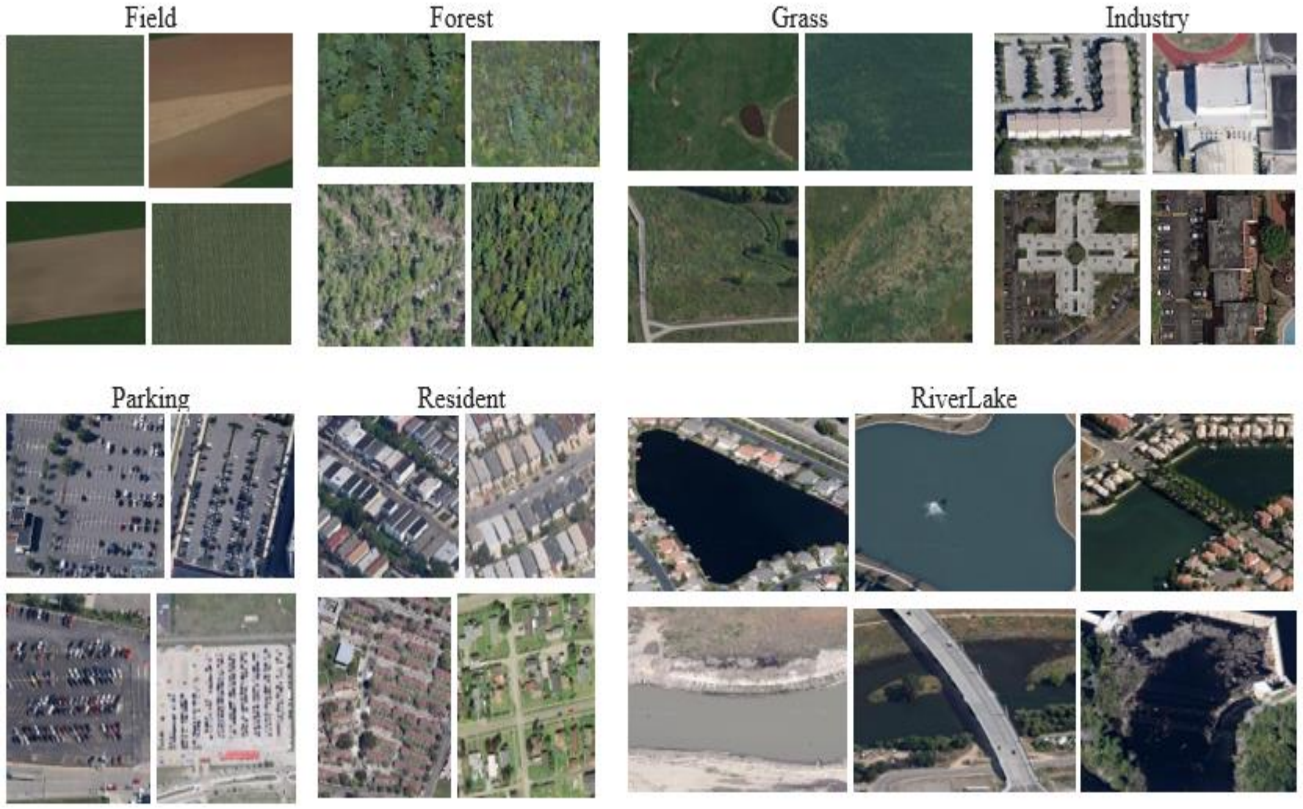


Figure 3.11. Sample images in each class of the RSSCN7 dataset used for evaluating DL methods of the InceptionV3, EfficientNetB7 and MobileNet models performance comparisons

3.3 Tools and Frameworks for Designing DL Models

Tools make ML and/or DL swift and rapid for complex tasks. ML tools provide an interface to the ML developmental programming language; contain platforms that provide capabilities to run a module or project; and contain various libraries that provide all the capabilities to complete a project and provide different algorithms. These ML and DL tools have been used by programmers and researchers for RS image classifications. Python is the most popular interpreted, interactive, dynamically typed, garbage collected, and object-oriented general-purpose scripting high-level language and tool, as listed in Appendix II.

There are various tools and frameworks used in designing DL methods. Such tools include the central processing unit (CPU), Google Colaboratory, TensorFlow, Keras on API (application programming interface), Caffe, Theano, and PyTorch. Google Colaboratory, or simply "Colab," is an online cloud-based Jupyter notebook environment that allows us to train DL and ML models on CPUs, graphics processing unit (GPU), and vision processing unit (VPU). Specifically, we used the Python tool, Keras, and TensorFlow packages with other fitting libraries to design DL models for experimental implementations. Moreover, to visualize and compute the statistical metrics, we use the Panda and Scikit-Learning packages. These packages are listed and described in Appendix II. By deploying the materials of the RS dataset, tools, and relevant packages, we applied the DL methods for LCLU classification. The model design process is sketched in Figure 3.12.

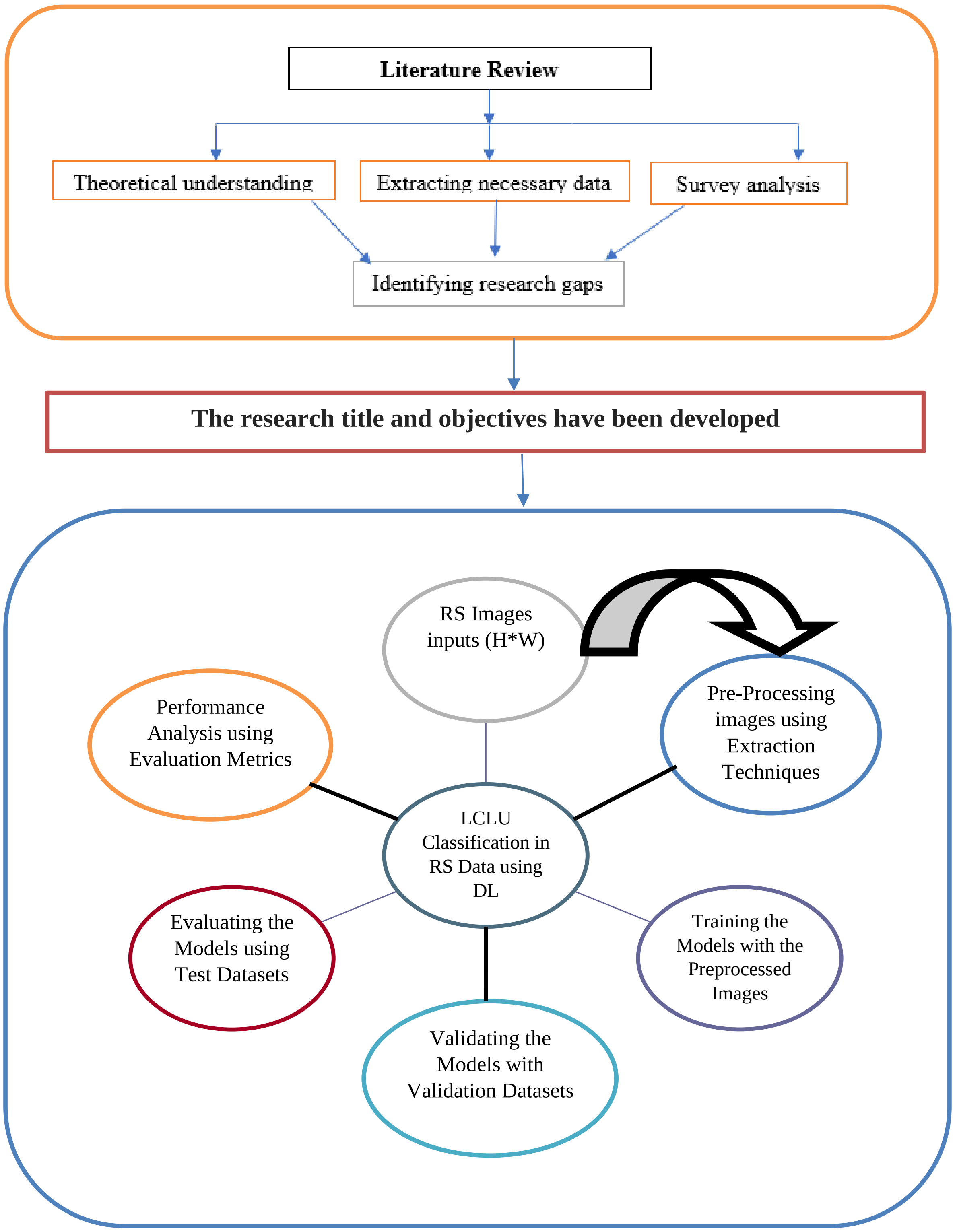

Literature Review
Theoretical understanding
Extracting necessary data
Survey analysis
Identifying research gaps
The research title and objectives have been developed
RS Images inputs (H*W)
Pre-Processing images using Extraction Techniques
Performance Analysis using Evaluation Metrics
LCLU Classification in RS Data using DL
Evaluating the Models using Test Datasets
Training the Models with the Preprocessed Images
Validating the Models with Validation Datasets

Figure 3.12. The overall DL Model for LCLU Classification processes

3.4 Chapter Summarization

In this chapter, we describe the research methodologies that enabled us to accomplish the thesis. These research methodologies include materials and methods. The materials we used in this study include publicly available datasets, computer hardware, software, tools, and DL frameworks. The methods also include the DL approaches, such as CNNs, TL, fine-tuning, and the pretrained networks that are able to classify the LCLU classification in RS images. Moreover, we also described the DL hyperparameters.

We used the UCM, RSSCN7, and SIRI-WHU RS imagery datasets. These datasets have different properties, such as different pixel sizes, resolutions, categories, and locations, as we described in Table 2.1. The CPU, Colab NVIDIA-SMI 460.32.03 Tesla T4 GPU hardware, and Python high-level integrated language were used for implementing the objectives. Analytical tools such as panda (pd), matplotlib (plt), numpy (np), and sckit-learn were used for statistical analysis.

Research methods are the algorithms that are used to design the DL models using materials, tools, and frameworks. The DL methods, such as the CNN, TL, and fine-tuning technique with the base line networks of pretrained neural networks, are applied in this research. The most recently applied pretrained neural networks are listed and described in Table 3.1. To design these models, we used the DL frameworks, such as TensorFlow and Keras, integrated with Python. Dropout, learning rate, and number of epochs or iterations are used to demonstrate the impact of DL hyperparameters on DL performance. Moreover, among the activation functions described in this chapter, Relu and Softmax are used for weight adjustment and learning capability purposes.

4. DESIGNING DL CONVOLUTIONAL NEURAL NETWORK MODEL FOR LCLU CLASSIFICATION USING REMOTE SENSED IMAGES: AN END-TO-END APPROACH

4.1 Introduction

A RS is the art and science of extracting information about an object or phenomenon without making physical contact using advanced sensing technologies. Sensing technologies [95] are remote sensors used to collect large amounts of RS images [96] from the observed earth. RS images are spatial data since they contain spatial information [97]. RS image classification is a hot research challenge in many domains [37], [98], such as environmental monitoring, agricultural and urban planning, and other related domains. Every day, RS technologies generate a large number of RS images. They could be collected from the earth's environment or from space. These images are difficult to analyze since they are varied due to weather, distance, and other determinants. The images could be RGB [97], multispectral [99], [100], or hyperspectral [101]. We aimed to classify LCLU using satellite RS multispectral images.

The LCLU classification problem is the recent focal point of research in RS images [3], [58], [102]–[107]. LCLU in RS images has pixel-level classification and boundary mapping [9]. Thus, RS images are sensitive, according to recent studies [108]. Land is one of the four pillars of sustainable development (social, human, economic, and environmental). Therefore, managing, controlling, and planning the land could be critical for any nation's development. It could better support the tasks in machine-aided LCLU classification systems. The DL approaches, especially CNNs, could be applied to LCLU classification in RS images [109].

The DL approach, CNN is proposed to solve the LCLU classification problem. DL is a robust recent ML approach that enables performance improvement for RS images [26], [102], [103], [110]–[112]. CNNs are prevalent DL techniques that consist of more than two layers [101], and they involve convolution filters [37]. Convolution is the weighted sum of the pixel values of the RS images. The purpose of using convolution is to reduce the size of the input image shape and the total number of parameter in the network [113]. Therefore, the convolutional feature extractor is our image extraction technique for our CNN-FE model.

In recent times, deep CNNs have become pillars and new trends in computer vision [114], and RS image classification is one of the application domains in computer vision [115]. The CNNs could be applied in various domains using RS imagery data, such as LCLU classifications [3], [98], [116]–[119], and object detection [111], [120]–[122]. LCLU classification in labeled RS images has been investigated in the recent era, and we selected this problem to solve with our proposed DL method by applying its hyperparameters.

Nowadays, CNNs methods get more civility in RS image classification problems for their powerful performance improvements [8], [26], [111], [116], [123]–[126], [37], [86], [97], [98], [101], [105], [106], [110]. The CNNs DL approach consists of three main layers: convolutional, pooling, and fully connected [111]. We used various optimization techniques in each of these layers. As a result, CNNs perform various convolution processes from the input to fully trained CNNs. This process makes end-to-end predictions [102]. Deep CNN is an efficient end-to-end approach for outstanding results [5], [37], [104], [107], [127]. The end-to-end algorithms extract the image features from the input to the output processes without using other feature extractor algorithms. Thus, CNN-FE is end-to-end learning.

The deep CNNs models could be built for any classification problem, specifically RS images, in three ways: from scratch development, using pretrained models, or retraining the pretrained models. Pretrained models are modeled earlier on other large datasets, such as "ImageNet" images. From the literature, we observed that most researchers used pretrained models, such as [58], [98], [104], [106], [114], [119], [128], [129], for modeling LCLU classification in RS images.

However, training deep CNNs from scratch has not been widely investigated in RS images [97]. This could be the reason that building CNN models from scratch is difficult due to a lack of ample training data and the large amount of time needed for training [126], [130]. According to our review, very few researchers, such as [3], [35], have attempted to create CNN models in RS image classification. Moreover, despite the prominent results of deep CNNs, there are some problems to be solved regarding to parameter variations. This was our initiation to build the CNN-FE model for LCLU classification in RS images in this study.

In this section, we are motivated to apply the recent DL approaches, especially CNNs, by using various hyperparameters. Therefore, we applied the DL method, convolutional feature extractor

(CNN-FE), with various hyperparameters for LCLU classification using RS images to improve the performance. The recent studies showed that the DL hyperparameters affect the performance of the model [97], [111], [114]. For instance, varying values of the kernel size [117], [127], dropout [3], [131], [132], training data percentages or training data sampling size [111], [116], [126], [128], [133], learning rate [108], [111], [114], [131] could produce different performance results. This demonstrates how changing the hyperparameters affects the performance of the DL model. Therefore, we are also initiating the application of such hyperparameters with their valuable values in this study.

The CNN-FE technique was designed with sixteen layers (three Conv2D, three pooling, three dropouts, three batch normalization, one flatten, and three dense, including the output (softmax) at the top), and evaluating the model with test dataset samples was performed. After training the model, its performance was evaluated and compared with the pretrained network VGG19 in the UCM dataset. The performance improvement has been achieved.

4.2 Methods

4.2.1 DL Method: Convolutional Neural Network

CNN is one of the relevant DL approaches that consists of several sequentially connected layers. This study proposed CNN-based feature extraction (CNN-FE) for the LCLU classification problem using the inconsistent RS images. To get better performance in CNN-FE, we used various DL layers, as shown in Figure 4.1.

In this study, we used the most prominent DL approach for CNNs in the form of Conv2D, which took the image shape (height, width, channel), i.e., (256, 256, 3). In recent studies, CNNs are popular application areas in RS images [127]. As we have described earlier in Chapter 3 of this study, the CNNs consist of the convolution, pooling, and fully connected layers with other DL hyperparameters, including the activation functions. These are vital sequential parameters for the end-to-end DL approach. The sequences of the CNN layers have been depicted in Figure 4.1.

1) The input layer and convolutional (Conv2D) layers

The input layer is the entire input image layer with height*width*channel pixels shapes. It is introduced into the convolutional layer to be processed. Convolutional layers receive the input layers and image pixels and compute the perceptron with a given filter (f, f) or kernel, strides, and padding to the input image volume in a new output volume. The CNNs convolution could operate the mathematical operation of matrix multiplications in given layers. The CNNs are different from other conventional ML approaches in input data types and weight calculations [123] using the convolution method. The feature map of the model is created by the overall process of the convolutional layers.

Using downsampling and upsampling techniques, the model's feature map can be transformed into other resolution feature maps. The downsample is a convolution operation with strides to reduce the input image size and double the number of filters. In contrast, upsampling is a bilinear interpolation operation to double the input image size and reduce the number of filter sizes [95].

The convolutional layers consist of convolution filters or kernels with learnable parameters [118], [127]. Convolution could be performed with valid convolution (no padding), same convolution (with padding), and stride (slide or shift) convolution. The mathematical computation of the output volume of the image in each layer could be calculated using the input volume (height*width), stride(S), and padding (P) parameters. The stride (S) of the filter (f × f) is the intervals of the filter jumps or shifts **S** number of transitions from the first elements in a pixel or each spatial dimension, while padding (P) is the number of pixels added at the outer edges of the input image volumes (height × width). A filter is usually odd and small in size is 3×3, 5×5, and 7×7 with 1, 2, and 3 paddings, respectively. In the Keras DL tool, there is no padding for image border (0) to valid convolution and padding for image border to same convolution. Thus, the output volume (height_{new}* width_{new}) of a layer could be computed using (3.1), and the number of paddings for same convolution could be calculated using (3.2). The default values of P and s are 0 and 1, respectively.

In this study, we used same convolution with the filter size (3,3) and three paddings. The Conv2D layers are used to extract the input image features by sliding a convolution filter size of (3, 3) to produce a new output hierarchical feature map. There are three convolutional block layers in our sequential model training, including 64, 128, and 256 convolution kernels with a filter size of three each. Therefore, convolution is used as our feature extraction method for RS images.

The total parameter numbers of the model are the summations of the calculated results from the Conv2D and dense layers. We design the model with four Conv2D layers that calculate the number of parameters for those layers in the same norm by (3.5) and two dense layers by (3.6). However, the calculation formula for dense parameters differs from Conv2D, as equated in (3.6). The number 1 means the bias associated with each filter for learning.

According to (3.5) and (3.6), we found the total parameter number to be 800,981. However, the number of parameters for all MaxPooling2D and Flatten layers is zero because these layers do not learn anything from the built model.

2) **Pooling Layer**

The pooling layer is used to resize and downsample the spatial representations. We used the common pooling technique called max pooling. It was used for both avoiding overfitting and reducing the number of parameters.

The pooling layers in CNNs are essential for the downsampling processes used to reduce the size of the input RS images. In addition, the block layers involve various max-pooling with 2, the stride with 2, and the padding with "same."

3) **Fully-connected Layers (FCNs)**

FCNs are feature classifiers in the last couple of layers of the network. They include flatten layers, dense layers, and an output layer at the end. Perceptrons in an FCN are fully connected to all previous layer activations.

CNNs also have various activation functions, which should be non-linear as linear functions have a constant derivative, as described in earlier sections. These are softmax, Relu, tanh, and sigmoid or logistic functions. We used the Relu at the entire convolutional layer to activate the weights in each convolution process and the softmax at the output layer since it is common for our multi-class classification. The softmax function is a feature classifier and introduces a probability score for each class. The class with the highest probability score is predicted to be our predicted class. This probability score will be used for performance evaluations later.

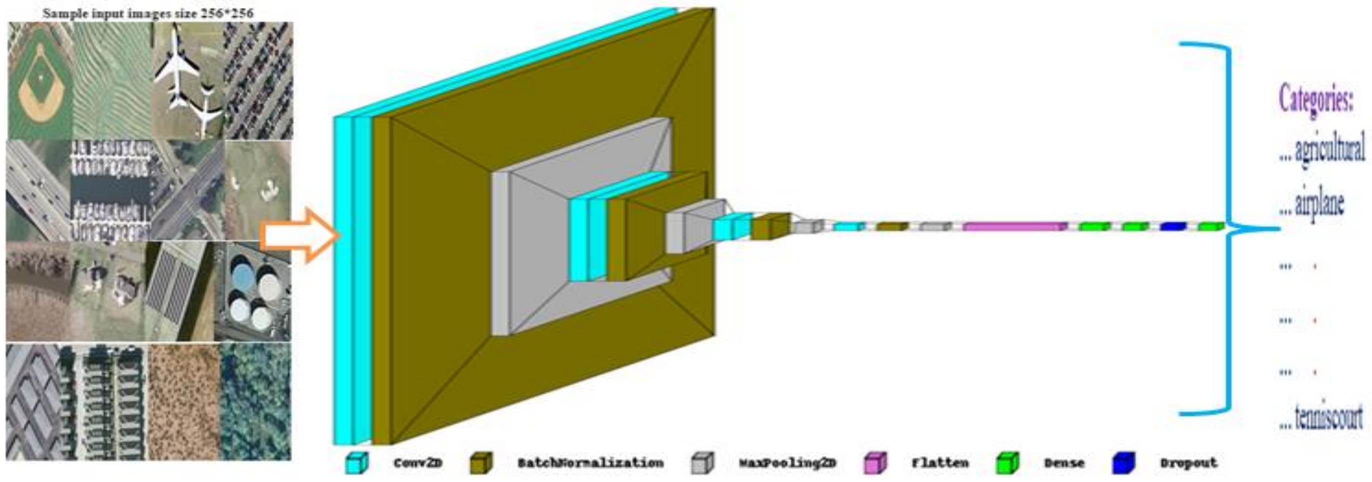


Figure 4.1. Structure of the CNN DL approach

4.2.2 Dataset Descriptions

The RS dataset was collected initially through advanced sensor technologies, and then it could be labeled manually for research or other commercial purposes. On the base of the channel, there are three types of RS images: RGB (that consists of three channels), multispectral (that consists of more than three and under hundreds of channels), and hyperspectral (that consists of hundreds of channels). Recently, various researchers have investigated these data types. We used the UCM RS dataset, which is multispectral.

To test the built model's applicability to the target UCM, we used the rarely studied SIRI-WHU dataset. For training, validating, and testing samples, we used 60%, 20%, and 20% of each labeled dataset, respectively.

The UCM dataset is an LCLU data set collected from the earth, labeled manually, and introduced by [70]. It has 21 classes, each with 100 images that measure 256 × 256 pixels and have a spatial resolution of about 30 cm per pixel. However, the UCM dataset is inconsistent, as about 44 images have different pixel shapes. The variety of properties of the dataset could affect the performance results. Sample images in each class are depicted in Figure 3.9. This dataset is available at http://weegee.vision.ucmerced.edu/datasets/landuse.html.

The SIRI-WHU dataset was collected from Google Earth and covered urban areas in China; it was introduced by [75]. The dataset contains 12 categories and 200 images per category with 200*200 pixels in a spatial resolution of 200 cm per pixel. Sample images in each category are depicted in Figure 3.10. The dataset is publicly available for research purposes at https://figshare.com/articles/dataset/SIRI_WHU_Dataset/8796980.

4.3 Experimental Results and Discussions

4.3.1 Experimental Setting

The dataset and the DL hyperparameters could be considered for their appropriate settings to build our model. As we described in earlier sections, there are 2100 images in the UCM dataset and 2400 images in the SIRI-WHU dataset. Therefore, to reduce the overfitting of the model, we split both the UCM and SIRI-WHU datasets into three sets: the training set, the validation set, and the test set, which compromise 60%, 20%, and 20% of the dataset, respectively. Then, after splitting, the total sample images in the training set, validation set, and test set become 1260, 420, and 420 for UCM and 1440, 480, and 480 for SIRI-WHU, respectively. Each dataset is loaded into the experiment and preprocessed. First, we built the model on the UCM dataset as follows; then, we rebuilt the model on the SIRI-WHU dataset for its applicability approval in the same manner.

Batch size is one of the hyperparameter that can influence on the model performance. It determines the number of samples processed in each iteration or epoch during training. It also affects the speed and stability of training, such as the larger batch sizes can speed up training but may require more memory, and the smaller batch sizes may provide more stable updates but can slow down training. The training set is a collection of 1260 images that have been used to fit and train our model with a batch size of 64 and hundreds of epochs, as shown in Table 4.1 (right column).

Epoch or iteration is a complete pass of the entire dataset during training and it defines how many times the model iterates over the entire dataset. Too few epochs may result underfitting while too many epochs may lead to overfitting that effect on the model’s performance. Therefore, to avoid such problems, we set the appropriate number of epoch (100), validate with validation dataset and use early stopping technique.

In each epoch, the same training images are fed to the CNN-FE architecture recurrently, and the model could learn and continue to learn from the hidden image features. In general, the model was trained in four CNNs sequential layers on a training set, and its performance was evaluated with the validation set during training and with a test set after training.

The validation set is a collection of 420 images separate from the training set that was used to validate our model's performance during the training. Splitting the dataset into a validation set is critical to reducing the overfitting of the training data and evaluating the model during its development.

On the other hand, the test set is a set of 420 images used to evaluate the performance of our model after completing the training. The test set is the support, as shown in the last column of Table 4.3, Table 4.4, Table 4.5, and Table 4.6. It is used to analyze the performance evaluation metrics, including accuracy, loss, precision, recall, F1-score, and confusion matrix.

In addition to setting the dataset splitting, we have chosen the DL hyperparameters to build, compile, and fit our model on the UCM dataset and evaluate the model's performance, as shown in Table 4.5. To reduce overfitting, dropout and early stopping hyperparameters are used. Early stopping is a technique that could automatically stop the train when either validation loss has stopped decreasing or validation accuracy has stopped increasing. In addition to these techniques, the convolutional techniques were applied to preprocess and extract feature maps by reducing the image shape (256, 256, 3) into other reduced feature maps.

Table 4.1. Hyperparameters settings compared with earlier comparative studies

DL Hyperparameters	Chosen values for each DL hyperparameters in both earlier and our studies		
	CNN [35]	CNN [3]	**CNN-FE (Ours)**
Optimizers	Stochastic gradient descent (SGD)	Adagrad	Adam
Batch size	16	10	64
Learning rate	le-3	-	0.0001
Iteration Epochs	120	300	100
Loss function	categorical_cross_entropy	binary_cross_entropy	categorical_cross_entropy
Activation functions	Relu	Relu, sigmoid	Relu, softmax
Dropout	-	0,0.25, 0.50, 0.75	0.5
Early stopping	-	-	Automatically stopping

4.3.2 Performance Evaluation Metrics and Experimental Results

After building the model, we evaluated its performance using the evaluation measurement metrics of accuracy, precision, recall, F1-score, and confusion or error matrix (CM). In addition to these evaluation metrics, we used the loss function, i.e., the categorical cross-entropy, to evaluate the training and validation errors. The training losses are calculated during each epoch, whereas the validation losses are computed after each training epoch for the errors. At most, when the number of epochs increases, the losses are decreased, and the accuracy is increased.

The model's accuracy was evaluated in two ways, i.e., with and without using the early stopping technique using equation (4.1). The early stopping has stopped at a random iteration epoch out of 100 epochs when either the validation accuracy has been stopped increasing (as depicted in Figure 4.2b, Figure 4.6b, Figure 4.8b,and Figure 4.11b) or the validation loss stopped decreasing (as depicted in Figure 4.3b, Figure 4.6b, Figure 4.9b, and Figure 4.12b) while evaluating the models with test set sample images. Therefore, from the experiments with and without early stopping, we observed that the accuracy results increased using the early stopping technique in each model of CNN-FE and VGG19 trained on both datasets, as shown in Table 4.8. In most circumstances, the higher the number of iterations or epochs at which the early stopping technique has been applied, the better the performance of the model could be achieved when comparing the iteration numbers, as shown in Table 4.8.

In addition to evaluating the overall accuracy of both models, we assessed each class with 20 sample images per class using precision, recall, and F1-score performance measurement merits as stated in Table 4.3, Table 4.4, Table 4.5, and Table 4.6. The performance score for precision, recall, and F1-scoremetrics could be computed using equations (4.2), (4.3) and (4.4), respectively, based on the CM summarization Table 4.2.

Furthermore, the CM metric was also used to identify the predicted classes based on the higher normalized probability values at each class intersection. CM analyzes errors and confusion between the column with the occurrences in a predicted class and the row with the occurrences in an actual class [124]. Because it categorizes errors, CM could also be called an error matrix.

The errors could be type I errors (false negatives-FF) or type II errors (false positives-FT), as shown in Table 4.2. A **type I error** is an outcome where the model incorrectly predicts the positive class when it is the actual negative value. In contrast, a **type II error is** an outcome where the model incorrectly predicts the negative class when it is the actual positive value.

The CM considers the normalized probability values for each class category in rows (True labeled class) and columns (predicted labeled class), as shown in Figure 4.4, Figure 4.7, Figure 4.10, and Figure 4.13. CM measures the performance of the DL model, whether each class is correctly classified or incorrectly classified. Therefore, according to Figure 4.4, Figure 4.7, Figure 4.10, and Figure 4.13, the score in the diagonal intersection showed the correct classified classes with a higher normalized probability. In contrast, the results in other rows-columns wise are predicted in misclassified classes with lower a normalized probability. CM in table form is summarized in Table 4.2.

Table 4.2. CM table format for performance evaluations

		Actual Values		
		True	False	
Predictive Values	True	TT	FT/ Type I Error	True Prediction achieved by Precision
	False	FF/ Type II Error	TF	False Prediction

Accuracy is the measure of predictions that the model classified correctly.

$$\text{Accuracy} = \frac{\#\text{ of correct predictions}}{\text{Tot.}\#\text{of predictions}} = \frac{\text{TT}+\text{TF}}{\text{TT}+\text{TF}+\text{FF}+\text{FT}} \tag{4.1}$$

Precision computes a positive predictive value, i.e., a ratio of the positive classes identified correctly to all the expected positive classes. It determines how many positive identifications were actually correct.

$$\text{Precision} = \frac{\#\text{ Positive Predictions}}{\text{Tot. \#of Positive Predicts}} = \frac{\text{TT}}{\text{TT} + \text{FT}} \tag{4.2}$$

A recall is used to identify all actual correct relevant classes retrieved from the dataset.

$$\text{Recall} = \frac{\#\text{ Correct Actual Positives}}{\text{Tot. \#of Actual Positives}} = \frac{\text{TT}}{\text{TT} + \text{FF}} \tag{4.3}$$

The F1 score is the harmonic mean of precision and recall. Its score becomes 1 when both precision and recall are perfect and becomes 0 when either precision or recall results 0. The F1 score measures the preciseness and robustness of the classification model.

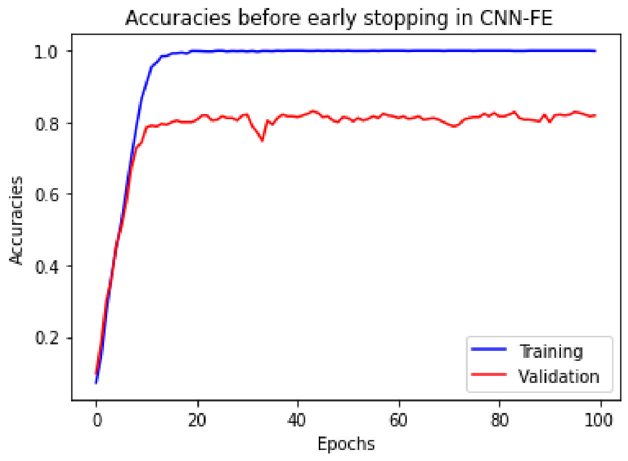
$$\text{F1 Score} = \frac{2(\text{Precision} * \text{Recall})}{\text{Precision} + \text{Recall}} = \frac{2\left(\left(\frac{\text{TT}}{\text{TT+FT}}\right) * \left(\frac{\text{TT}}{\text{TT+FF}}\right)\right)}{\left(\frac{\text{TT}}{\text{TT+FT}}\right) + \left(\frac{\text{TT}}{\text{TT+FF}}\right)} = \frac{2\text{TT}}{2\text{TT} + \text{FT} + \text{FF}} \tag{4.4}$$

After training and modeling our model using various hyperparameters, we retrained it by combining the training and validation datasets with an early stopping technique. Hereafter, the training dataset becomes 80% of the dataset. The training has been stopped at a random iteration out of 100 epochs. This is why the validation loss has stopped decreasing or the validation accuracy has stopped increasing at this epoch. After retraining the model, which was stopped at a random iteration number, we fit the model and evaluated it with 420 test sample images. While assessing the model, precision, recall, f1-score, accuracy, and CM performance measurement metrics were technically used according to Table 4.3.

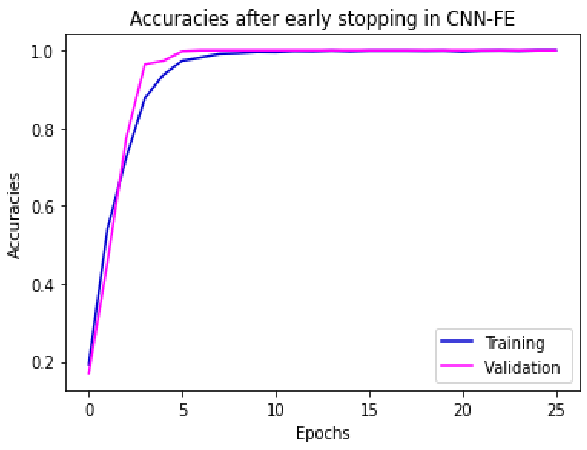
After designing our CNN-FE model, we compared its performances with the VGG19 pretrained feature extractor, which was trained in the same hyperparameters to check the applicability of CNN-FE on RS image classifications. The performances of CNN-FE in various metrics have been provided in Table 4.3, Figure 4.2, Figure 4.3, and Figure 4.4 on the UCM dataset and Table 4.5, Figure 4.8, Figure 4.9, and Figure 4.10 on the SIRI-WHU dataset, respectively. Similarly, the comparable performance of VGG19 has also been provided in Table 4.4, Figure 4.5, Figure 4.6, and Figure 4.7 on the UCM dataset and Table 4.6, Figure 4.11, Figure 4.12, and Figure 4.13 on the SIRI-WHU dataset, respectively.

Table 4.3. Summarizations of the classification performance of CNN-FE for each class with performance measurement metrics in the UCM dataset.

Class name	Precision	Recall	F1-score	Support
Agricultural	0.91	1.00	0.95	20
Airplane	0.94	0.80	0.86	20
Baseballdiamond	0.85	0.85	0.85	20
Beach	0.95	1.00	0.98	20
Buildings	0.79	0.75	0.77	20
Chaparral	1.00	0.95	0.97	20
Denseresidential	0.57	0.80	0.67	20
Forest	0.86	0.95	0.90	20
Freeway	0.95	0.90	0.92	20
Golfcourse	0.80	0.60	0.69	20
Harbor	0.91	1.00	0.95	20
Intersection	0.88	0.75	0.81	20
Mediumresidential	0.58	0.90	0.71	20
Mobilehomepark	0.92	0.55	0.69	20
Overpass	0.81	0.85	0.83	20
Parkinglot	1.00	0.85	0.92	20
River	0.78	0.90	0.84	20
Runway	0.95	0.90	0.92	20
Sparseresidential	0.75	0.90	0.82	20
Storagetanks	1.00	0.90	0.95	20
Tenniscourt	1.00	0.70	0.82	20

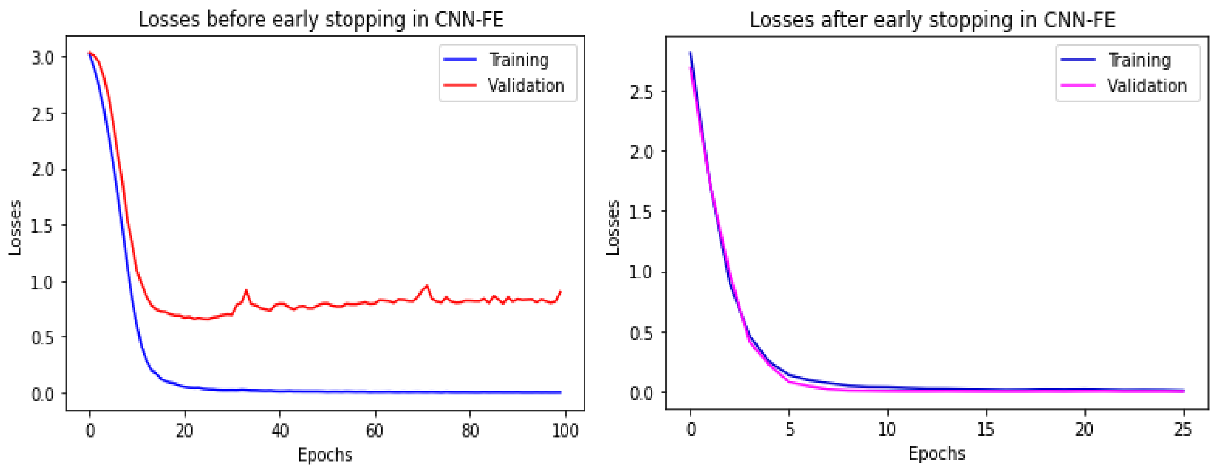


a) Before applying early stopping



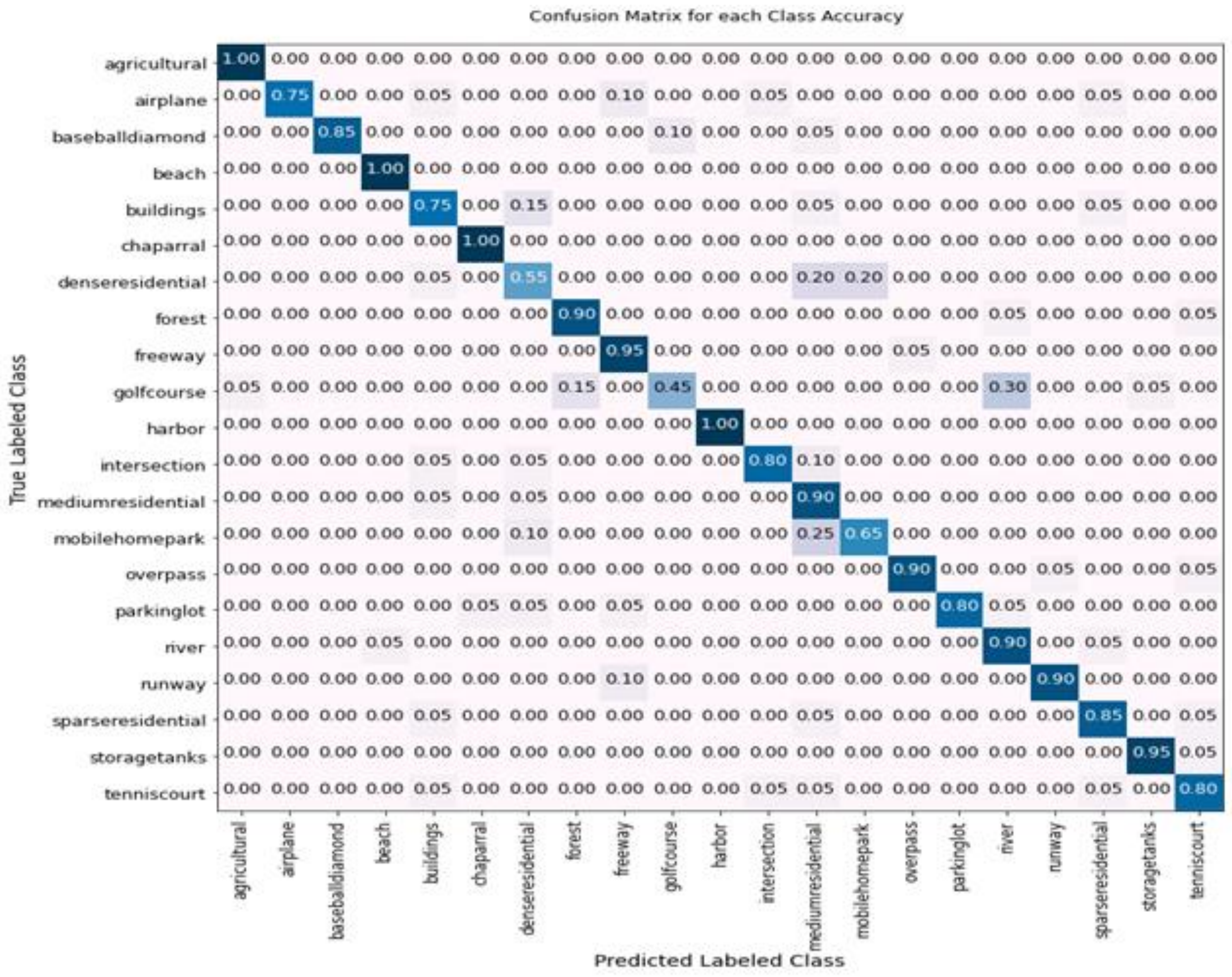
b) After applying early stopping

Figure 4.2. Training and validation accuracies with and without applying early stopping technique



a) Losses before applying early stopping b) Losses after applying early stopping

Figure 4.3. Training accuracy and loss vs. Validation accuracy and loss



	agricultural	airplane	baseballdiamond	beach	buildings	chaparral	denseresidential	forest	freeway	golfcourse	harbor	intersection	mediumresidential	mobilehomepark	overpass	parkinglot	river	runway	sparseresidential	storagetanks	tenniscourt
agricultural	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
airplane	0.00	0.75	0.00	0.00	0.05	0.00	0.00	0.00	0.10	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00
baseballdiamond	0.00	0.00	0.85	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
beach	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
buildings	0.00	0.00	0.00	0.00	0.75	0.00	0.15	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00
chaparral	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
denseresidential	0.00	0.00	0.00	0.00	0.05	0.00	0.55	0.00	0.00	0.00	0.00	0.00	0.20	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00
forest	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.90	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.05
freeway	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.95	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00
golfcourse	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.15	0.00	0.45	0.00	0.00	0.00	0.00	0.00	0.00	0.30	0.00	0.00	0.05	0.00
harbor	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
intersection	0.00	0.00	0.00	0.00	0.05	0.00	0.05	0.00	0.00	0.00	0.00	0.80	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
mediumresidential	0.00	0.00	0.00	0.00	0.05	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.90	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
mobilehomepark	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.00	0.25	0.65	0.00	0.00	0.00	0.00	0.00	0.00	0.00
overpass	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.90	0.00	0.00	0.05	0.00	0.00	0.05
parkinglot	0.00	0.00	0.00	0.00	0.00	0.05	0.05	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.80	0.05	0.00	0.00	0.00	0.00
river	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.90	0.00	0.05	0.00	0.00
runway	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.90	0.00	0.00	0.00
sparseresidential	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.85	0.00	0.05
storagetanks	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.95	0.05
tenniscourt	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.05	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.80

Figure 4.4. CM performance results for each labeled class

4.3.3 Model Validations with VGG19 pretrained Network and SIRI-WHU Dataset

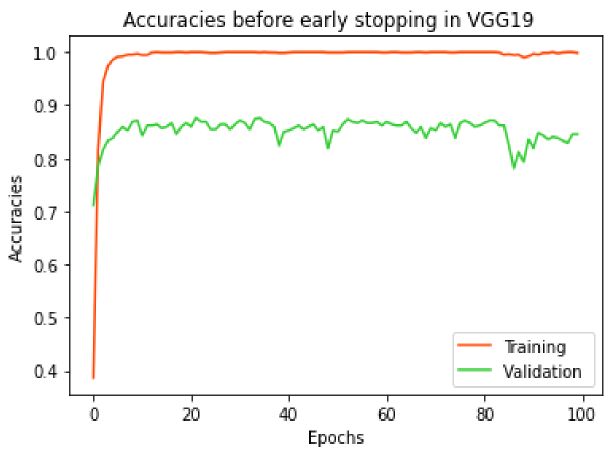
After building and evaluating the CNN-FE model, we assured its possible applicability to the LCLU classification in RS images by comparing its performance with the VGG19 feature extractor network and retraining on another dataset called SIRI-WHU.

The VGG19 pretrained feature extractor was trained on the pretrained network, which was trained on the large dataset “ImageNet” in the same hyperparameters to check the applicability of CNN-FE for LCLU classification in RS images. The VGG19 was designed by [134] to analyze the neural network depth effect on the accuracy of image recognition. Therefore, we created the VGG19 pretrained model to compare its performance with CNN-FE trained on UCM and SIRI-WHU. While comparing the accuracy performances of both DL models, CNN-FE outperformed VGG19, as shown in Table 4.8. Using the early stopping technique improved the accuracy performance of VGG19 in both datasets as well as CNN-FE, as shown in Table 4.8.

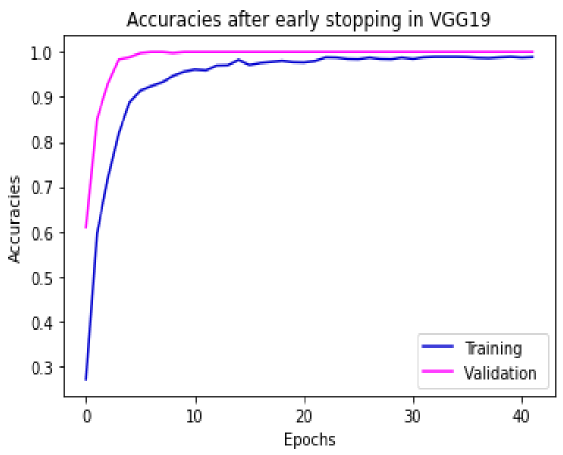
We retrained the CNN-FE model on the SIRI-WHU dataset in addition to testing its applicability on the other DL-pretrained model. As we stated earlier, the properties of the dataset could influence the performance of DL models. To observe this effect, we used the SIRI-WHU dataset with properties different from the target dataset UCM. After training the CNN-FE model on the SIRI-WHU dataset, the validation accuracy and loss fluctuated, especially between epochs 60 and 80 than the validation accuracy and loss trained in UCM, as shown in Figure 4.8a, and Figure 4.9a.

Table 4.4. Summarizations of the classification performance of VGG19 for each class in performance measurement metrics in the UCM dataset

Class name	Precision	Recall	F-score	Support
Agricultural	1.00	1.00	1.00	20
Airplane	0.95	0.90	0.92	20
baseballdiamond	1.00	0.90	0.95	20
Beach	1.00	0.95	0.97	20
Buildings	0.82	0.70	0.76	20
Chaparral	1.00	1.00	1.00	20
Denseresidential	0.50	0.55	0.52	20
Forest	0.82	0.90	0.86	20
Freeway	1.00	0.85	0.92	20
Golfcourse	0.86	0.60	0.71	20
Harbor	1.00	1.00	1.00	20
Intersection	0.81	0.85	0.83	20
mediumresidential	0.69	0.90	0.78	20
mobilehomepark	0.61	0.55	0.58	20
Overpass	0.84	0.80	0.82	20
Parkinglot	0.95	0.95	0.95	20
River	0.76	0.95	0.84	20
Runway	0.87	1.00	0.93	20
Sparseresidential	0.90	0.90	0.90	20
Storagetanks	0.95	0.95	0.95	20
Tenniscourt	0.85	0.85	0.85	20

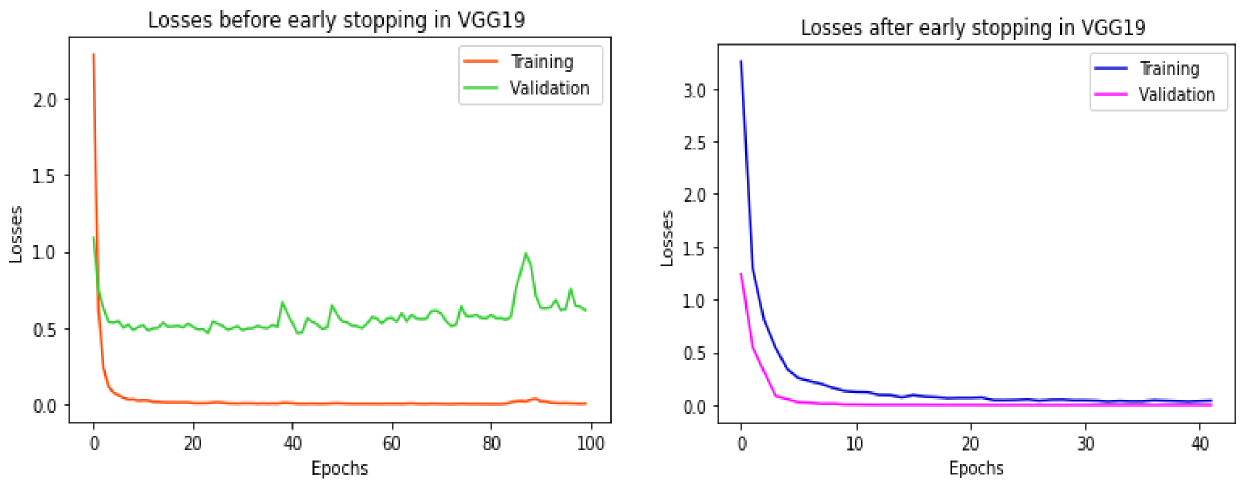


a) Before applying early stopping



b) After applying early stopping

Figure 4.5. Training and validation accuracies in VGG19 with and without applying early in stopping technique in UCM dataset



a) Before applying early stopping b) After applying early stopping

Figure 4.6. Training and validation losses in VGG19 with and without applying the early stopping technique in the UCM dataset

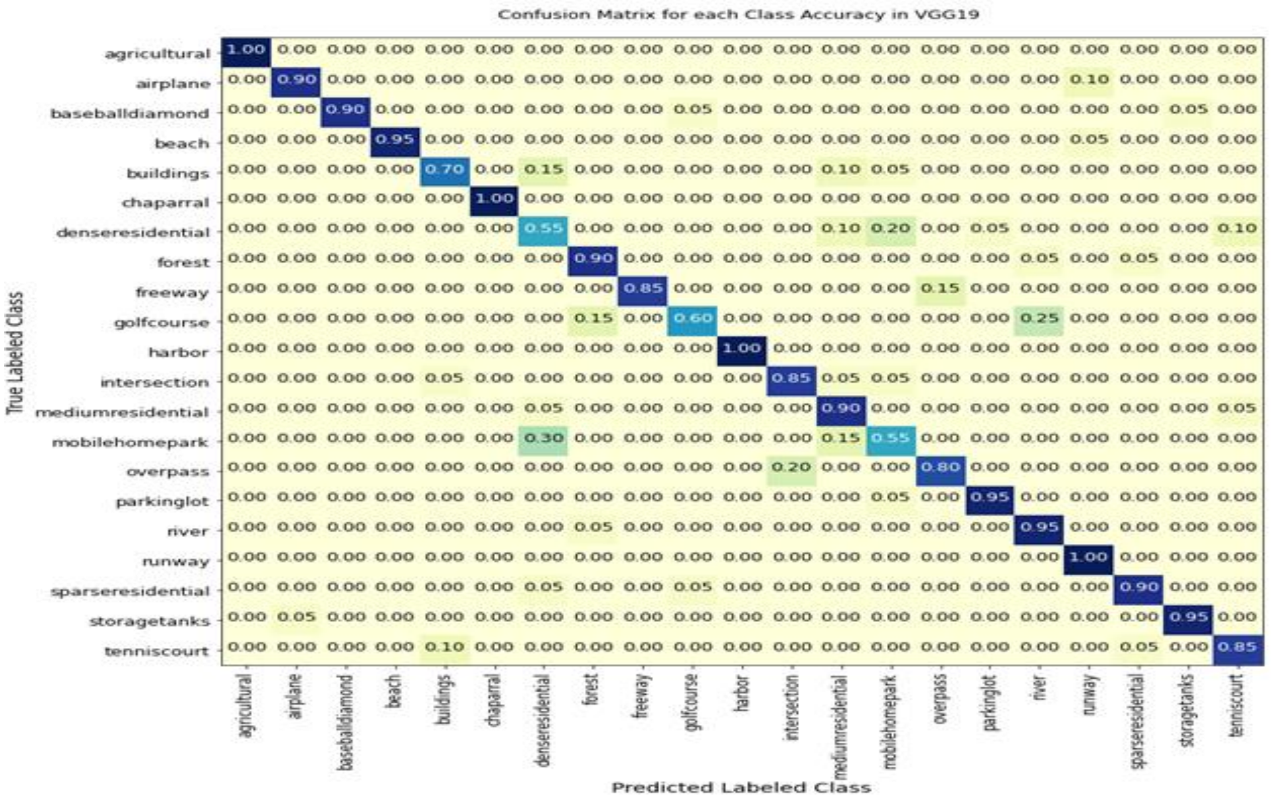
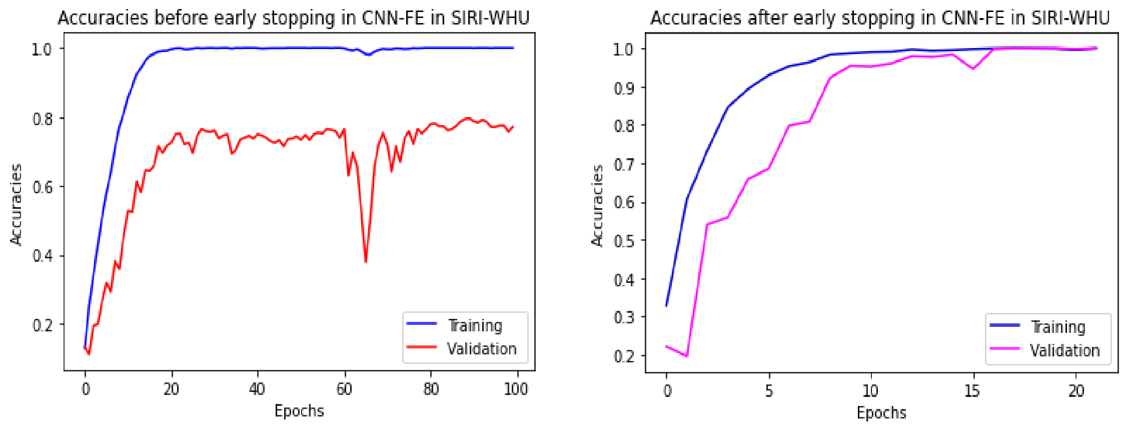


Figure 4.7. CM performance results of VGG19 pretrained for each labeled class

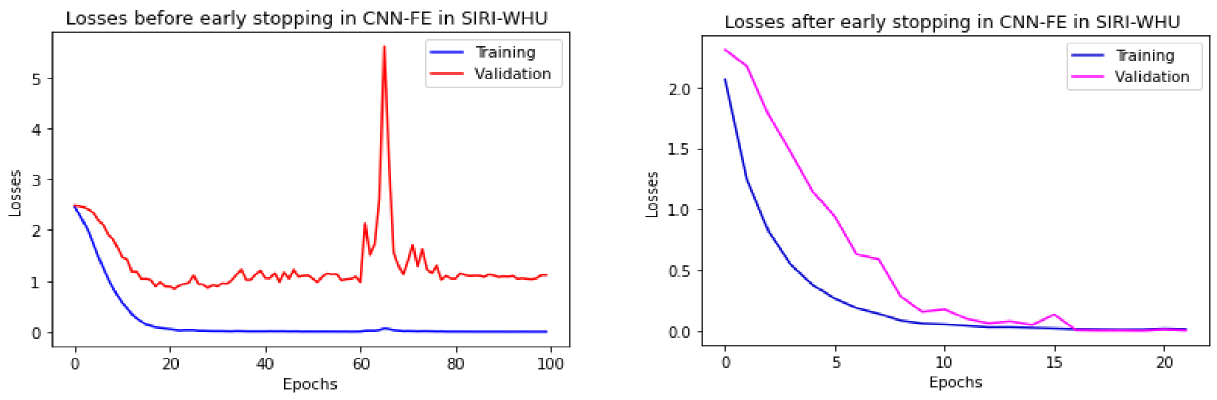
Table 4.5. Summarizations the classification performance of CNN-FE for each individual class with performance measurement metrics in SIRI-WHU dataset

Class name	Precision	Recall	F1-score	Support
Agriculture	0.81	0.65	0.72	40
Commercial	0.94	0.82	0.88	40
Harbor	0.88	0.75	0.81	40
idle_land	0.62	0.78	0.69	40
Industrial	0.90	0.90	0.90	40
Meadow	0.55	0.60	0.57	40
Overpass	0.92	0.90	0.91	40
Park	0.79	0.68	0.73	40
Pond	0.72	0.82	0.77	40
Residential	0.83	0.95	0.88	40
River	0.81	0.75	0.78	40
Water	0.93	1.00	0.96	40



a) Before applying early stopping b) After applying early stopping

Figure 4.8. Training and validation accuracies of CNN-FE model in SIRI-WHU dataset with and without applying early stopping technique



a) Before applying early stopping b) After applying early stopping

Figure 4.9. Training and validation losses of CNN-FE model in SIRI-WHU dataset with and without applying early stopping technique

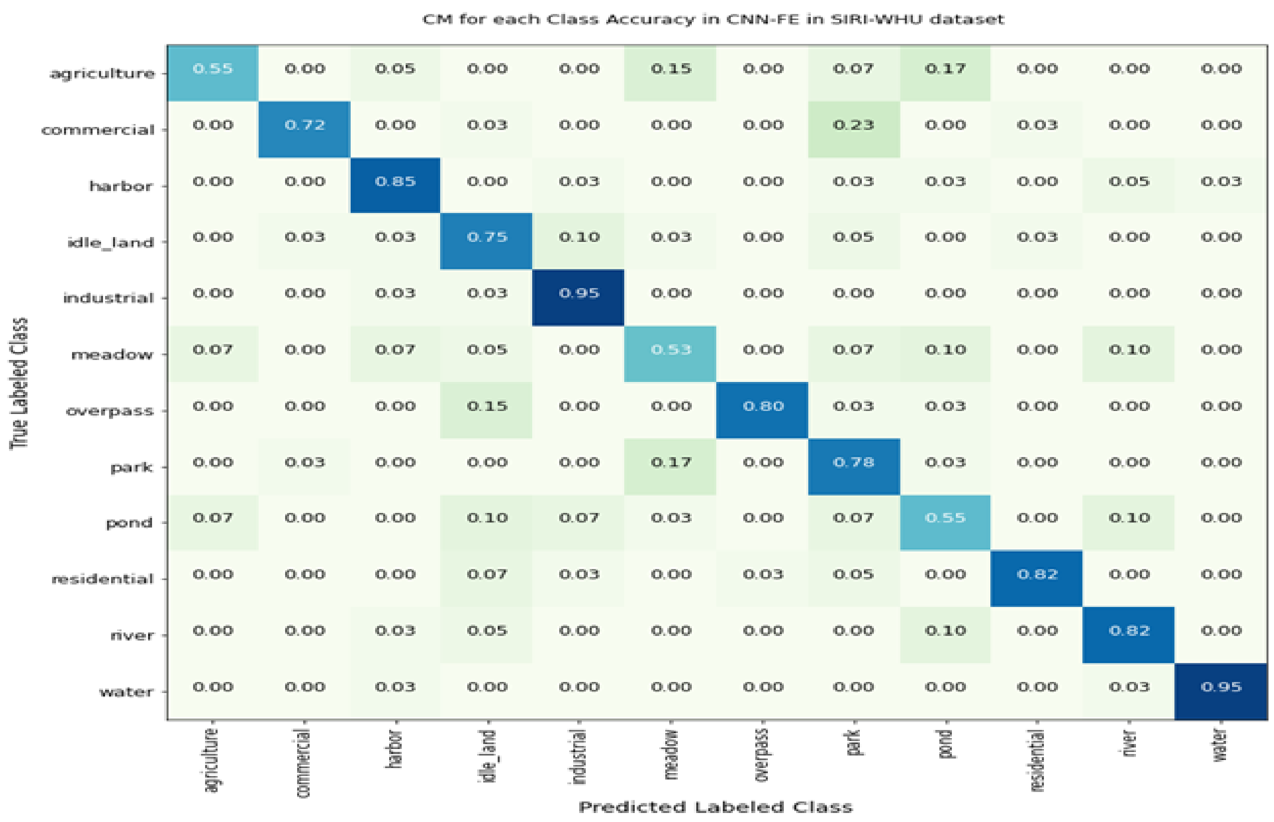
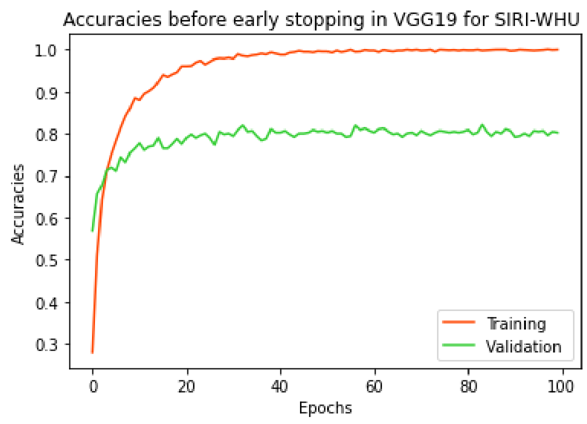


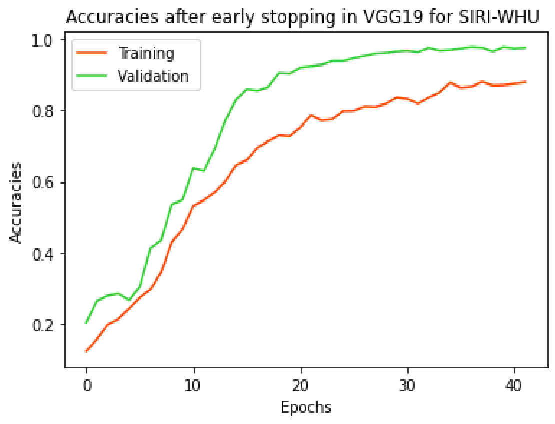
Figure 4.10. CM performance results of CNN-FE for each class classification in SIRI-WHU

Table 4.6. Summarizations of the classification performance of VGG19 for each class with performance measurement metrics in the SIRI-WHU dataset

Class name	Precision	Recall	F1-score	Support
Agriculture	0.91	0.50	0.65	40
Commercial	0.72	0.95	0.82	40
Harbor	0.88	0.93	0.90	40
idle_land	0.86	0.62	0.72	40
Industrial	0.85	0.88	0.86	40
Meadow	0.63	0.60	0.62	40
Overpass	0.80	0.97	0.88	40
Park	0.50	0.60	0.55	40
Pond	0.67	0.75	0.71	40
Residential	0.92	0.88	0.90	40
River	0.85	0.72	0.78	40
Water	0.98	1.00	0.99	40

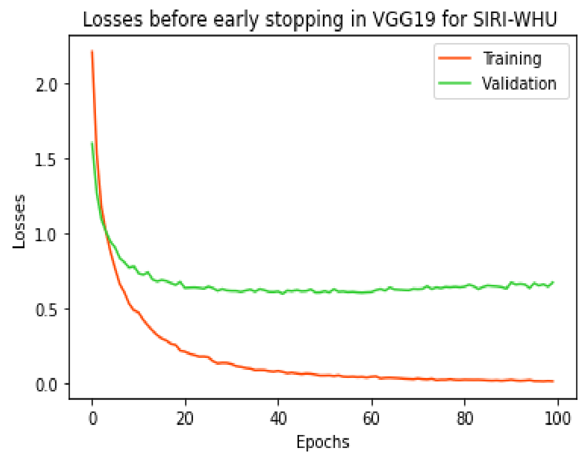


a) Before applying early stopping

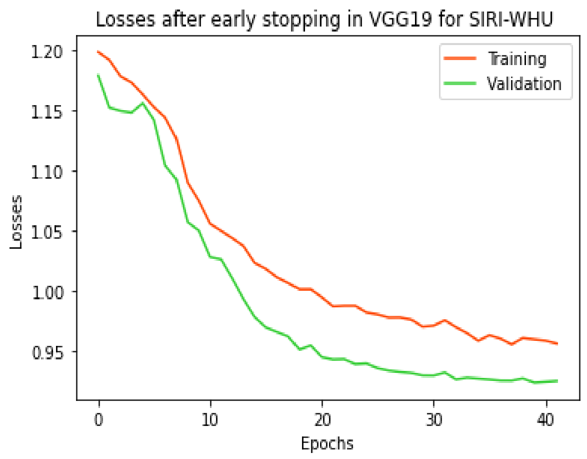


b) After applying early stopping

Figure 4.11. Training and validation accuracies of VGG19 in the SIRI-WHU dataset with and without applying the early stopping technique



a) Before applying early stopping



b) After applying early stopping

Figure 4.12. Training and validation losses of VGG19 in SIRI-WHU dataset with and without applying early stopping technique

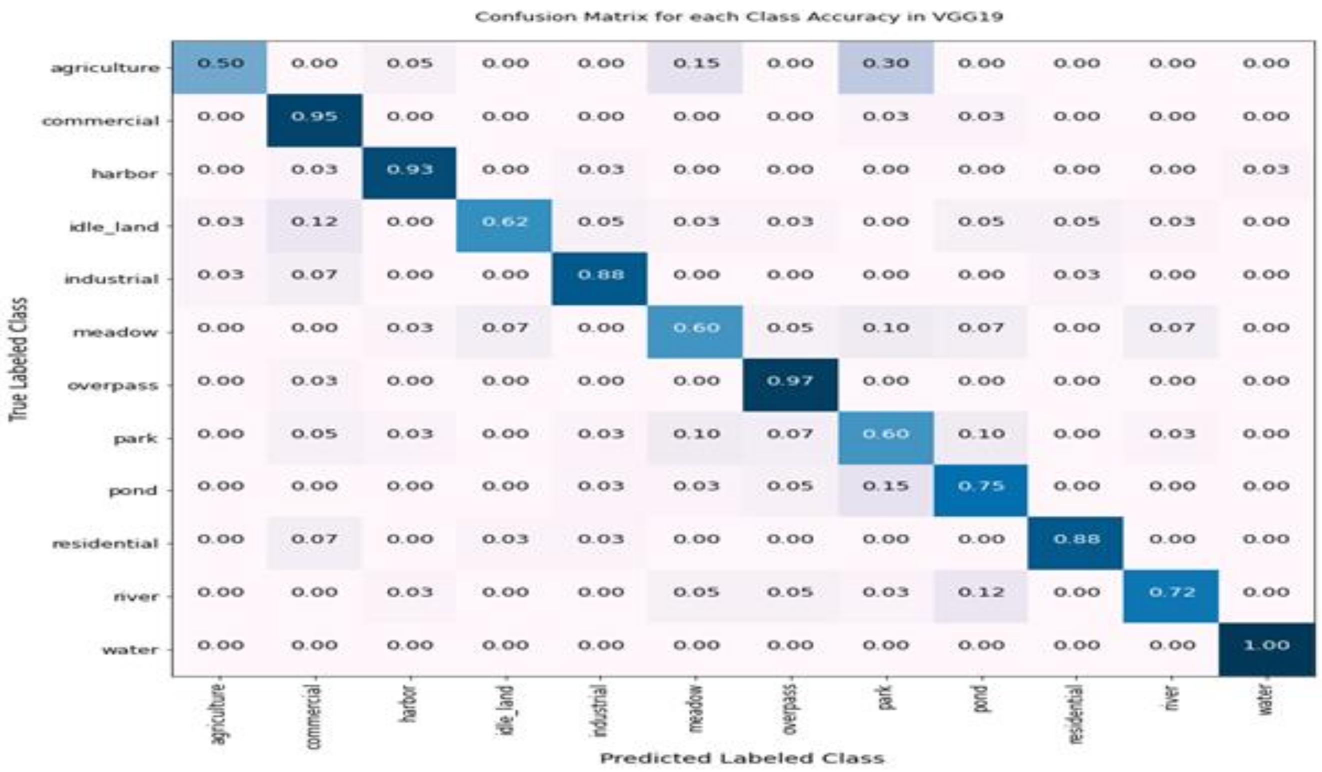


Figure 4.13. CM performance results of VGG19 for each class classification in SIRI-WHU

4.3.4 Discussions

This study investigated the application of an end-to-end DL approach called CNN-FE for LCLU classification using RS images. We showed the possibility of designing a CNN-FE model for LCLU classification in complex RS images using two different datasets. We also developed a comparative VGG19 pretrained network using the same hyperparameters. In addition to validating this DL pretrained model, we retrained the CNN-FE on the SIRI-WHU dataset and assured its applicability in the domain. Therefore, as far as our knowledge, CNN-FE is significant in this study.

4.3.4.1 Discussions on Results

The performance of the CNN-FE model shows that DL models could be built and applied for the LCLU classification domain. It is comparable to those trained from pretrained models in the UCM dataset. When compared to the VGG-19 pretrained architecture, the significant results were reported. In addition, the CNN-FE was retrained on the SIRI-WHU dataset, and a considerable accuracy performance was achieved in the UCM, as shown in Table 4.8. To sum up, the performance of the CNN-FE model resulting from various measurement metrics showed that it is possible to prove its applicability to the classification problem in RS images.

Each class classification performance was evaluated with precision, recall, and an F1-score. Therefore, according to Table 4.7, the classes such as chaparral, parkinglot, storagetanks and tennis-court have the best precision performed, which means that these classes were precisely predicted. However, the lower result precisions were reported for dense-residential (i.e., 0.57), which means that it has inflexible properties to predict precisely. The classes such as agricultural beach and harbor were classified in best recall performance, while mobile-home-park class scored the lower recall performance. Classes with perfect or lower performance in both precision and recall also have perfect or lower results in the F1-score. Thus, there were no classes with perfect or lower performance in both metrics, and there were no perfect classes in the F1-score. However, a lower F1-score was recorded in no in dense-residential (0.67) class. Perfect performance means 100% accurately and precisely classified when measured in given metrics.

To sum up, the individual class performance of the two models in the two datasets is summarized in Table 4.7. The dense-residential class has lower performance in both methods than other classes in the UCM dataset, while the meadow and park classes have lower performance in CNN-FE and VGG19, respectively, in the SIRI-WHU dataset. In the case of CM metrics, better result performance for each class has been observed in both methods in the UCM dataset than in the SIRI-WHU dataset, as compared and shown Figure 4.4, Figure 4.7, Figure 4.10, and Figure 4.13.

The classes, such as agricultural, harbor, overpass, and river, are common in both datasets. However, most of these classes have different performance values, as shown in Table 4.3, Table 4.4, Table 4.5, Table 4.6, and Table 4.7. This could be why the two datasets have inconsistent properties, which were collected from different locations with different resolutions and pixel values.

In addition to evaluating the individual classes, we also evaluated the two methods within the two datasets. Thus, while comparing the CNN-FE from the pretrained VGG19 network, outperformed results in CNN-FE have been achieved in both datasets, as shown in Table 4.8.

4.3.4.2 Discussions on State-of-the-art Studies Comparisons

In this objective, we aimed to improve the performance of the DL model from the existing state-of-the-art studies studied by [3] and [35] by considering their limitations for the DL hyperparameters. The DL hyperparameters influence the DL model's performance. Therefore, to see the effect, we used various hyperparameters, such as dropout, learning rate, batch size, epochs, and early stopping, with their respective values. The study [3], has analyzed the dropout hyperparameter effects on the CNN performance with different values (null, 0.25, **0.50,** and 0.75), which generates the accuracy of 81.2, 81.3, **81.4,** and 79.7 with augmentation and 68.0, 73.7, **75.7**, and 77.7 without data augmentation technique, respectively. Among these provided accuracy and dropout values, we listed and compared the last two accuracy performances with unaugmented data, with corresponding dropout values of 0**.5** and 0.75, respectively, shown in Table 4.9.

The CNN-FE model has achieved 89.76% and 80% accuracy in the UCM and SIRI-WHU datasets, respectively. The VGG19 pretrained model has also achieved 85.95% and 78.33% accuracy, as shown in Table 4.8. Moreover, the CNN-FE model outperformed the state-of-the-art studies and the pretrained network, as shown in Table 4.9.

Table 4.7. Class comparisons in precision, recall, and F1-score (%) on the two models and datasets

Dataset	Method	Precision performance		Recall performance		F1-score performance	
		Perfect (1) classes	Lower (-) classes	Perfect (1) classes	Lower (-) classes	Perfect (1) classes	Lower (-) classes
UCM	CNN-FE	Chaparral, parkinglot, storagetanks, and tenniscour	Denseresidential (0.57)	Agricultural, beach and harbor	Mobilehomepark (0.55) and golfcourse (0.60)	None	Denseresidential (0.67)
	VGG19	Agricultural, baseballdiamond, beach, charparral, freeway, and harbor	Denseresidential (0.50)	Agricultural, charparral, harbor and runway	Denseresidential (0.55)	Agricultural, charparral, and harbor	Denseresidential (0.52)
SIRI-WHU	CNN-FE	None	Meadow (0.55)	Water	Meadow (0.60)	None	Meadow (0.57)
	VGG19	None	Park (0.50)	Water	Agriculture	None	Park (0.55)

Table 4.8. Results of accuracy (%) performances at random early stopping technique

Dataset	Methods	Stopped at epoch # out of 100	Accuracy performance results	
			Before early stopping	After early stopping
UCM	CNN-FE	26	85.95	**89.76**
	VGG19	42	85.00	85.95
SIRI-WHU	CNN-FE	22	78.67	**80.00**
	VGG19	41	76.88	78.33

Table 4.9. Comparisons of the accuracy (%) with the state-of-the-arts in the UCM target dataset

Method	Dropout	Precision	Recall	F1-score	Accuracy
CNNs [3]	0.50	85.4	83.3	84.3	75.7
	0.75	85.5	85.8	85.7	77.7
CNN [35]	-	-	-	-	82.38
CNN-FE (Ours)	0.50	90.00	88.00	88.99	89.76
VGG19 (Ours)	0.50	88.00	86.00	86.99	85.95

4.4 Chapter Summarization

In this chapter, we have applied the CNN-FE model to address the challenge of LCLU classification in RS images. CNNs are powerful DL approaches to analyzing RS images for LCLU classification systems. CNNs are powerful DL approaches to analyzing RS images for LCLU classification systems. Therefore, designing CNNs models for LCLU classification in RS imagery data needs more investigation. Some researchers have expressed concerns about the high cost of training time and the scarcity of large datasets required to create CNN models from scratch. It is vital to create CNNs from scratch for RS images since these images are inconsistent, and modeling them from pretrained networks could affect their practical deployment.

Therefore, we applied an end-to-end CNN-FE DL model to extract the inconsistent UCM RS image features for LCLU classification in RS images. We retrained this model on the other SIRI-WHU dataset to analyze whether the dataset influences the model's performance. We also built a VGG19 pretrained DL model on both datasets and evaluated their performances to validate the CNN-FE possible applicability in the domain. We compare its results to previous state-of-the-art studies and the VGG19 pretrained model, which was trained in the same hyperparameters. The CNN-FE outperformed the accuracy performance of state-of-the-art earlier studies and the VGG19 pretrained model. Therefore, we proved that the developed CNN-FE model is possibly applicable to the domain area and improves performance.

5. DESIGNING TRANSFER LEARNING FOR LCLU CLASSIFICATION USING REMOTE SENSED IMAGES

5.1 Introduction

Land cover is variable and dynamic on the earth’s surface [135], whereas land use is the intervention of human activities on the earth. Land cover is the earth’s surface covered by physical features like a forest, river, vegetation, or others. In contrast, land use is the ability of a human to use natural resources for various purposes [136]. Thus, LCLU describe the earth’s features and human interaction. Classification is needed in land cover mapping [3], [137], [138], and land use resource management [47], [136], [139]. LCLU classification is an important and challenging task [12], and it contributes to agricultural decision-making and urban forecasting in the earth observation environment for sustainable development. This classification problem will be solved using TL models for RS images.

RS images are geospatial earth observation data and environmental records. As they raise exceptional problems to new scientific questions, RS data face "big data" challenges as well as some new DL challenges [22]. RS imagery data classification is a significant problem in various domains [83], [84], [99], [115], [124], [135], [140]–[142]. Thus, our consideration of classification was one of the major research problems in RS imagery data. Nowadays, researchers are exploring the application of DL to confront these challenges.

DL gets more attention for the LCLU classification problem in RS images [143]. The DL approaches could extract the earth’s features from remotely sensed imagery data to manage the earth’s environment by properly utilizing deep classification systems. DL algorithms are calling focuses on their automatically learning ability from large datasets [22], [56], [144]–[147], [80], [82]–[86], [140], [141].

In recent studies, the DL methods, especially CNNs, have been widely used in RS image classification for their outstanding performance and accuracy [5], [56], [124], [143], [146]. However, DL algorithms could take more time and complexity, creating overfitting [11], [56], [139], [148], [149] when training the DL models from scratch. TL, the innovative DL model in ML, could overtake this problem because TL is an optimization technique used to reduce processing time or improve performance or accuracy [150].

Thus, TL could apply formerly learned techniques to resolve new problems efficiently [151]. Now a day, TL has gotten increasing attention lately for reducing training time for large datasets [8], [56], [57], [60], [152], [153].

The TL models could be applied in various RS domains. For instance, it has been applied for forest variable estimation [154], for object (airplane) detection [151], [155], for poverty mapping [85], for labeling the Synthetic Aperture Radar (SAR) [148], for change analysis [156], and for marsh vegetation classification [157].

In the domain area, related work has been attempted to investigate the CNN-based TL model. Few researchers [5], [11], [24], [58], [59], [149], [153] have investigated CNN-based models using pretrained architectures for RS image classification. Using TL, the LCLU classification problem was investigated by [142], [158] using TL. However, TL in RS has not been widely explored yet (Astola et al., 2021), especially in the LCLU classification. Thus, we applied deep neural network-based TL [80] in LCLU classification using RS images.

Our motivation was to apply the deep TL model with pretrained models for the LCLU classification in RS images and improve the performance efficiently. We have listed the related studies with their recommendations in our previous work [159]. Therefore, we were also motivated to investigate the recommended pretrained networks suggested by [3], [5]. Our objective in this study was to apply the deep TL models and improve their performance efficiently for LCLU classification in RS images. To achieve this objective, we followed the following procedures: preprocessed the UCM imagery data, extracted the image features using the bottleneck feature extraction technique, modeled the TL with four sequential layers (flatten, dense, two activations (Relu and softmax), and dropout layers), and evaluated using a confusion matrix.

5.2 Research Method: Deep Transfer Learning

In this chapter, we proposed the Deep TL method, which is a deep CNN technique, for efficient time consumption. Building the model for better performance uses various parameters, such as pretrained models, learning rate, early stopping, dropout, optimizer, loss, and activation functions.

Pretrained models have recently been used in RS image classification problems [3], [5], [142], [146], [160]. The pretrained CNN based [150] TL models used in this study included ResNet50V2

[115], [161], VGG19 [134], [162], [163], and InceptionV3 [90], [164]. These pretrained architectures are the deep CNN pretrained models used to design a new TL model from the existing problem.

Learning rate (LR) was used to facilitate the TL model's learning from the UCM dataset. It has various values such as 0.01, 0.001, and 0.0001. However, if the larger LR is used, training and learning may fluctuate [158]. Therefore, the smaller LR value is advisable to be used in building DL models. So, we used the LR of 0.0001 in this objective to optimize our model.

Reducing overfitting in the DL method is dynamic. Dropout and early stopping are the major optimization techniques used for reducing overfitting when training data. The percentage values for dropout expressed in decimal forms are usually recommended to use 0.2, 0.3, 0.4, and 0.5. We used 0.5 (i.e.50%) to reduce the training overfitting since higher dropout could perform better than lower values [3]. Early stopping is a deep CNN regularization technique used to stop the training after random epochs when the model performance could not improve [158].

In DL modeling techniques, classification loss functions are widely used. This classification loss could be binary cross-entropy or multi-class cross-entropy. We preferred the multi-class entropy loss function since our class is multi-classes of the RS images.

Activation functions could be used afterward for each convolutional layer to raise the capability of neural network [140]. In this study, the activation functions Relu [165] and softmax [58] were used because they are better than other common nonlinear functions like tanh and sigmoid functions. Relu and softmax are better at easily propagating errors; multiple layers of the neurons have been activated, and their mathematical operation is simpler than that of tanh and sigmoid functions.

Relu produces x if $x>0$ or 0 if $x<0$ as observed in (equation (3.13) and Figure 3.7). This output implies that neurons with negative values are not activated, while neurons with positive values are. The slope of the derivative (gradient) value of Relu is constant, i.e., either 1 $\forall x, x >= 0$ or 0 $\forall x, x<0$ (equation (3.14) and Figure 3.8).

Softmax (softargmax) is used to predict the class having the highest probability in multi-class classification problems for the input labels. We also used this function since our RS imagery data is a multi-class classification problem. The forward weights of the softmax function could be calculated using equation (12), and the backward or derivative function could be calculated using the simplified equation (3.19). The softmax output is between 0 and 1, and the sum of each class probability is **1.0**. If some **N** elements of the input vector are **N**<0 or **N**>1, they would be between (0, 1) after using the softmax function.

In summary of the method, the hyperparameters such as networks and weights were trained in the pretrained InceptionV3, Resnet50V2, and VGG19 models. We used the bottleneck feature extraction method to extract image features from these pretrained models. Bottleneck is a layer with fewer neurons than the other layers in CNN. The bottleneck layers are used to reduce the number of feature maps (channels) in a given network and to reduce the error (cost) function by updating all the weights of the pretrained neural networks.

A fully connected network for pretrained models was removed, and then a new model was built, and its weights were also removed. The bottleneck features, which become the inputs for FC, are trained for UCM images, as shown in Figure *5.1*. For each pre-trained model, the bottleneck feature extracted the features of shape (1264, 6, 6, 2048) in training bottleneck prediction and the shape of the features (420, 6, 6, 2048) in validation and testing bottleneck predictions.

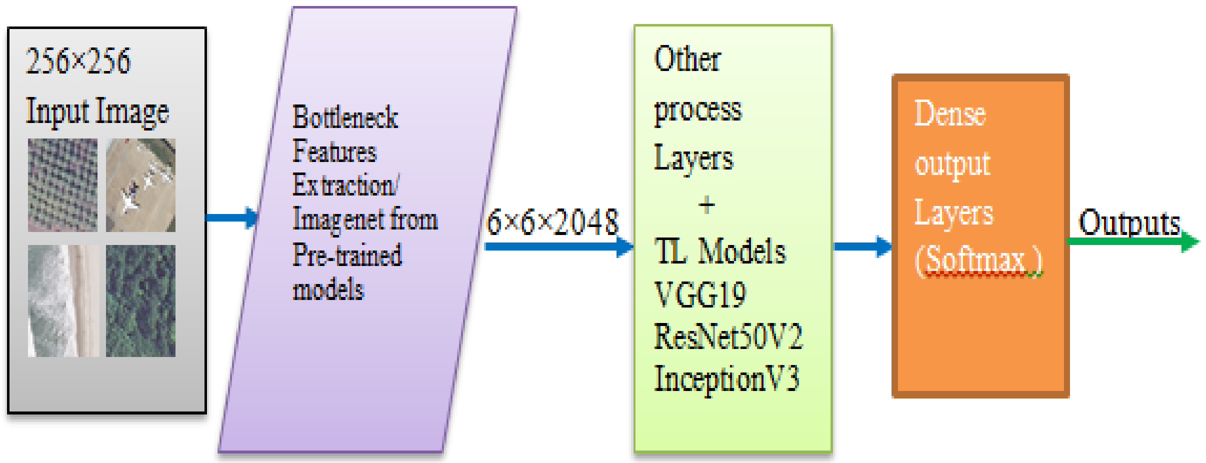

256×256
Input Image
Bottleneck Features Extraction/ Imagenet from Pre-trained models
6×6×2048
Other process Layers + TL Models VGG19 ResNet50V2 InceptionV3
Dense output Layers (Softmax)
Outputs

Figure 5.1. Sample Input Images Feeding into Pre-processing

5.3 Experiments and Performance Evaluations

5.3.1 Experimental datasets setting

The University of California Merced (UCM) data set is used to solve the problem of LCLU classification. The UCM Land Use data set was manually collected and introduced by [70] from the USGS National Map Urban Area Imagery. This dataset is made up of 21 land use and land cover classes, each with 100 images that measure 256 256 pixels and have a spatial resolution of about 30 cm per pixel. The dataset was divided into a 60:20:20 ratio for training samples, validation samples, and tasting samples for each class, respectively, as shown in Table 5.1.

Table 5.1. Parameter settings for UCM dataset

Classes	Training Samples	Validation Samples	Test Samples	Total
Agricultural	60	20	20	100
Airplane	60	20	20	100
Baseball diamond	60	20	20	100
Beach	60	20	20	100
Buildings	60	20	20	100
Chaparral	60	20	20	100
Dense residential	60	20	20	100
Forest	60	20	20	100
Freeway	60	20	20	100
Golf course	60	20	20	100
Harbor	60	20	20	100
Intersection	60	20	20	100
Medium residential	60	20	20	100
Mobile home park	60	20	20	100
Overpass	60	20	20	100
Parking lot	60	20	20	100
River	60	20	20	100
Runway	60	20	20	100
Sparse residential	60	20	20	100
Storage tanks	60	20	20	100
Tennis court	60	20	20	100
Total	1260	420	420	21000

5.3.2 Experimental Settings and Performance Results

As we discussed earlier, various hyperparameters do have important implications for classification problems. So, we have used some of the important parameters in our experiment listed in Table 5.2. In addition to using the dropout (0.5) layer, we used the early stopping technique to reduce the overfitting.

Table 5.2. Hyperparameters Setting for Training Data

Hyperparameters	Parameter values used
Optimizer	Adam
Activation functions	Relu and Softmax
Loss function	categorical cross entropy
Batch-size	64
Epochs	100
Learning rate	0.0001

We combined the training and validation data after validating the model during the process and then evaluating the model's performance with 20% of the testing data. CM measures the performance of the TL model, whether it is classified correctly or incorrectly. We used the classification metrics to calculate the model's performance: accuracy, precision, recall, and F1 measures using equations (4.1) through (4.4), respectively.

There are N (N = 21) classes with an integer labeled 0 to N-1. The generated records were transformed into a confusion matrix that generates the number of correctly classified classes out of 20 test sample images, as depicted in Figure 5.2. The three TL models generated the class label records for 21 classes ranging from 0 to 20 while testing each class with 20 samples. For instance, in the Inception_v3 model in Figure 3a, the first class is labeled with 0, and among 20 testing samples, 18 classes are correctly classified, but the other two classes, i.e., the actual class 3 and 18, are predicted as class 1.

Based on the confusion matrix depicted in Figure 5.2a, the performance of TL with the Inception_v3 model has been calculated and recorded in Table 5.3. Similarly, the performance of TL with the Resnet50v2 and VGG19 models has been measured in Table 5.4 and Table 5.5 based on the confusion matrix (Figure 5.2b and Figure 5.2c), respectively.

```
[[18  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0]
 [ 0 20  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 18  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 19  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0]
 [ 0  0  0  0 18  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 20  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 10  0  0  0  1  0  4  4  0  0  0  0  0  0  1]
 [ 0  0  0  0  0  0  0 19  0  0  0  0  0  0  0  0  0  0  1  0  0]
 [ 0  0  0  0  0  0  0  0 19  0  0  1  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  3  0  9  0  0  0  0  0  0  8  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 20  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  1  0  1  0  0  0  0 18  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  1  0  0  0  0  0 16  0  0  0  0  0  1  0  2]
 [ 0  0  0  0  2  0  0  0  0  0  0  0  4 14  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 19  0  0  1  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 20  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0 19  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 20  0  0  0]
 [ 0  0  1  0  0  0  0  0  0  1  0  0  1  0  0  0  0  0 17  0  0]
 [ 0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0 19  0]
 [ 0  0  1  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0 15]]
```

a) Inception_v3 model

```
[[20  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 20  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 19  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0]
 [ 0  0  0 19  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 15  0  4  0  0  0  0  0  1  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 20  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 14  0  0  0  0  0  3  3  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 18  0  0  0  0  0  0  0  0  1  0  1  0  0]
 [ 0  0  0  0  0  0  0  0 20  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  3  0 11  0  0  0  0  0  0  5  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 20  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  1  0  1  0  0  0  0 16  1  1  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  1  0  0  0  0  0 17  0  0  0  0  0  2  0  0]
 [ 0  0  0  0  1  0  5  0  0  0  0  0  2 12  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  3  0  0  1  0  0 16  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  1  0 19  0  0  0  0  0]
 [ 0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0 17  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 20  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  2  0  0  1  0  0  0  0  0 17  0  0]
 [ 0  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0  0 18  0]
 [ 0  0  0  0  2  0  0  0  0  0  0  2  0  0  0  0  0  0  1  0 15]]
```

b) Resnet50v2 model

```
[[20  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 18  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0]
 [ 0  0 17  0  0  0  0  0  0  2  0  0  0  0  0  0  1  0  0  0  0]
 [ 0  0  0 19  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0]
 [ 0  0  0  0 17  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 20  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 17  0  0  0  0  0  1  2  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 18  0  0  0  0  0  0  0  0  2  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 18  0  0  0  0  0  2  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  3  0 10  0  0  0  0  0  0  7  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 20  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  1  0  1  0  0  0  0 18  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  2  0  0  0  0  1 17  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  9  0  0  0  0  0  2  9  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  2  0  0 17  0  0  1  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 20  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0 19  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 20  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0 19  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 20  0]
 [ 0  0  0  0  0  0  1  0  0  0  0  1  0  0  1  0  0  0  0  0 17]]
```

c) VGG19 model

Figure 5.2. Confusion Matrix of each Model on UC Merced dataset

Table 5.3. Inception_v3 model for class performances in Precision, Recall and F1-Score

Class Name	Class Label	Precision	Recall	F1-score	Support
Agricultural	0	1.00	0.90	0.95	20
Airplane	1	1.00	1.00	1.00	20
Baseball diamond	2	0.86	0.90	0.88	20
Beach	3	0.95	0.95	0.95	20
Buildings	4	0.72	0.90	0.80	20
Chaparral	5	1.00	1.00	1.00	20
Dense residential	6	0.71	0.50	0.59	20
Forest	7	0.86	0.95	0.90	20
Freeway	8	1.00	0.95	0.97	20
Golf course	9	0.75	0.45	0.56	20
Harbor	10	0.95	1.00	0.98	20
Intersection	11	0.95	0.90	0.92	20
Medium residential	12	0.64	0.80	0.71	20
Mobile home park	13	0.78	0.70	0.74	20
Overpass	14	1.00	0.95	0.97	20
Parking lot	15	1.00	1.00	1.00	20
River	16	0.70	0.95	0.81	20
Runway	17	0.87	1.00	0.93	20
Sparse residential	18	0.85	0.85	0.85	20
Storage tanks	19	1.00	0.95	0.97	20
Tennis court	20	0.83	0.75	0.79	20
Average Accuracy in each Measures		**0.88**	**0.87**	**0.87**	**420**

Table 5.4. Resnet50v2 model for class performances in Precision, Recall, and F1-Score

Class Name	Class Label	Precision	Recall	F1-score	Support
Agricultural	0	1.00	1.00	1.00	20
Airplane	1	1.00	1.00	1.00	20
Baseball diamond	2	1.00	0.95	0.97	20
Beach	3	0.83	0.95	0.88	20
Buildings	4	0.75	0.75	0.75	20
Chaparral	5	1.00	1.00	1.00	20
Dense residential	6	0.56	0.70	0.62	20
Forest	7	0.86	0.90	0.88	20
Freeway	8	0.83	1.00	0.91	20
Golf course	9	0.85	0.55	0.67	20
Harbor	10	1.00	1.00	1.00	20
Intersection	11	0.84	0.80	0.82	20
Medium residential	12	0.65	0.85	0.74	20
Mobile home park	13	0.71	0.60	0.65	20
Overpass	14	1.00	0.80	0.89	20
Parking lot	15	1.00	0.95	0.97	20
River	16	0.71	0.85	0.77	20
Runway	17	1.00	1.00	1.00	20
Sparse residential	18	0.81	0.85	0.83	20
Storage tanks	19	1.00	0.90	0.95	20
Tennis court	20	1.00	0.75	0.86	20
Average Accuracy in each Measures		**0.88**	**0.86**	**0.86**	**420**

Table 5.5. VGG19 model for class performances in Precision, Recall, and F1-Score

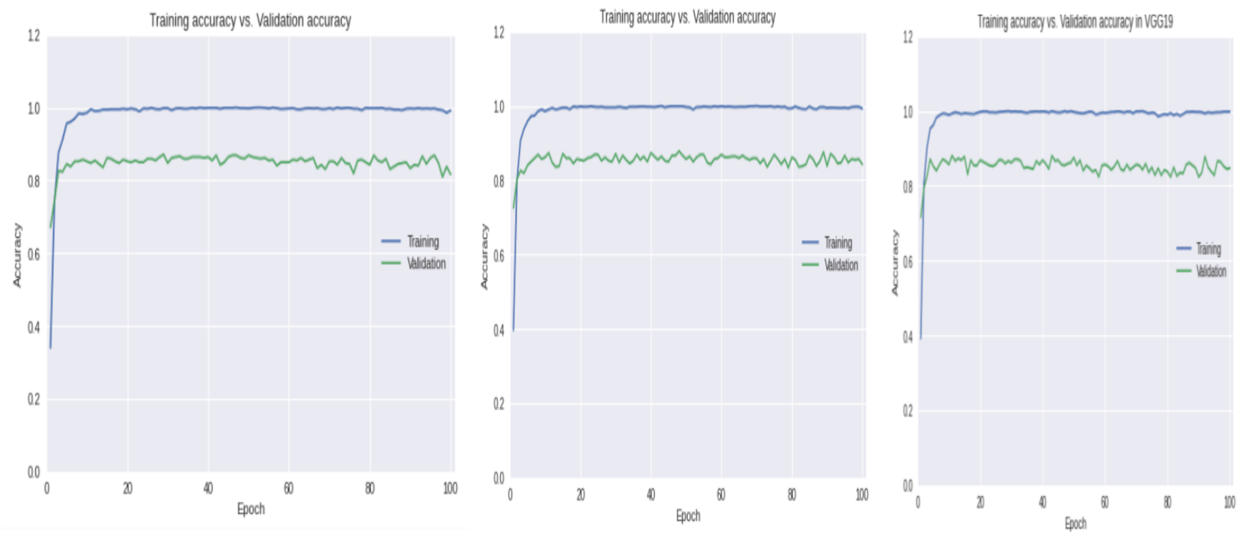
Class Name	Class label	Precision	Recall	F1-score	Support
Agricultural	0	1.00	1.00	1.00	20
Airplane	1	1.00	0.90	0.95	20
Baseball diamond	2	1.00	0.85	0.92	20
Beach	3	0.95	0.95	0.95	20
Buildings	4	0.94	0.85	0.89	20
Chaparral	5	1.00	1.00	1.00	20
Dense residential	6	0.52	0.85	0.64	20
Forest	7	0.86	0.90	0.88	20
Freeway	8	1.00	0.90	0.95	20
Golf course	9	0.83	0.50	0.62	20
Harbor	10	1.00	1.00	1.00	20
Intersection	11	0.82	0.90	0.86	20
Medium residential	12	0.85	0.85	0.85	20
Mobile home park	13	0.82	0.45	0.58	20
Overpass	14	0.85	0.85	0.85	20
Parking lot	15	1.00	1.00	1.00	20
River	16	0.61	0.95	0.75	20
Runway	17	0.87	1.00	0.93	20
Sparse residential	18	1.00	0.95	0.97	20
Storage tanks	19	1.00	1.00	1.00	20
Tennis court	20	1.00	0.85	0.92	20
Average Accuracy in each Measures		**0.90**	**0.88**	**0.88**	**420**

The accuracies of the three TL models for training and validation data are shown in Figure 5.3, and the overall accuracies are recorded in Table 5.6. Since F1-Score finds the harmonic mean of precision and recall, it shows how precise and reliable the classification model is.

The categorical-cross-entropy loss function was used while compiling the model. For the correct class, the value of the loss function becomes closer to 0, as we observed in Figure 5.4.

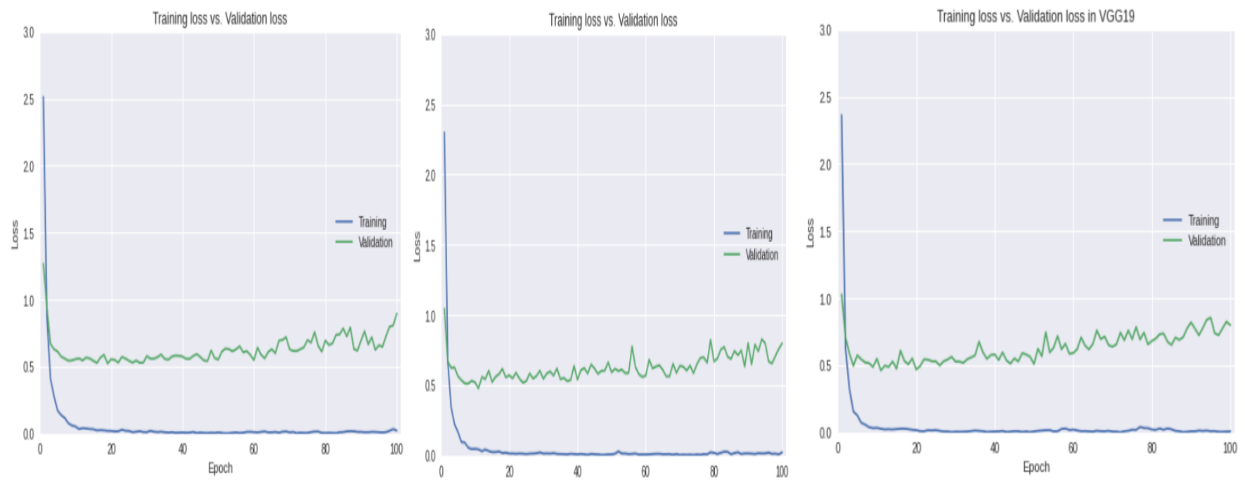
Table 5.6. Number of early stoppings at epoch #, time is taken for training and overall accuracy in the three models

Architecture	Total #of layers	Weight layers used	Early stopping at epoch#	Time (s)	OA
Resnet50V2	152	3*3	19	6	92.46
InceptionV3	22	5*5	18	9	94.36
VGG19	16	3*3	95	5	**99.64**



a) Accuracy of resnet50V2 b) Accuracy of InceptionV3 c) Accuracy in VGG19

Figure 5.3. Accuracies in Training vs. Validation for TL Classification Models



a) Loss in resnet50V2 b) Loss in InceptionV3 c) Loss in VGG19

Figure 5.4. Losses in Training vs. Validation for TL Classification Models

5.4 Discussions

5.4.1 Discussions on results, methods, and TL performances

The experiments described in Section 5.3.2 prove that the pretrained models are applicable to LCLU classification in RS images. Accuracy is also our aim, and we got better results in each model and most individual classes. We evaluated each class's accuracy using precision, recall, and F1-score measurements for each model. Precision is outperformed, i.e., 88%, 88%, and 90% for all three models, as shown in

Table 5.3, Table 5.4 and

Table *5.5*, respectively. That means the relevant classes were retrieved and predicted correctly. If the F1-score is perfect (1), i.e., 100% accurate for certain classes, precision and recall are also perfect for all classes.

Precision was perfect in the agricultural, airplane, chaparral, freeway, overpass, and parking classes, as shown in

Table 5.3 and Table 5.7 for the Inception_v3 model. On the other hand, the medium-residential class had the worst result, with a precision of only 64%. The best recall results were in the airplane, chaparral, harbor, parking lot, and runway classes, while the worst were in the golf course and dense residential classes, with 45% and 50% recall results, respectively. Also, the f1-score is most accurate for classes like airplane, chaparral, and parking lot, while it is least accurate for dense golf course (56%) and residential (59%). For similar situations, we grouped classes according to their best, worst, or poorest value under each Resenet50v2 and VGG19 model measurement in Table 5.7.

In all three models, the medium and dense residential classes have the lowest precision, and the golf course has the lowest recall. The poorest accuracy results could be the cause of image variant similarity and resolution differences.

Table 5.7. The best and worst class accuracies in precision, recall, and F1-score measurements

Model	Precision		Recall		F1-score	
	Perfect (1) class	Worst class/value	Perfect (1) class	Worst class/value	Perfect (1) class	Worst class/value
Inception_v3 (Table 5.3)	Agricultural, Airplane, Chaparral, Freeway, Overpass, Parking	Medium residential/ 0.64	Airplane, Chaparral, Harbor, Parking lot, Runway	Golf course/0.45 and Dense residential/ 0.50	Airplane, Chaparral, Parking lot	Golf course /0.56 and Dense residential /0.59
Resnet50v2 (Table 5.4)	Agricultural, Airplane, Baseball diamond, Chaparral, Harbor, Overpass, Parking lot, Runway, Storage tanks, and Tennis court	Dense residential /0.56 and Medium residential /0.65	Agricultural, Airplane, Chaparral, Freeway, Harbor, and Runway	Golf course /0.55	Agricultural, Airplane, Chaparral, Harbor, and Runway	Dense residential /0.62 and Golf course /0.67
VGG19 (Table 5.5)	Agricultural, Baseball diamond, Chaparral, Freeway, Harbor, Parking lot, Sparse residential, Storage tanks, and Tennis court	Dense residential /0.52	Agricultural, Chaparral, Harbor, Parking lot, Runway and Storage tanks	Mobile home park /0.45 and Golf course /0.50	Agricultural, Chaparral, Harbor, Parking lot, and Storage tanks	Mobile home park /0.58

By applying the hyperparameters listed in Table 5.2, the TL model has been modeled using the Adam optimizer with a LR of 0.0001 and compiled with the categorical-cross-entropy loss function. The loss function predicts an integer value for each class N assigned from 0 to N-1 in the UCM dataset, where N = 21 classes. The cross-entropy loss has become lower and lower for the deeper network training process to identify the correct class. A correct cross-entropy value is 0 for a correct class. The value of the cross-entropy loss function increases for misclassified classes, and the trained network fails to find the correct class [86]. In Figure 5.4, the training loss graph with a blue color is closer to 0. So, the trained network is good for predicting the correct class in TL.

In addition to dropout, we used the early stopping technique to reduce overfitting and improve performance. The training was stopped early when either the performance of the validation loss stopped decreasing even though the performance of the training loss decreased or the performance of the validation accuracy stopped increasing even though the performance of the training accuracy increased. We assigned the epoch value 100, and the early stopping stopped at epochs 19, 18, and 95 randomly when validation loss stopped decreasing for Resnet50v2, InceptionV3, and VGG19, respectively. In this study, we observed that the larger number of epochs of early stopping produced greater accuracy.

Therefore, VGG19 has superior TL model performance. As shown in Table 5.6, the VGG19 model outperformed all other method with a superior accuracy of 99.64% for an 80% training ratio.

5.4.2 Discussions on similar studies

In this study, we utilize the Resnet50V2, InceptionV3, and VGG19 as our baselines. We compared the classification performance accuracy of this study with the UCM dataset's state-of-the-art classification studies, as stated in Table 5.8. According to Table 5.8, all of the proposed TL models outperformed the most state-of-the-art studies in terms of accuracy. We used the adaptive optimizer with the smallest LR value, i.e., 0.0001, while the others used SGD with various parameters. Most of the researchers listed in Table 5.8 have used the epoch number 50, but we have used 100 epochs and early stopping while validating the model.

Therefore, the proposed VGG19 achieved the superior accuracy of 99.64% for a 70% training ratio among all methods we used. The Resnet50v2 model results in lower performance than the other two methods. The results in the three pretrained models demonstrate that the TL model can prove its availability on RS images.

Table 5.8. Comparative state-of-the-art classification methods and OA (in %) on the UCM dataset

Authors	Methods	Dataset	Accuracy on UCM	Optimizer
[158]	**Resnet50,** VGG-16	EuroSAT	**99.04,** 98.14	-
[142]	**ResNet50,** VGG16, Inception-v4	**UCM**, AID, NWPU	**95.95**, 92.50, 91.73	SGD
[139]	RSSCNet	**UCM, RSSCN7**, WHU-RS19	**99.81**, **97.41**, 99.46	SGD
[60]	**Inception-V3, VGG-19**	**UCM**, AID, PatternNet	**91.00, 94.3**	SGD, **Adam** and Adamax
[140]	VGG-16: with multiple pyramid pooling	**UCM**, NWPU	**93.24,** 88.62	SGD
[124]	VGG-16- CapsNet and Inception-V3-CapsNet	AID, **UCM**, NWPU	**98.81** 99.02	SGD
[70]	bag-of-visual-words (BOVW)	**UCM**	**81.19**	-
Proposed	**Resnet50V2, Inception-V3 and VGG-19**	**UCM**	**92.46, 94.36 99.64**	**Adam (Adaptive)**

5.5 Chapter Summarization

In this chapter, we addressed the problem of LCLU classification in RS images using deep TL models with bottleneck feature extraction. Our objective was to apply the TL model and improve the classification performance for LCLU classification in RS images. The training time of TL is more efficient (trained in seconds) than the other deep CNN models (trained in days when they were trained from scratch), as observed in the state-of-the-art studies by [58], [146]. We used the bottleneck feature extraction method to make the training of the model go faster and be more accurate.

The model’s performance is also prominent in all models, i.e., 92.46%, 94.36%, and 99.64% accuracy results for Resnet50V2, InceptionV3, and VGG19, respectively. However, the superior accuracy is profound in the VGG19 model with efficient time. Most of the classes’ performances are characterized by prominent accuracy except for some classes, such as the medium residential, dense residential classes and the golf course, which have the poorest accuracy when evaluated by precision, recall, and F1-score.

The LCLU classification in RS image contributes significant values [60], [166] to rural and urban decision-making and planning. Our contribution is to use deep TL with bottleneck feature extraction to solve the LCLU classification problem using RS images. This contribution directs environmental resource management and sustainable development for agricultural and urban planning. In addition to this contribution, we evaluated and improved the performance of the TL models and proved their availability for the LCLU classification in RS images.

6. COMPARING THE PERFORMANCE OF CNN, TL AND FINE-TUNING MODELS FOR LCLU CLASSIFICATION

6.1 Introduction

The LCLU classification learning system is essential for environmental monitoring, agricultural decision-making, and urban planning in the contemporary dynamic world. The LCLU classification using RS images is a critical issue in managing natural resources and human-made activities that affect natural phenomena in the earth's environment. RS image classification is the most recently focused area for the RS societies in the computer vision trends and image processing research areas. From time to time, the world's population is increasing dramatically, and the demand for land use is increasing. A learning system could be applied to the domain to utilize this land properly.

Thus, the LCLU classification is the most recent hot and challenging task in RS [58], [66], [80], [167]. RS images are satellite data collected from the earth's environment using advanced sensor technologies. The DL method could be applied to solve the challenge.

The DL approach is a recent specialized ML approach that could automatically extract features of the image for large datasets with admirable performance improvements. Thus, DL is a recently focused research area applied in various domains, including classification [7], [8], [16], [80], [84], recognition [53], and object detection [168]. It is also potentially challenging in many other domains [169].

The DL techniques proposed in this objective are CNNs, TL, and fine-tuning, which make the classification task more attractive. CNN is one of computer vision's most common DL methods [18] for feature extraction and LCLU modeling using RS images. The CNN is a feedforward and backward neural network consisting of convolutional calculations and deep structures. Therefore, CNN models have powerful feature extraction capability for classification performance improvement in RS images [48].

Nevertheless, the DL algorithms such as CNN require a large amount of data and very high computational power to train the classification models [170] from scratch. Whereas TL [171] and fine-tuning [37], [58] can solve the classification problem in smaller dataset training samples and less training time. Thus, the main issue with deep CNN models is that training them from scratch requires a large dataset and takes longer. To solve such DL problems, we proposed the TL and fine-tuning approaches and compared their performances with the convolutional neural network feature extractor (CNN-FE) model.

TL is another recent DL technique used to train the DL model by reusing pretrained networks. TL and fine-tuning are used for smaller datasets and can be made from the top fully connected layer of a network that has already been trained, so that the features can be used again. The training time in TL and fine-tuning could be much less than that of deep CNN model. So, TL could solve the problems of building DL models from scratch by training the models in less time with smaller datasets by freezing the network that has already been trained.

TL adopts the features from the pretrained network to train the new models. Moreover, fine-tuning is a DL technique used to train the model by unfreezing the pretrained networks. This technique is vital to increasing the performance of the model. The TL adopts the properties of the pretrained layers, excluding the last fully connected layer, i.e., the dense layer, which is replaced by our classifier with a number of neurons of 21 and an activation function of softmax.

This objective of the study designed and evaluated the DL models CNN, TL, and fine-tuning. The CNN has been developed with four CNN blocks. Using Keras applications, the deep CNN-based TL and fine-tuning models have been developed on the pretrained model EffificientNetB7 [93].

Few studies have been conducted in recent years to compare the capabilities of DL models developed from scratch with those developed using the pretrained network. For instance, [115] has applied the TL and fine-tuning methods to the ResNet50 pretrained network and compared their performances with other pretrained based networks in scene image classification. However, the scratch development models' evaluation and comparison with pretrained development models have not been widely researched. We chose the recently pretrained network, EffificientNetB7, which was trained on the "ImageNet" large dataset, for designing the TL and fine-tuning the model. Recently, [93] achieved 84.4% top-1 accuracy of the state-of-the-art EffificientNetB7 on the “ImageNet.” According to [93], eight scaling-up series of EffificientNet pretrained models from EffificientNetB0 through EffificientNetB7 were designed on the larger dataset called "ImageNet." The performance of each successive version has improved.

According to [172], who have applied EfficientNetB3, larger versions of EfficientNet models perform better than smaller ones. Thus, we proposed the EffificientNetB7 pretrained network to design TL and fine-tuning models in the domain of LCLU classification using RS images to evaluate their performances and compare them with the CNN-FE model. We selected the UCM dataset to assess and compare the DL models.

Therefore, this chapter aims to design the DL models and evaluate their performance with various performance measurement metrics. First, we developed the CNN-FE model and compared its performance with the deep TL and fine-tuned models for LCLU classification using the UCM dataset. Second, we applied the recent advanced EfficientNetB7 pretrained network to design TL and fine-tune DL models for LCLU classification in RS images. Then finally, we evaluated the models, compared their performances using different performance evaluation metrics, and concluded that the fine-tuning model improved performance with efficient training time.

6.2 Materials and Proposed Methods

6.2.1 Datasets and Tools

We used the publicly available University of California Merced (UCM) dataset for modeling the CNN, CNN-based TL, and fine-tuning. The UCM dataset is an LCLU data set collected from the earth, labeled manually, and introduced by [70] at the University of California Merced. It contains twenty-one classes. Each class contains 100 images with 256 × 256 pixels resolution and a spatial resolution of about 30 centimeters per pixel. However, the UCM dataset is inconsistent since about 44 images have different pixel shapes. This dataset is available at:

http://weegee.vision.ucmerced.edu/datasets/landuse.html.

As a tool, the Python high-level computer language is used. Python is a versatile and user-friendly programming language that can be used to create many interactive libraries for the DL model. Tensorflow and Keras are also other DL tools used with Python.

6.2.2 Proposed DL Methods

Previously, the DL method was investigated for classification problems in RS images from various datasets. However, evaluating and comparing the DL developed from scratch with those trained on pretrained networks has not been widely investigated yet. Further investigations are still needed to design and assess the current DL techniques for LCLU classification using the RS datasets.

Thus, to evaluate and compare the performances of different DL models applied for LCLU classification problems in the UCM RS dataset, we designed the CNN-FE model, the TL model and the fine-tuned model on an EfficientNet pretrained network. The EfficientNet was trained on the large-size dataset of "ImageNet" images. ImageNet is the most significant benchmark dataset introduced by [87] for designing DL models.

CNN's performance was influenced by the DL hyperparameters [24]. For instance, according to [3], using different dropout values produced different performance results. We also showed that the dropout value (0.25) generated an accuracy of 84.76%, which is different from our previous work with the dropout value (0.50), which caused an accuracy of 89.76%. Therefore, by considering their effects, we set the same hyperparameters for all three DL models on the given dataset to evaluate the models' performances, as shown in Table 6.1.

6.2.2.1 The convolutional neural network (CNN) algorithm

The CNN algorithm is the most critical DL technique that could extract and automatically learn features from the data. From the input images, features are newly extracted and learned weights of pixels in the image in the new value (usually reduced). The CNN method consists of several sets of connected layers. These layers shared weights throughout the process, i.e., from the start to the end layer (classifier), as depicted in Figure 6.1. This process creates the feature map for the model's entire set of layers as well as the class prediction for the output layer. The feature map of the model can be built up with pixel-wise multiplication of the input image pixels and the provided weight or kernel pixels with learnable parameters [127], [173].

The CNNs can be capable of spatial feature representations for RS image classifications using the convolution technique in the form of pixels [174]. This convolution process updates weights with each layer's provided non-linear activation function. The input data types and weight calculations in the convolution method make the CNNs different from other conventional ML approaches [123].

In DL model training, Relu and Softmax non-linear activation functions are the most relevant functions to update the weights in the convolution process. We used the Relu at the entire convolutional layers to activate the weights in each convolution process and the softmax at the output layer since it is reliable for multiclass classification problems. The softmax function is a feature classifier based on a probability score for each class.

As we discussed in Chapter 3 of this thesis, the number of convolutional and dense parameters (params) could be calculated using equations (3.5) and (3.6), respectively, in the convolution process. The total parameter numbers of the model are the summations of the computed results from the Conv2D and dense layers. We designed the CNN-FE model with four Conv2D layers that calculate the number of parameters for those layers in the same norm (3.5) and two dense layers (3.6). However, the calculation formula for dense parameters differs from Conv2D, as equated in (3.6). The number 1 means the bias associated with each filter for learning.

We could get a total calculated parameter number according to (3.5) and (3.6). However, the number of parameters for all MaxPooling2D and Flatten layers is zero because these layers do not learn anything from weights (filters) or the built model. As a result, 1.68 million parameters were found and learned in the CNN-FE model, while 18.88 million parameters were found and learned in both the TL and fine-tuning models.

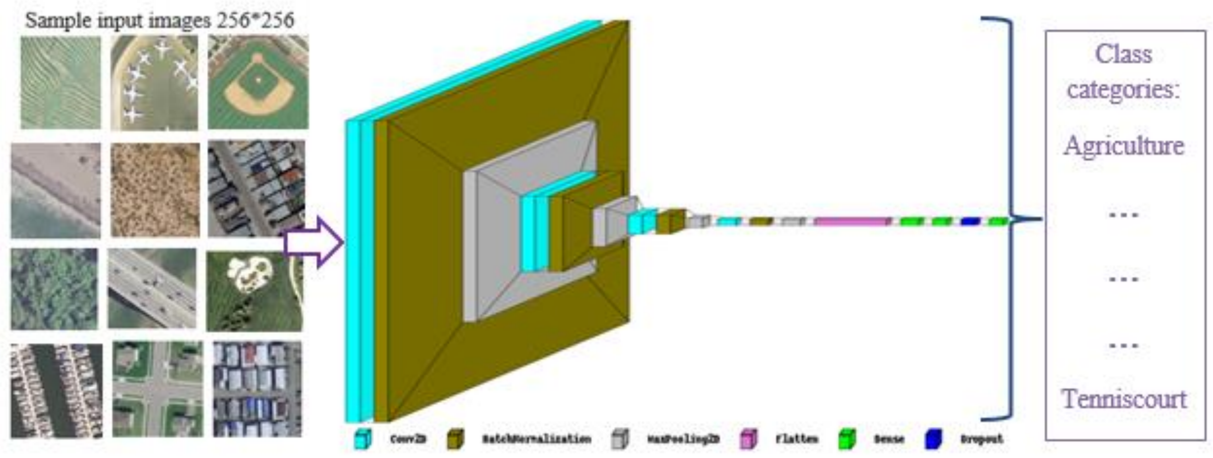


Figure 6.1. Layers of CNN-FE model with the input sample images

6.2.2.2 The deep transfer learning (TL) method

TL is a method of training the DL model by replacing the input layer with an image embedding as the EfficientNet transfers the knowledge learned from the much larger dataset called "ImageNet" to our classification problem. The TL has been trained by making the layers in EfficientNet on ImageNet images non-trainable (pre_trained_model.trainable = False). We trained only the last flatten (1D vector form) and two dense layers, including Relu and Softmax activation functions, and dropout optimization on the 21 LCLU RS UCM dataset classes. Therefore, the classification head with dense layers can be appended to manipulate our new classification problem. TL is an efficient, reliable DL technique used to propose various domains, especially the image classification problem in this study. Recently, TL has been applied for LCLU in RS image classifications [142], [158]. TL is used to train DL models in a short amount of time with improved results [129], [146]. However, deep TL is used for limited dataset training samples, while deep CNN from scratch is used for large dataset training samples. Therefore, we applied the TL model in this objective of the study to compare its performance with other DL techniques for LCLU classification in RS images.

6.2.2.3 The fine-tuning technique on EfficientNet

Fine-tuning is a DL technique used to train a model by allowing and adapting the EfficientNet pretrained layers on the ImageNet large dataset to be trainable (pre_trained_model.trainable = True). EfficientNet is a recent advanced CNN-based network that could be applied to classification tasks on ImageNet. To get the improved performance, EfficientNet has been fine-tuned, and the final fully connected layer is treated as the output classifier layer as we did in TL, except the layers are allowed to be trained.

As stated by [58] and [37], fine-tuning a pretrained network is the optimal solution for a limited number of training samples. The EfficientNet pretrained network was introduced by [93] for rethinking model scaling for CNN. We selected the EfficientNet pretrained network as it is the most recent and advanced network, which has not been applied yet to the CLCU classification domain.

Fine-tuning the EfficientNet pretrained network has been trained on the last three fully connected layers on the ImageNet. The final, fully connected layers include a flatten layer that transforms the input image into vector form, two dense layers, dropout, relu, and activation functions. Accordingly, the pretrained weights are used randomly as initial weights for our fine-tuning neural network. Fine-tuning is used to compare the results of the fully connected layer and the convolutional layer. Thus, we proposed a fine-tuning technique to compare its performance with the convolutional layer-based CNN-FE model and the fully connected layer-based TL model.

Table 6.1. The DL hyperparameters settings for training the datasets

Hyperparameters	Values
Optimizers	Adam
Learning rate	0.001
Batch size	64
Epochs	100
Loss function	Cross-entropy
Activation functions	Relu, softmax
Dropout	0.25

6.3 Experimental Results and Discussions

6.3.1 Experimental Setting and Results

We used the UCM dataset for experiments to design and evaluate the DL models for LCLU classification problems. We split each dataset used to train, validate, and test samples into 60%, 20%, and 20%, respectively. We also set the DL hyperparameters as indicated in Table 6.1. Then we trained the models, validated them with the validation dataset during training, and evaluated their performances with the test dataset.

After the experimental parameters were set, we trained and evaluated the model during and after the experiments with validation and test datasets, respectively. We evaluated the model's performance using accuracy, precision, recall, f1-score, and confusion matrix (CM) metrics. CM measures the class performance, whether classified correctly or incorrectly in rows-column intersections. In addition to the accuracy, we used the categorical-cross-entropy loss function to calculate the errors. The training and validation losses or mistakes are expected to decrease as the epochs increment, as shown in Figure 6.2, Figure 6.3 and Figure 6.4 (on the **right**).

Therefore, we evaluated the models with 420 test or support images, as shown in Table 6.2, Table 6.3, and Table 6.4, using the UCM dataset. The UCM is an imbalanced RS dataset. The accuracy performance metric, the percentage of correctly classified images, could not be suitable for the imbalanced dataset. Thus, each class's performance is evaluated using errors, precision, recall, f1-score, and CM metrics in addition to the accuracy metric. The f1-score is the harmonic mean of precision and recall metrics, and it generalizes the performance of each class and the average performance of the DL models built. If both precision and recall have the best performance result in a category, then the f1-score has the best performance result in that class. Whereas, if either precision or recall has a **0** performance result, then the f1-score has **0** performance, which is nothing the model is predicting.

Accordingly, the categories that have scored best (100%) in the f1-score metric are agricultural and chaparral in CNN-FE (Table 6.2); chaparral, parkinglot, and storagetanks in the TL model (Table 6.3); and agricultural, airplane, chaparral, freeway, and runway in a fine-tuning model (Table 6.4), respectively.

Table 6.2. CNN-FE classification performances in precision, recall, and f1-score on 420 support images

Class name	Precision	Recall	F1-score	Support
Agricultural	1.00	1.00	1.00	20
Airplane	1.00	0.85	0.92	20
Baseballdiamond	0.89	0.80	0.84	20
Beach	1.00	0.95	0.97	20
Buildings	0.64	0.70	0.67	20
Chaparral	1.00	1.00	1.00	20
Denseresidential	0.50	0.55	0.52	20
Forest	0.86	0.95	0.90	20
Freeway	1.00	0.90	0.95	20
Golfcourse	0.80	0.60	0.69	20
Harbor	0.87	1.00	0.93	20
Intersection	0.88	0.70	0.78	20
Mediumresidential	0.53	0.85	0.65	20
Mobilehomepark	0.93	0.65	0.76	20
Overpass	0.94	0.75	0.83	20
Parkinglot	1.00	0.90	0.95	20
River	0.76	0.95	0.84	20
Runway	0.87	1.00	0.93	20
Sparseresidential	0.83	0.95	0.88	20
Storagetanks	0.95	0.95	0.95	20
Tenniscourt	0.94	0.80	0.86	20
Average performance	**0.87**	**0.85**	**0.86**	**420**

Table 6.3. TL classification performance in precision, recall, and f1-score on 420 support images

Class name	Precision	Recall	F1-score	Support
Agricultural	1.00	0.95	0.97	20
Airplane	0.95	1.00	0.98	20
Baseballdiamond	1.00	0.90	0.95	20
Beach	0.95	0.95	0.95	20
Buildings	0.90	0.90	0.90	20
Chaparral	1.00	1.00	1.00	20
Denseresidential	0.56	0.50	0.53	20
Forest	0.82	0.90	0.86	20
Freeway	1.00	0.95	0.97	20
Golfcourse	0.92	0.55	0.69	20
Harbor	0.91	1.00	0.95	20
Intersection	0.83	0.95	0.88	20
Mediumresidential	0.57	0.80	0.67	20
Mobilehomepark	0.67	0.50	0.57	20
Overpass	0.95	0.90	0.92	20
Parkinglot	1.00	1.00	1.00	20
River	0.69	0.90	0.78	20
Runway	1.00	0.95	0.97	20
Sparseresidential	0.83	0.95	0.88	20
Storagetanks	1.00	1.00	1.00	20
Tenniscourt	1.00	0.80	0.89	20
Average performance	**0.88**	**0.87**	**0.88**	**420**

Table 6.4. Fine-tuning classification performance in precision, recall, and f1-score on 420 support images

Class name	Precision	Recall	F1-score	Support
Agricultural	1.00	1.00	1.00	20
Airplane	1.00	1.00	1.00	20
Baseballdiamond	1.00	0.90	0.95	20
Beach	0.95	0.95	0.95	20
Buildings	0.80	0.80	0.80	20
Chaparral	1.00	1.00	1.00	20
Denseresidential	0.62	0.65	0.63	20
Forest	0.86	0.95	0.90	20
Freeway	1.00	1.00	1.00	20
Golfcourse	0.80	0.60	0.69	20
Harbor	0.95	1.00	0.98	20
Intersection	1.00	0.65	0.79	20
Mediumresidential	0.66	0.95	0.78	20
Mobilehomepark	0.61	0.55	0.58	20
Overpass	0.91	1.00	0.95	20
Parkinglot	0.95	1.00	0.98	20
River	0.73	0.95	0.83	20
Runway	1.00	1.00	1.00	20
Sparseresidential	0.89	0.80	0.84	20
Storagetanks	0.95	0.95	0.95	20
Tenniscourt	1.00	0.80	0.89	20
Average performance	**0.89**	**0.88**	**0.89**	**420**

The accuracy of the DL models is also measured in terms of accuracy and loss measurement metrics in graphical representation, as shown in Figure 6.2, Figure 6.3 and Figure 6.4 for CNN-FE, TL, and fine-tuning models, respectively. The training accuracies (with a blue color curve) are smoothly increasing, while the validation accuracies (with a red color curve) are somewhat fluctuating in increasing the accuracies in all models, especially in fine-tuning, as depicted in Figure 6.2, Figure 6.3 and Figure 6.4 (on the **left**). We used the cross-entropy loss function to reduce errors in the model performance. The training losses (with a blue color curve) are smoothly decreasing, while the validation losses (with a red color curve) are somewhat fluctuating in reducing the errors in all models, as depicted in Figure 6.2, Figure 6.3 and Figure 6.4 (on the **right**).

In addition to deploying precision, recalls, and f1-score, we used CM to evaluate class performances in each DL model. Like f1-score, better class performance is observed in most classes in the CM metric. The CM measures the class performance, whether it is classified correctly or incorrectly. CM considers each class label in rows (True labeled class) and columns (predicted labeled class), as depicted in Figure 6.5 through Figure 6.7. The probability score in the diagonal intersection showed the correct classified class. In contrast, the results in other rows-columns wise are predicted in misclassified classes.

For evaluating the model with test set sample images, we used the argmax function for predicting a class with the maximum argument probability score. For instance, our classification problem has twenty-one possible classes in the UCM dataset. If the output probabilities are [0.0, 0.0, 0.0, 0.0, 0.05, 0.0, 0.55, 0.0, 0.0, 0.0, 0.05, 0.0, 0.30, 0.05, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], the arg max (maximum argumentative-class) probability is 0.55 and it is associated with the dense-residential class that is predicted by the CNN-FE model as shown in Figure 6.5. Like ways, the arg max probability can correspond to each class prediction in CM metric. The sum of the output probabilities of each class is **1.00.**

While evaluating the class performance in CM metric, the lowest performance result in the two first classes are dense-residential (55%) and golf-course (60%) in CNN-FE, dense-residential (50%), mobile-home-park (50%) and golf-course (55%) in TL and mobile-home-park (55%) and golf-course (60%), as shown in Figure 6.5, Figure 6.6, and Figure 6.7, respectively. The lower result showed that the class property is mostly associated to other classes. For instance, the property of dense-residential has a more common feature with class medium-residential. The class performance in the CM metric is generally better in the fine-tuning model.

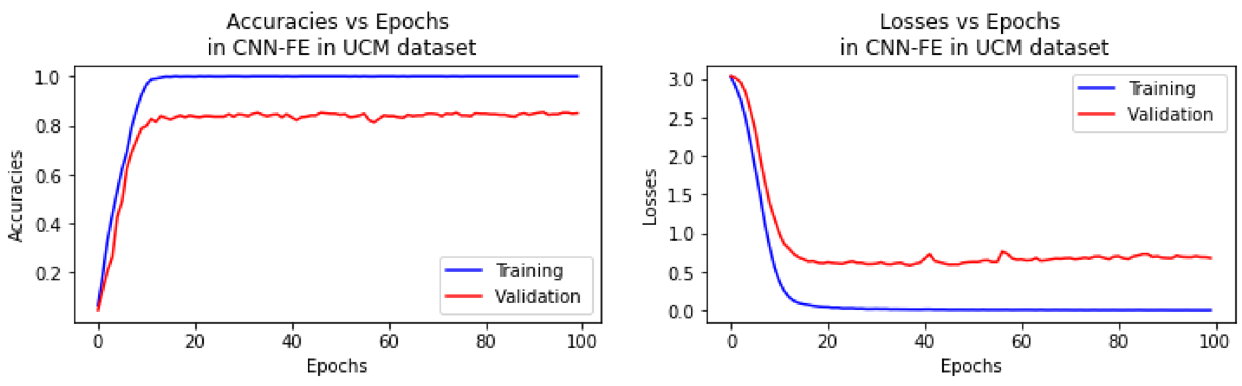


Figure 6.2. The training and validation accuracies and losses in the CNN-FE model

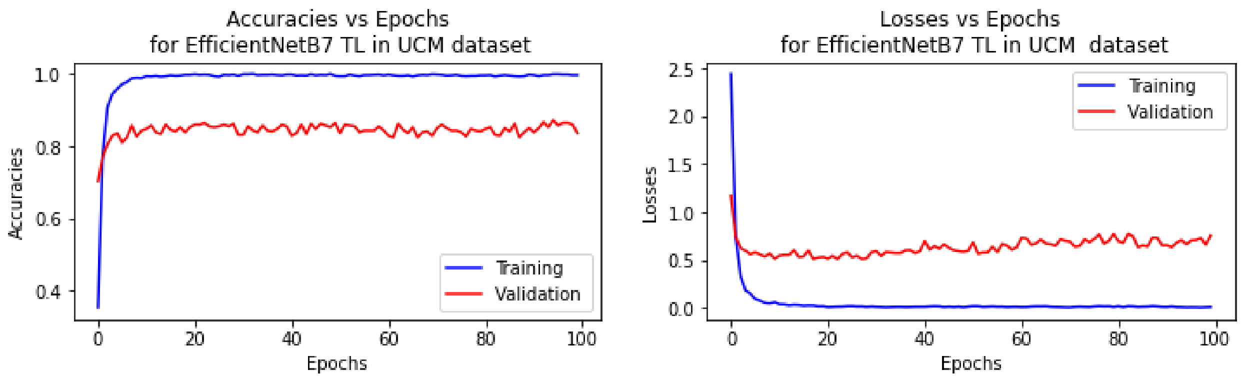


Figure 6.3. The training and validation accuracies and losses in the TL model

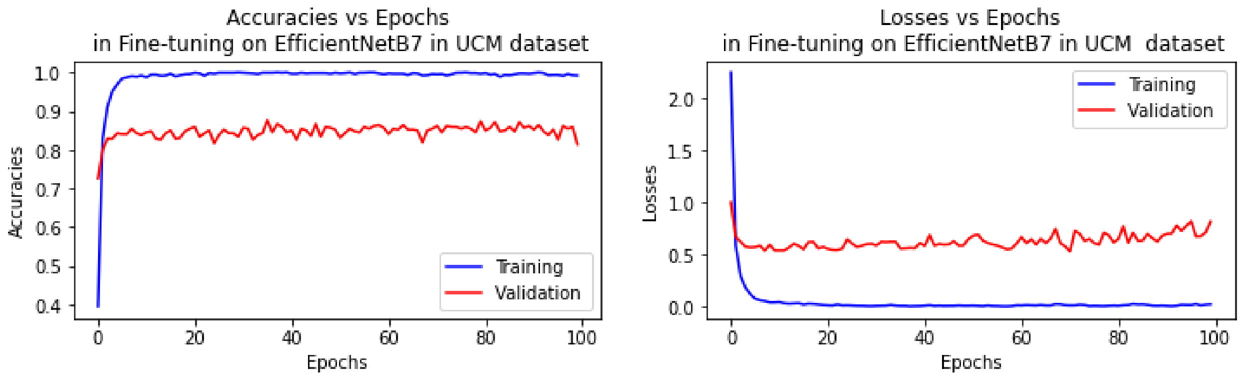


Figure 6.4. The training and validation accuracies and losses in the fine-tuning DL model

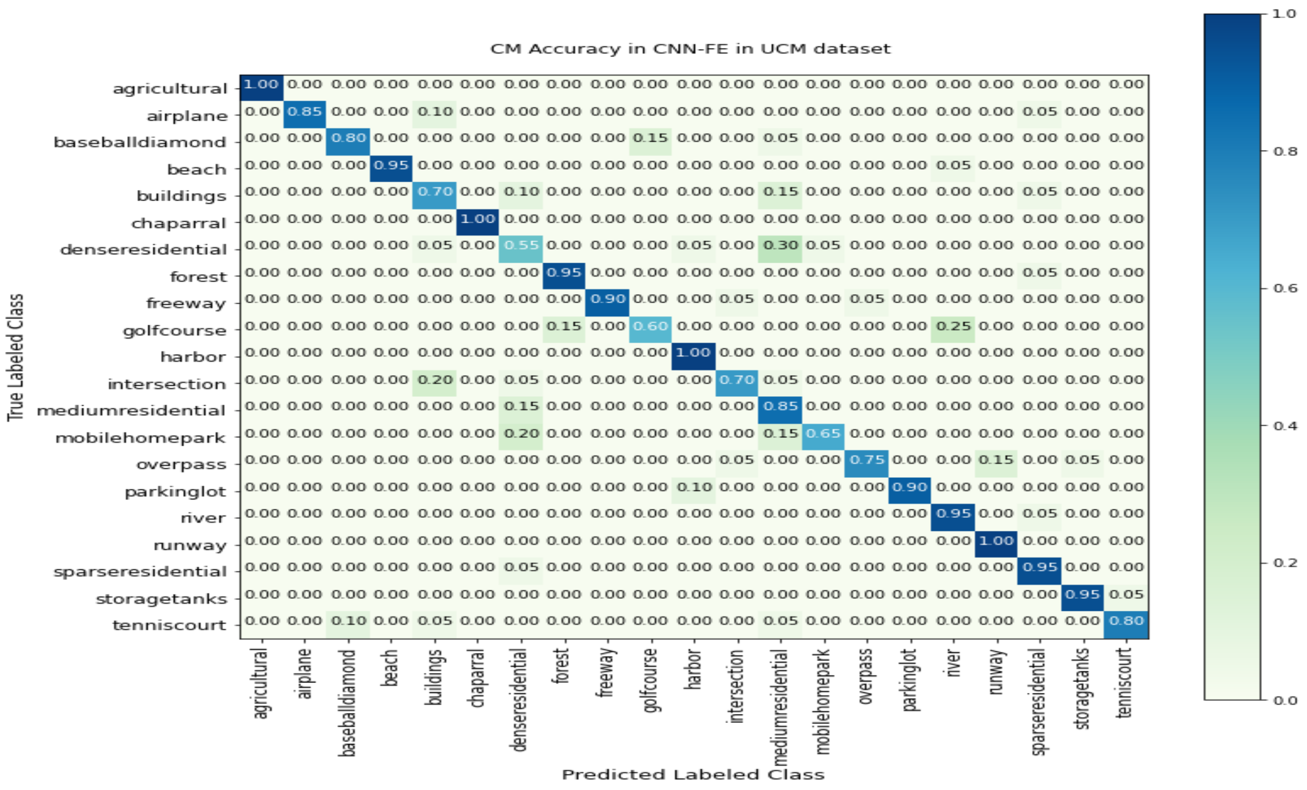


Figure 6.5. CM performance results for CNN-FE model in the UCM dataset

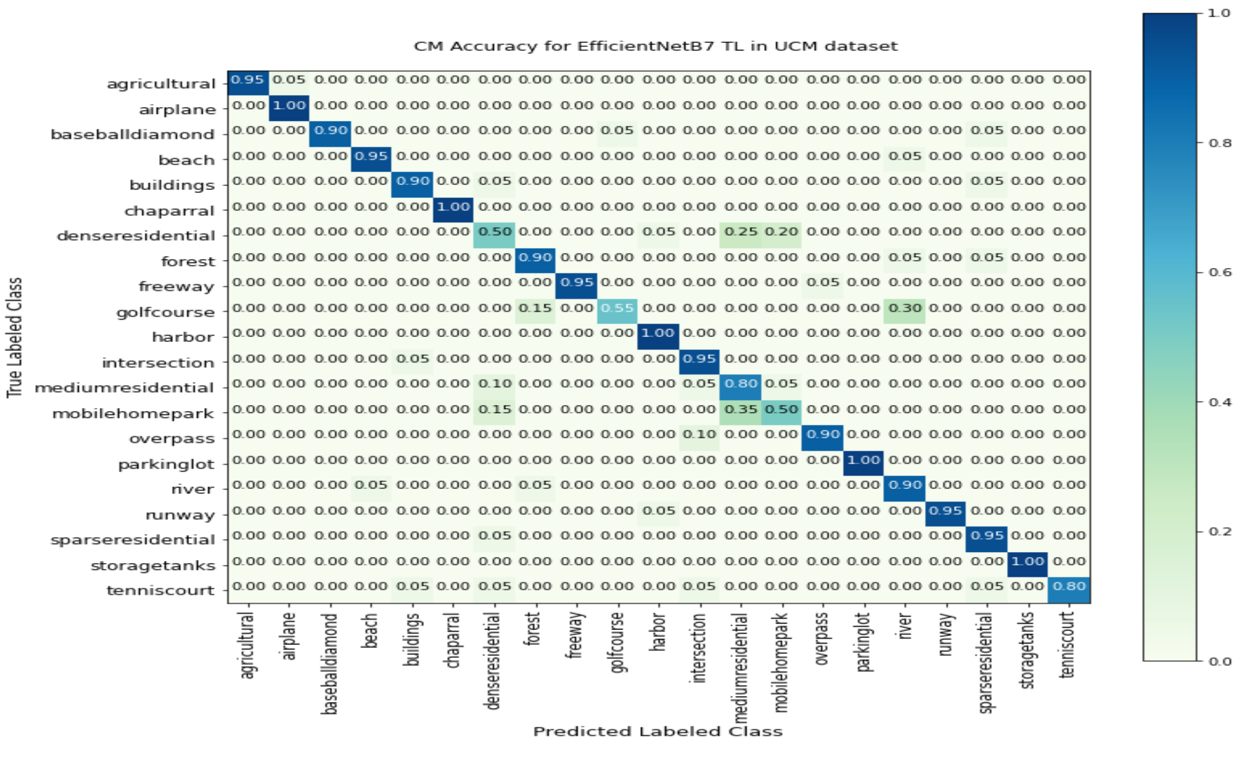


Figure 6.6. CM performance results for TL model in the UCM dataset

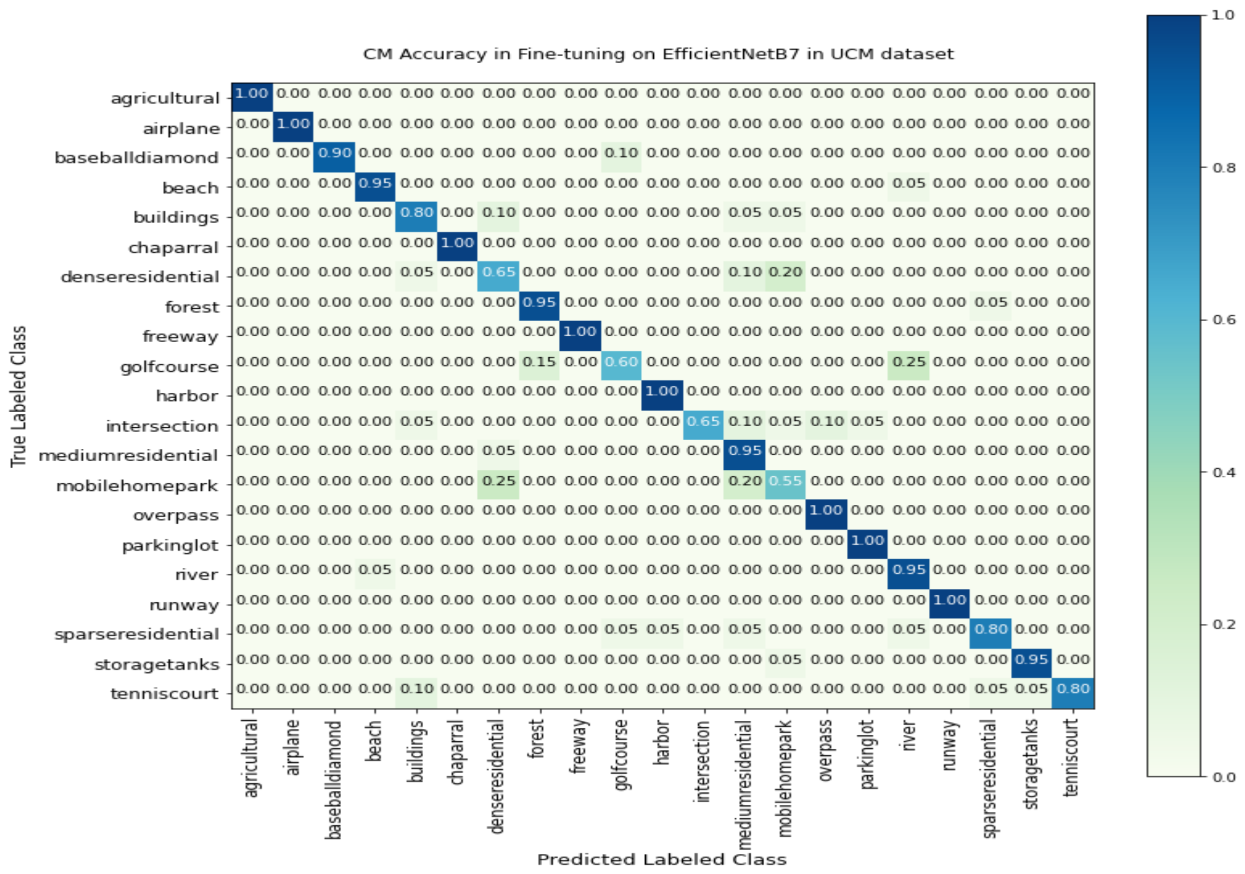


Figure 6.7. CM performance results for fine-tuning model in the UCM dataset

6.3.2 Discussions

This objective of the study applied the DL models for LCLU classification using RS images. The performances of these models resulted from various measurement metrics and showed good performance results for the classification problem, as shown in Table 6.5. The experimental results showed that the proposed DL algorithms could adapt and learn features of RS images since the Adam (adaptive movement estimation) learning rate took on that responsibility. The TL and fine-tuning performances are significantly improved over the CNN-FE.

To address our objective stated in this section, Table 6.2 through Table 6.4 and Figure 6.2 through Figure 6.7 compare the DL model performances on the UCM dataset. From the results, good class performance has been achieved in precision, recall, F1-score, and CM though some class performances scored lower in values. The training accuracy increases smoothly in CNN-FE, TL, and fine-tuning DL models, as shown in Figure 6.2, Figure 6.3 and Figure 6.4, respectively.

The overall accuracy for each model is summarized in Table 6.5. According to Table 6.5, the fine-tuning model has outperformed performance in accuracy (88%), precision (89%), recall (88%), and f1-score (89%) with efficient time. Whereas the CNN-FE model performed lower in each metric compared to the other two models, this could be why the dataset used was smaller. Moreover, the CNN-FE spent much more time training the model than the TL, and the fine-tuning.

The maximum capability of a number of parameters in EfficientNet is 64 million. In TL and fine-tuning models, 18.88 million parameters have been discovered and learned. This parameter number is about 18 times greater than the parameters found in CNN-FE mode, i.e., 1.68 million. This is why the convolution technique used in CNN-FE reduces the number of parameters. The fine-tuning technique is used to compare the performance of the DL models designed using the convolutional method and fully connected layers. As a result, improved performance in the fine-tuning model was achieved in less time than the other DL techniques used in this study's objective.

Designing the CNN model from scratch is essential to identify the correct properties of the categories for large datasets that are usually recommended when they exceed about 5000 images per class. However, it may require a significant amount of training time and be prone to overfitting. This limitation could be overcome by the less training time-consuming DL techniques, TL, and fine-tuning. The TL and fine-tuning DL techniques are efficient in terms of training time and produce improved performance results. But we recommend that TL and fine-tuning DL be applicable for small data sizes that may be less than 5,000 images per class. Therefore, we can conclude that the TL and fine-turning DL techniques are economically relevant in terms of time savings and essential for performance improvement, as observed in Table 6.5.

Table 6.5. The DL model performance evaluations using performance measurement metrics in the UCM dataset and the time (in seconds) consumed for training each DL model

DL Models	DL Performance results in each measurement metrics				Training	Params#
	Precision	Recall	F1-score	Accuracy	time (**Sec**.)	(**millions**)
CNN-FE	87.00	85.00	86.00	84.76	51.81	1.68
TL	88.00	87.00	88.00	87.38	41.77	88.88
Fine-tuning	**89.00**	**88.00**	**89.00**	**88.10**	**36.39**	18.88

6.4 Chapter Summarization

In this objective of the study, we designed the three DL models: CNN-FE, TL, and fine-tuning for LCLU classification problems using RS images. The TL and fine-tuning models have been trained on the recent EfficientNetB7 pretrained baseline network using the UCM dataset. The models' performances were evaluated using accuracy, precision, recall, f1-score, and CM metrics. The fine-tuned model in the UCM dataset has a profound accuracy result. We could observe that the nature of the dense-residential class is mainly similar to the properties of the medium-residential category. Thus, its performance results in precision, recall, f1-score, and CM is the worst result compared to other class categories. In addition to those metrics, the training time is another critical evaluation metric that is used to compare the economic advantages of TL and fine-tuning models over the DL models developed from scratch. We found that the TL and fine-turning DL models are efficient in saving time and essential for performance improvements.

7. EVALUATING THE PERFORMANCE OF DEEP LEARNING CLASSIFICATION MODELS IN VARIOUS REMOTE SENSED HYPERSPECTRAL IMAGES

7.1 Introduction

Land cover is the earth’s surface covered with natural resources and artificial activities and contains dynamic information [175]. On the other hand, land use is the ability of human activities to utilize natural resources on the land cover. These two entities are essential for human life on earth since they are the base for everything. In the contemporary, dynamic world, proven LCLU classification learning systems are required to manage and monitor the earth’s environment. Thus, LCLU classification is an important and challenging task in RS [4], [61], as it contains dynamic data. As stated in the previous sections, this dynamic data could be collected by using advanced RS technologies. The collected data could be RBG, multispectral, or hyperspectral images. The hyperspectral dataset has a larger number of continuous spectral bands [45].

RS images are vital information sources about the earth’s environment [7], [16], [66], [78] for LCLU classification problem analysis. The LCLU data are imbalanced, which caused the classification problem due to the model's imbalanced learning [14]. Thus, the recent challenging tasks in RS are the RS hyperspectral image (RSHI) classification [58], [66], [80], [167]. The DL approach could be applied to the hyperspectral image to solve the challenges.

DL is a recent specialized ML approach that attracts researchers for its powerful ability to analyze large datasets and its dynamic performance improvement. Therefore, DL is the recently focused research area applied to RSHI domains, such as classification [7], [8], [16], [80], [84], recognition [53], and object detection [168]. It is also potentially challenging in many other domains [169].

DL includes various techniques that are used to design task modeling, such as the classification task in this study. CNN is one of the most common DL methods in computer vision [18] and is used for feature extraction and LCLU modeling using RSHIs. Deep CNNs are the recent dominant paradigm in various domains. Thus, the CNN models have powerful feature extraction capability for classification performance improvement in RSHIs [48]. The Keras applications and deep CNN-based pretrained models used in this objective are EffificientNetB7 [93], InceptionV3 [90], [164], and MobileNet [92].

In recent related work studies, very few studies have been conducted to design the EffificientNetB7, MobileNet, and InceptionV3 DL models from various perspectives, such as considering their hyperparameters and different datasets collected in other locations. The deep CNN model for LCLU classification and crop identification in the Indian Pines dataset was evaluated by [45] with the optimizers (Adam, SGD, Adagrad, and RMSprop), the filter size (2 and 3), and the activation functions (Relu and Tanh). In the case of using different datasets with different locations, the deep CNN models such as InceptionV3 and VGG19 for AID, UCM, and PatternNet datasets have been evaluated by [24]. Therefore, we selected the most examined UCM dataset and the recently used SIRI-WHU and RSSCN7 datasets to assess and compare the recent EfficientNetB7, InceptionV3, and MobileNet DL models to understand their effects on this objective.

Therefore, this objective aims to design and evaluate DL models with various RSHIs that have different properties. The dynamic information collected on the earth's surfaces has different properties that could affect the model's performance [97], [111], [114].

7.2 Materials and Methods

7.2.1 Datasets

The publicly available HRSI datasets were collected from various sources on the web. On the base of the channel, there are three types of RS images: RGB (that consists of three channels), multispectral (that consists of more than three and under hundreds of channels), and hyperspectral (that consists of hundreds of channels). We used the UCM, SIRI-WHU, and RSSCN7 datasets as described in Table 2.1. The sample images from each class have been depicted in Figure 3.9, Figure 3.10, and Figure 3.11 for the UCM, SIRI_WHU, and RSSCN7 datasets, respectively.

The UCM dataset is an LCLU data set collected from the earth, labeled manually, and introduced by [70]. It has twenty-one classes, each with 100 images that measure 256 × 256 pixels and have a spatial resolution of about 30 cm per pixel. However, the UCM dataset is inconsistent, as about

44 images have different pixel shapes. This dataset is available at: http://weegee.vision.ucmerced.edu/datasets/landuse.html.

The SIRI-WHU dataset was collected from Google Earth and covered urban areas in China [75]. The dataset contains twelve categories with 200 images per category at 200 x 200 pixels. The dataset is available at: https://figshare.com/articles/dataset/SIRI_WHU_Dataset/8796980.

The RSSCN7 dataset is a challenging scene classification dataset collected from Google Earth and released by [41] at Wuhan University. The dataset is divided into seven categories, each of which contains 400 images with a 400 x 400 pixels resolution. We selected this dataset due to its higher pixel size and the fact that it has not been more thoroughly investigated as a domain yet. The dataset is available at: https://www.kaggle.com/datasets/yangpeng1995/rsscn7.

To this end, [176] and [177] have studied the performance of different methods in HRSI classification problems using the UCM and SIRI-WHU datasets, which are also used in this objective. In addition, the DBN was applied to the RSSCN7 HRSIs by [41].

7.2.2 DL Methods

Previously, the DL method was investigated for classification problems in RSHIs using commonly used datasets. However, further investigations are still needed to design and evaluate the current DL methods on the most commonly and recently used datasets.

Thus, to evaluate the effect of performances in different DL models for LCLU classification problems in different RSHIs datasets, we applied the deep CNN-based models, such as EfficientNet, InceptionV3, and MobileNet, to the selected two datasets described earlier. All of these DL models have been trained on the pretrained ImageNet images using the specified DL hyperparameters. The choices of DL hyperparameters have an impact on CNN performance [24]. We used almost all the same DL hyperparameters in this chapter for experimental settings as in designing deep CNN and TL, except the batch size is 128 instead of 64 and the learning rate is 0.001 instead of 0.0001. We set the same hyperparameters for all three DL models on the two datasets to evaluate the models' performances, as shown in Table 7.1.

Table 7.1. The DL hyperparameters settings for training the datasets

Hyperparameters	Chosen values
Optimizers	Adam with 0.0001
Batch size	128
Epochs	100
Loss function	Cross-entropy
Activation functions	Relu, softmax
Dropout	0.5

7.3 Experimental Results and Discussions

7.3.1 Experimental Setting and Evaluation Experimental Results

We used the UCM and SIRI-WHU datasets for experiments to design and evaluate the DL models for LCLU classification problems. We split each dataset to train, validate, and test samples into 60%, 20%, and 20%, respectively. We also set the DL hyperparameters as indicated in Table 7.1. Then we trained the models, validated them with the validation dataset during training, and evaluated their performances with the test dataset.

After the experimental hyperparameters were set, we trained and evaluated the model during and after the experiments with validation and test datasets, respectively. We evaluated the DL models using accuracy, precision, recall, f1-score and confusion matrix (CM) metrics. CM measures the class performance whether it is classified correctly or incorrectly in rows-columns intersections. In addition to the accuracy, we used the categorical-cross-entropy loss function to calculate the errors. The training and validation losses or mistakes are expected to decrease in epoch increments, as shown in Figure 7.1 (**a to c right**), Figure 7.2 (**a to c right**) and Figure 7.3 (**a to c right**).

We evaluated the EfficientNetB7, InceptionV3, and MobileNet models with 20, 40, and 80 test support images in the UCM (Table 7.2, Table 7.5, and Table 7.8), SIRI-WHU (Table 7.3, Table 7.6, and Table 7.9), and RSSCN7 (Table 7.4, Table 7.7, and Table 7.10) datasets, respectively. Each class's performance is good in each metric, especially in the harmonic mean metric f1-score. However, we observed the better class performance in the UCM dataset even though similar classes were found in the SIRI-WHU and RSSCN7 datasets, as shown in Table 7.2 through Table

7.10. On the UCM dataset, the best (100%) F1-score categories are agricultural, chaparral, harbor, parking lot, and runway in EfficientNetB7 (Table 7.2), agricultural in InceptionV3 (Table 7.5), and airplane, chaparral, and freeway in MobileNet (Table 7.8), whereas no classes scored best (100%) F1-score in the SIRI-WHU and RSSCN7 datasets. This is because the properties of the datasets are different.

Table 7.2. EfficientNetB7 classification reports for the UCM dataset

Class name	Precision	Recall	F1-score	Support
Agricultural	1.00	1.00	1.00	20
Airplane	0.95	1.00	0.98	20
Baseballdiamond	0.94	0.80	0.86	20
Beach	0.86	0.95	0.90	20
Buildings	0.76	0.65	0.70	20
Chaparral	1.00	1.00	1.00	20
Denseresidential	0.65	0.65	0.65	20
Forest	0.71	1.00	0.83	20
Freeway	0.95	0.95	0.95	20
Golfcourse	0.75	0.75	0.75	20
Harbor	1.00	1.00	1.00	20
Intersection	0.84	0.80	0.82	20
Mediumresidential	0.62	0.80	0.70	20
Mobilehomepark	0.52	0.70	0.60	20
Overpass	0.94	0.75	0.83	20
Parkinglot	1.00	1.00	1.00	20
River	0.75	0.45	0.56	20
Runway	1.00	1.00	1.00	20
Sparseresidential	0.89	0.85	0.87	20
Storagetanks	1.00	0.95	0.97	20
Tenniscourt	0.94	0.80	0.86	20

Table 7.3. EfficientNetB7 classification reports for the SIRI-WHU dataset

Class name	Precision	Recall	F1-score	Support
Agriculture	0.58	0.45	0.51	40
Commercial	0.82	0.82	0.82	40
Harbor	0.80	0.82	0.81	40
idle_land	0.67	0.72	0.70	40
Industrial	0.94	0.78	0.85	40
Meadow	0.53	0.50	0.51	40
Overpass	0.95	0.90	0.92	40
Park	0.65	0.78	0.70	40
Pond	0.74	0.72	0.73	40
Residential	0.80	0.88	0.83	40
River	0.76	0.78	0.77	40
Water	0.89	0.97	0.93	40

Table 7.4 EfficientNetB7 classification reports for the RSSCN7 dataset

Class name	Precision	Recall	F1-score	Support
Field	0.70	0.91	0.79	80
Forest	0.82	0.94	0.88	80
Grass	0.65	0.53	0.58	80
Industry	0.73	0.55	0.63	80
Parking	0.72	0.78	0.75	80
Resident	0.88	0.72	0.79	80
RiverLake	0.89	0.97	0.93	80

Table 7.5. InceptionV3 classification reports for the UCM dataset

Class name	Precision	Recall	F1-score	Support
Agricultural	1.00	1.00	1.00	20
Airplane	0.95	1.00	0.98	20
Baseballdiamond	0.89	0.80	0.84	20
Beach	0.86	0.95	0.90	20
Buildings	0.87	0.65	0.74	20
Chaparral	1.00	0.95	0.97	20
Denseresidential	0.87	0.65	0.74	20
Forest	0.86	0.90	0.88	20
Freeway	0.95	1.00	0.98	20
Golfcourse	0.82	0.45	0.58	20
Harbor	0.95	1.00	0.98	20
Intersection	0.93	0.65	0.76	20
Mediumresidential	0.53	0.90	0.67	20
Mobilehomepark	0.64	0.80	0.71	20
Overpass	1.00	0.80	0.89	20
Parkinglot	1.00	0.90	0.95	20
River	0.57	0.80	0.67	20
Runway	0.91	1.00	0.95	20
Sparseresidential	0.86	0.95	0.90	20
Storagetanks	1.00	0.90	0.95	20
Tenniscourt	0.89	0.85	0.87	20

Table 7.6. InceptionV3 classification reports for the SIRI-WHU dataset

Class labels	Precision	Recall	F1-score	Support
Agriculture	0.80	0.50	0.62	40
Commercial	0.90	0.88	0.89	40
Harbor	0.78	0.88	0.82	40
Idle_land	0.83	0.72	0.77	40
Industrial	0.80	0.90	0.85	40
Meadow	0.77	0.57	0.66	40
Overpass	0.85	0.97	0.91	40
Park	0.61	0.50	0.55	40
Pond	0.63	0.90	0.74	40
Residential	0.82	0.90	0.86	40
River	0.82	0.70	0.76	40
Water	0.85	1.00	0.92	40

Table 7.7 InceptionV3 classification reports for the RSSCN7 dataset

Class name	Precision	Recall	F1-score	Support
Field	0.72	0.89	0.79	80
Forest	0.85	0.94	0.89	80
Grass	0.73	0.61	0.67	80
Industry	0.62	0.49	0.55	80
Parking	0.67	0.75	0.71	80
Resident	0.82	0.64	0.72	80
RiverLake	0.79	0.91	0.85	80

Table 7.8. MobileNet classification reports for the UCM dataset

Class name	Precision	Recall	F1-score	Support
Agricultural	0.95	0.95	0.95	20
Airplane	1.00	1.00	1.00	20
Baseballdiamond	1.00	0.85	0.92	20
Beach	0.83	0.95	0.88	20
Buildings	0.88	0.75	0.81	20
Chaparral	1.00	1.00	1.00	20
Denseresidential	0.75	0.60	0.67	20
Forest	0.82	0.90	0.86	20
Freeway	1.00	1.00	1.00	20
Golfcourse	0.81	0.65	0.72	20
Harbor	0.95	1.00	0.98	20
Intersection	0.68	0.75	0.71	20
Mediumresidential	0.51	0.90	0.65	20
Mobilehomepark	0.69	0.55	0.61	20
Overpass	1.00	0.70	0.82	20
Parkinglot	1.00	0.95	0.97	20
River	0.63	0.85	0.72	20
Runway	1.00	0.95	0.97	20
Sparseresidential	0.89	0.85	0.87	20
Storagetanks	0.87	1.00	0.93	20
Tenniscourt	1.00	0.70	0.82	20

Table 7.9. MobileNet classification reports for the SIRI-WHU dataset

Class labels	Precision	Recall	F1-score	Support
Agriculture	0.93	0.68	0.78	40
Commercial	0.97	0.95	0.96	40
Harbor	0.82	1.00	0.90	40
Idle_land	0.83	0.85	0.84	40
Industrial	0.92	0.85	0.88	40
Meadow	0.61	0.57	0.59	40
Overpass	0.95	1.00	0.98	40
Park	0.82	0.78	0.79	40
Pond	0.88	0.88	0.88	40
Residential	0.91	0.97	0.94	40
River	0.84	0.95	0.89	40
Water	1.00	0.97	0.99	40

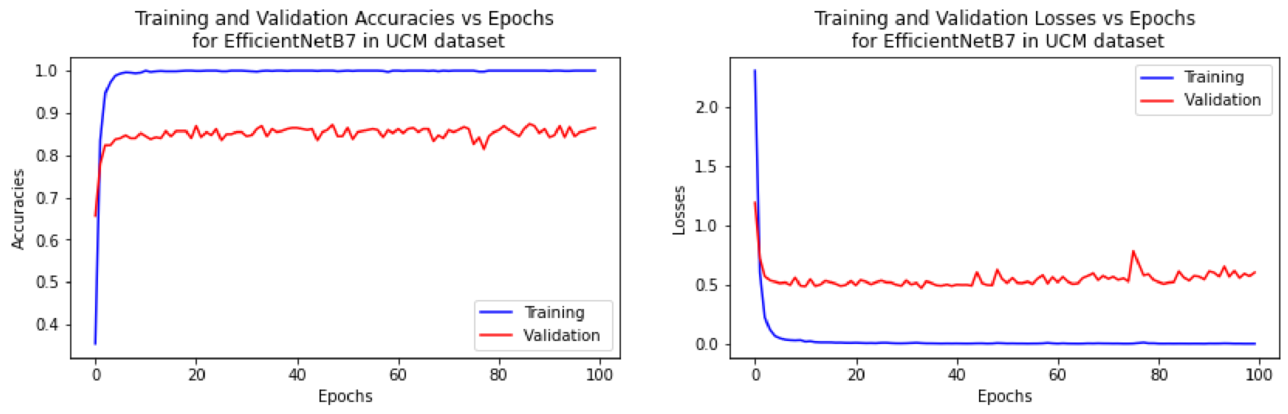
Table 7.10 MobileNet classification reports for the RSSCN7 dataset

Class name	Precision	Recall	F1-score	Support
Field	0.65	0.90	0.76	80
Forest	0.79	0.95	0.86	80
Grass	0.72	0.53	0.61	80
Industry	0.62	0.57	0.60	80
Parking	0.84	0.72	0.78	80
Resident	0.85	0.64	0.73	80
RiverLake	0.81	0.94	0.87	80

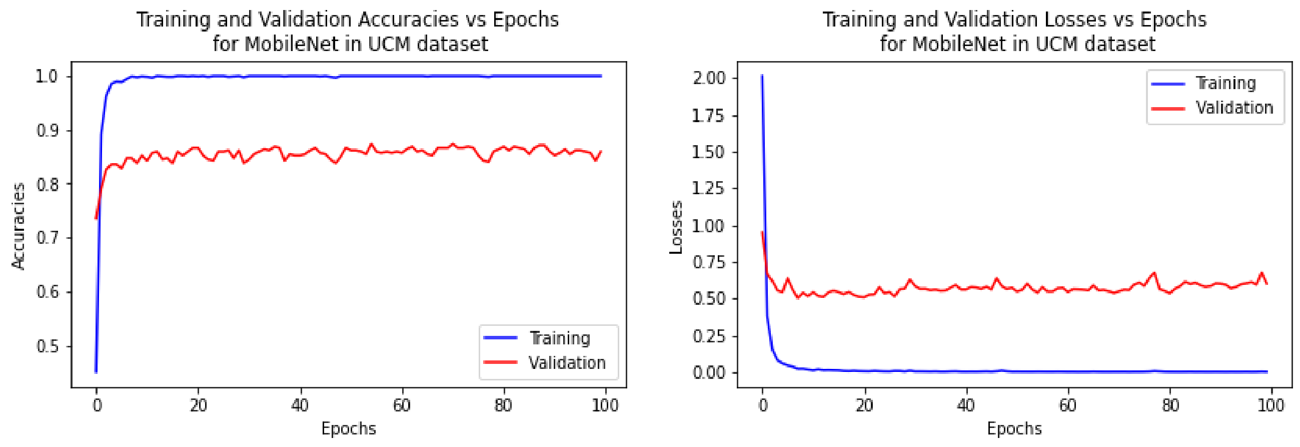
The training and validation accuracies and losses are also used for evaluating the models in the two datasets. The training accuracies (with a blue color curve) are smoothly increasing, while the validation accuracies (with a red color curve) are somewhat fluctuating in increasing the accuracies in all models and all datasets, as depicted in Figure 7.1 (**a to c left**), Figure 7.2 (**a to c left**), and Figure 7.3 (**a to c left**). We used the cross-entropy loss function to reduce errors in the model performance. The training losses (with a blue color curve) are smoothly decreasing, while the validation losses (with a red color curve) are somewhat fluctuating in reducing the errors in all models and all datasets, as depicted in Figure 7.1 (**a to c right**), Figure 7.2 (**a to c right**), and Figure 7.3 (**a to c left**).

In addition to deploying accuracy error, precision, recalls, and the F1-score, we used CM to evaluate class performances in each DL model and all datasets. The CM metric, like the F1-score, shows better class performance in most classes in the UCM dataset. The CM measures the class performance, whether it is classified correctly or incorrectly. CM considers each class label in rows (the "true labeled class") and columns (the "predicted labeled class"), as depicted in Figure 7.4 through Figure 7.12. The probability score in the diagonal intersection showed the correct classified class while the results in other rows-columns wise are predicted to be in misclassified classes.

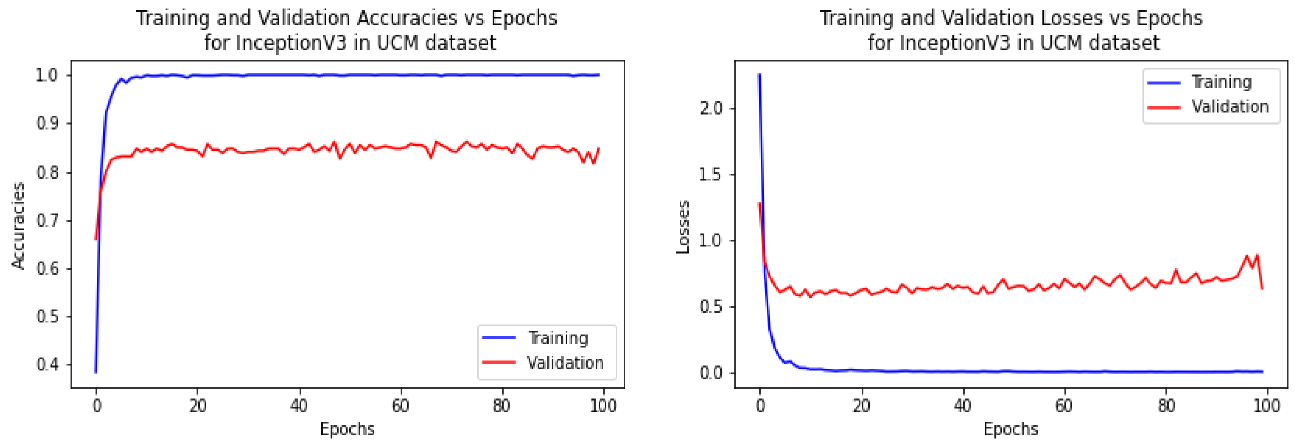
While evaluating the class performance in CM, the worst accuracy was scored in each model and dataset. For instance, the river (45%), agriculture (45%), and grass (53%) in EfficientNet7, as shown in Figure 7.4, Figure 7.5, and Figure 7.6.; the golf-course (45%), agriculture (45%) and park (45%), and industry (48%) in InceptionV3, as shown in Figure 7.7, Figure 7.8, and Figure 7.9, the mobile-home-park (55%), meadow (57%) and grass (53%) in MobileNet, shown in Figure 7.10, Figure 7.11, and Figure 7.12 for UCM, SIRI-WHU, and RSSCN7 datasets, respectively. In general, the class CM performance is better in UCM than SIRI-WHU datasets; for instance, the shared class agriculture has the best CM performance in UCM but the worst performance in SIRI-WHU and RSSCN7.



a) EfficientNetB7

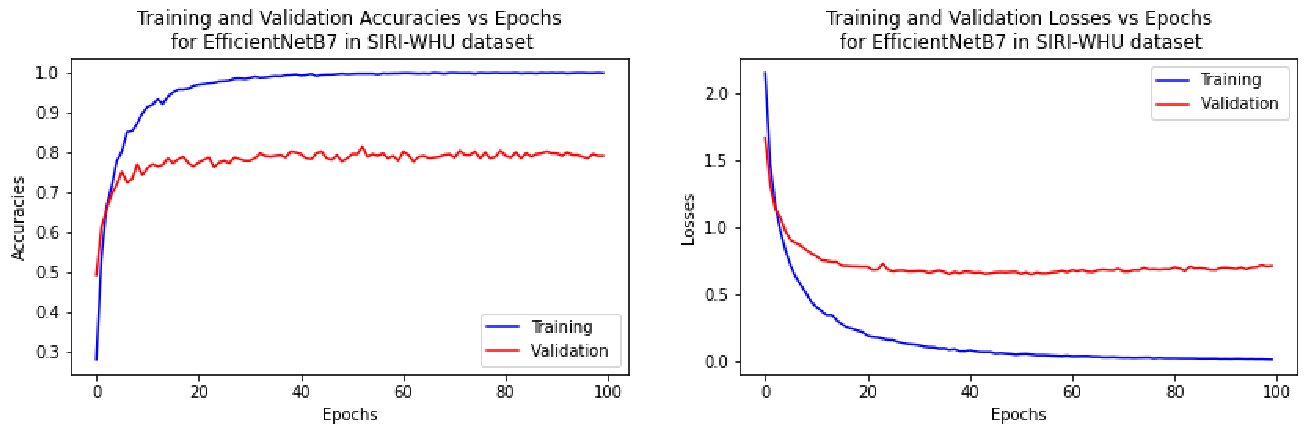


b) MobileNet

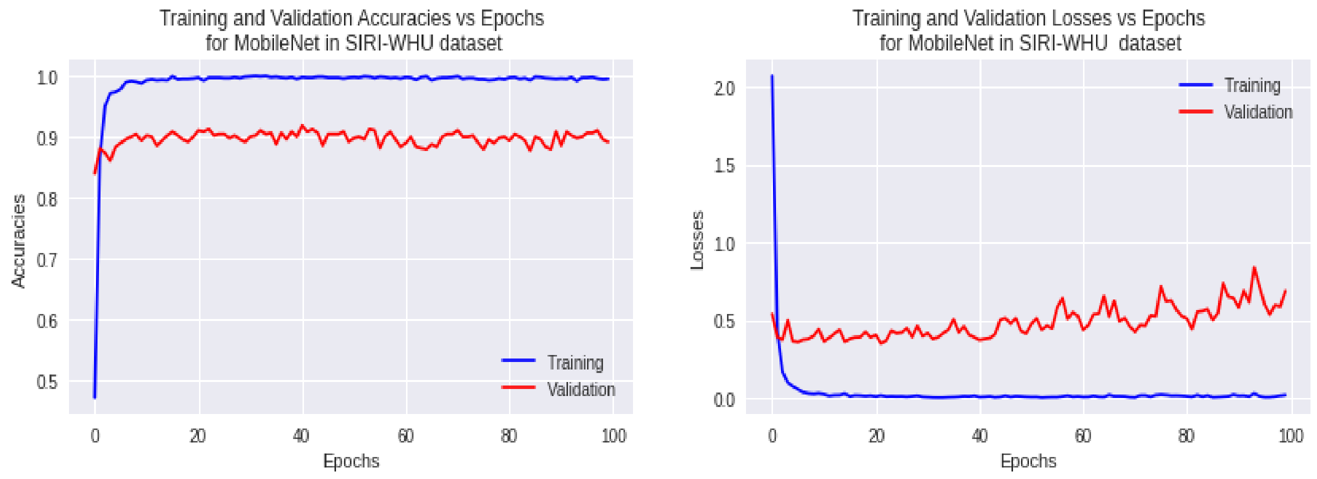


c) InceptionV3

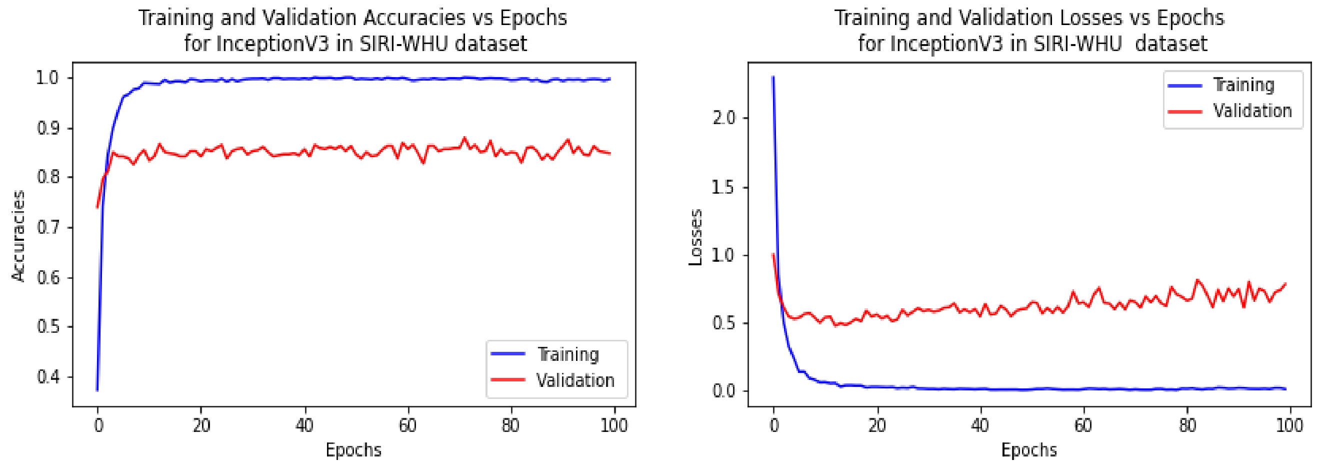
Figure 7.1. Training and Validation losses and accuracies of the three DL models (a, b, c) for the UCM dataset



a) EfficientNetB7

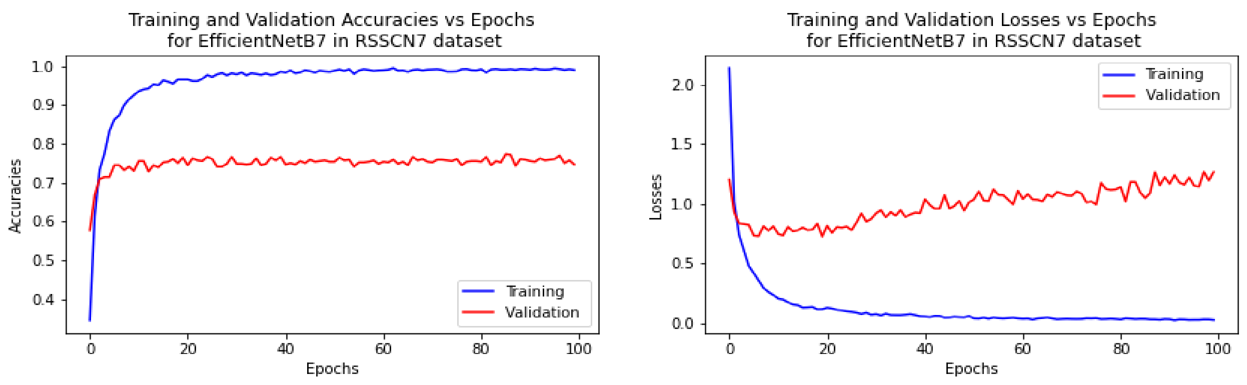


b) MobileNet

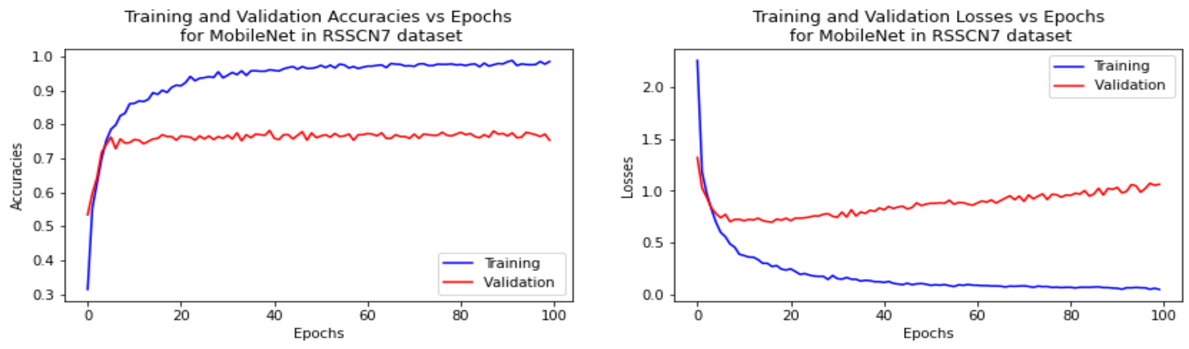


c) InceptionV3

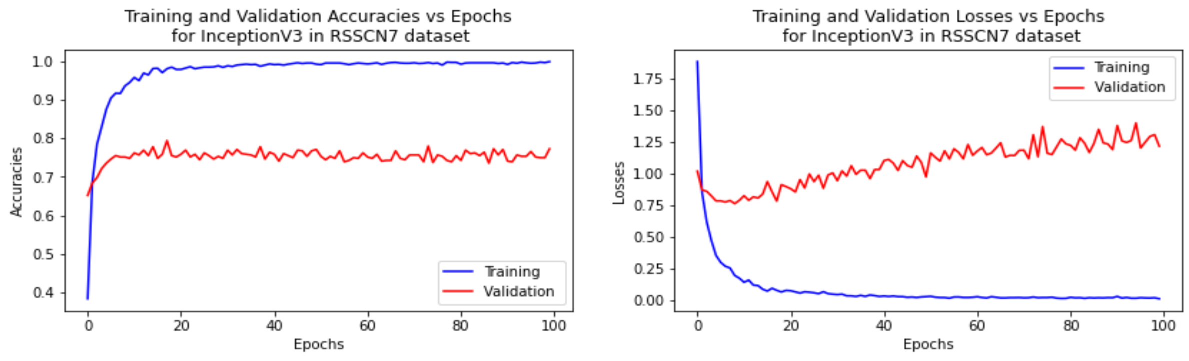
Figure 7.2.Training and Validation losses and accuracies of the three DL models (a, b, c) for the SIRI-WHU dataset



a) EfficientNetB7



b) MobileNet



c) InceptionV3

Figure 7.3. Training and Validation losses and accuracies of the three DL models (a, b, c) for the RSSC7 dataset

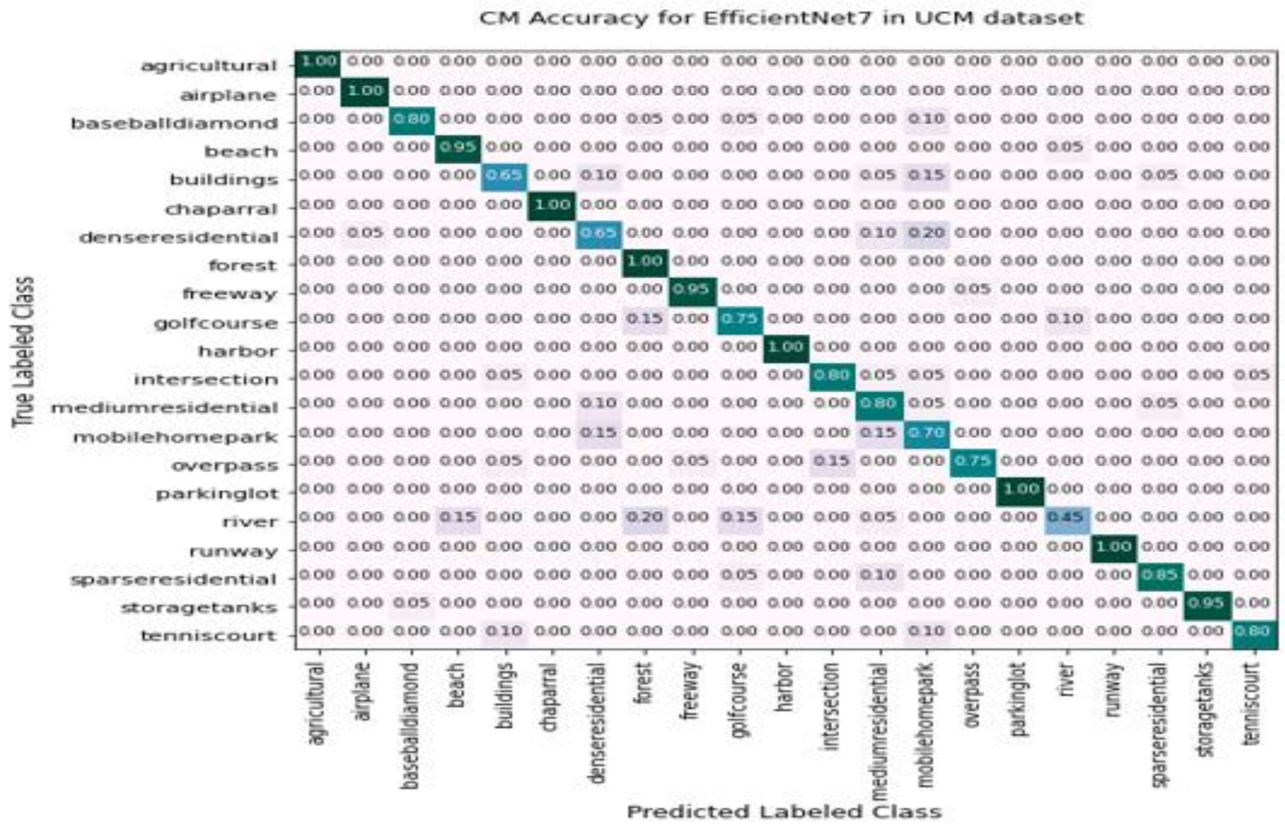


Figure 7.4. CM results for EfficientNet7 model in the UCM dataset

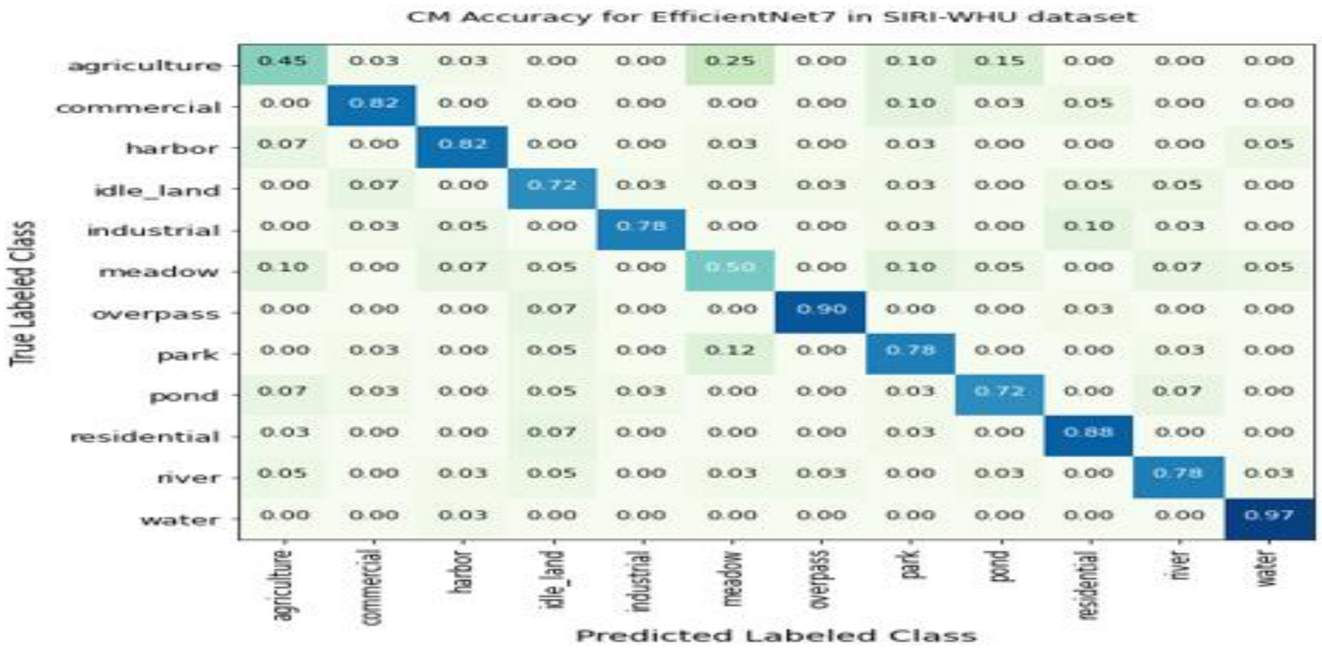


Figure 7.5. CM results for EfficientNet7 model in the SIRI-WHU dataset

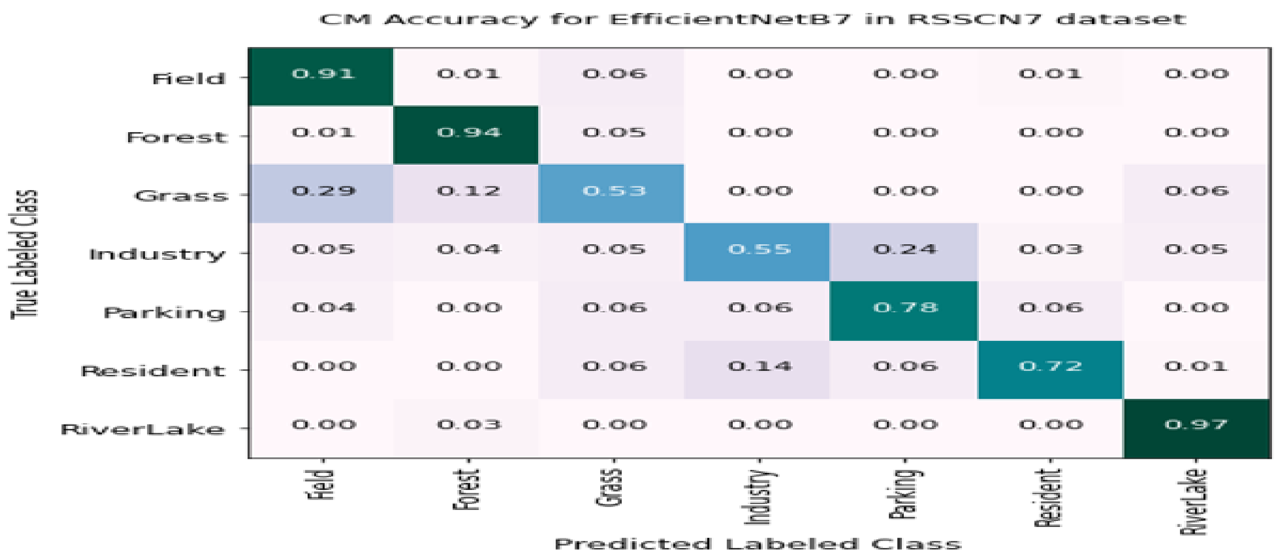


Figure 7.6. CM results for EfficientNet7 model in the RSSCN7 dataset

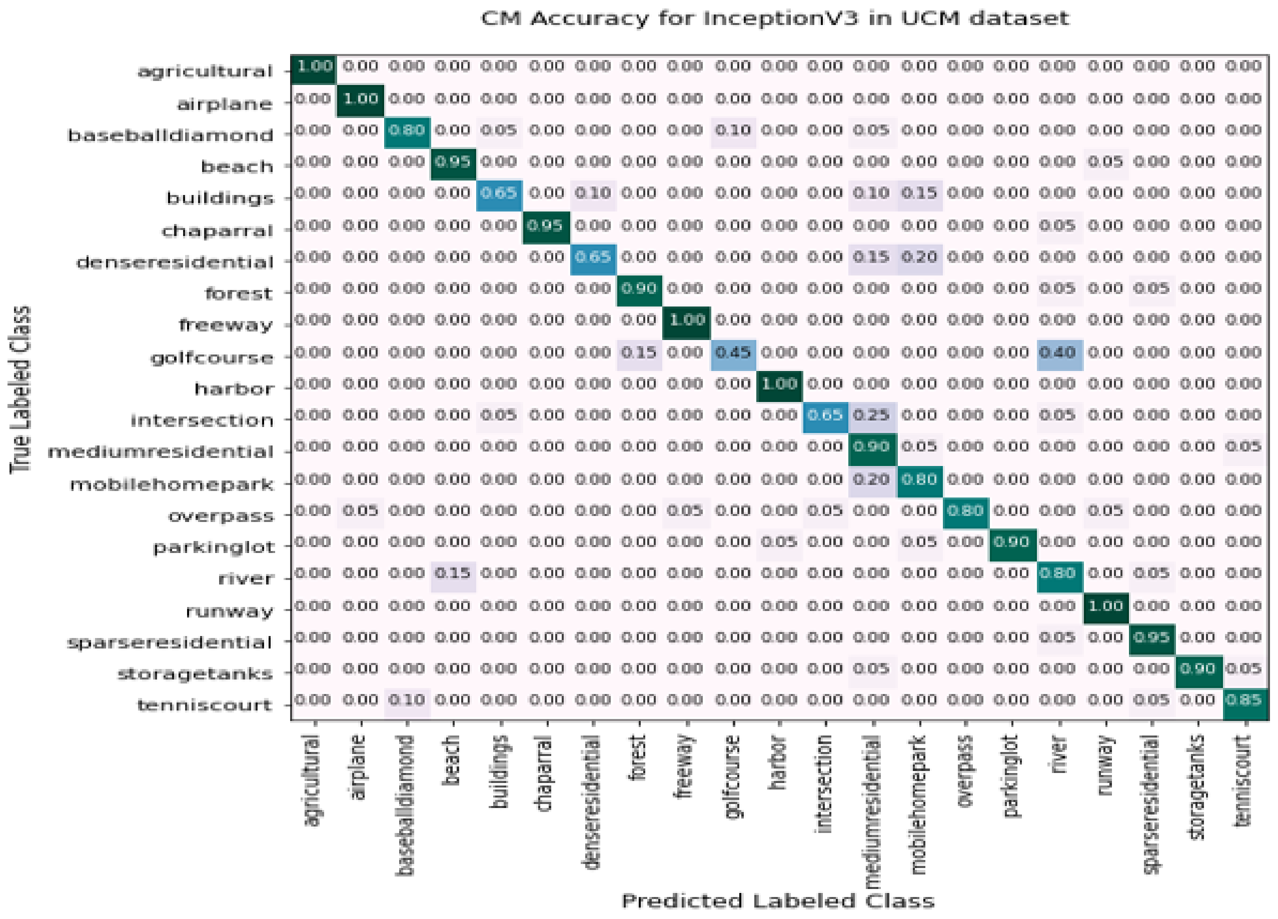


Figure 7.7. CM results for InceptionV3 model in the UCM dataset

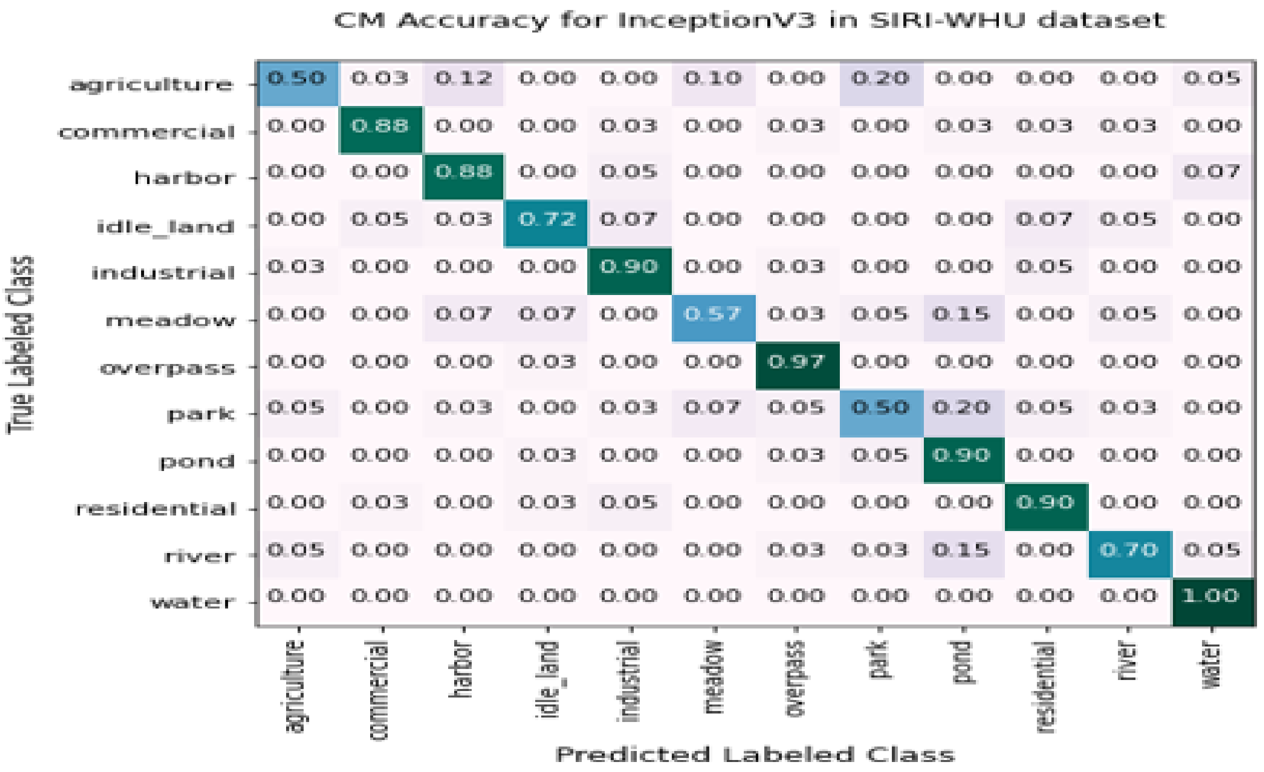


Figure 7.8. CM results for InceptionV3 model in the SIRI-WHU dataset

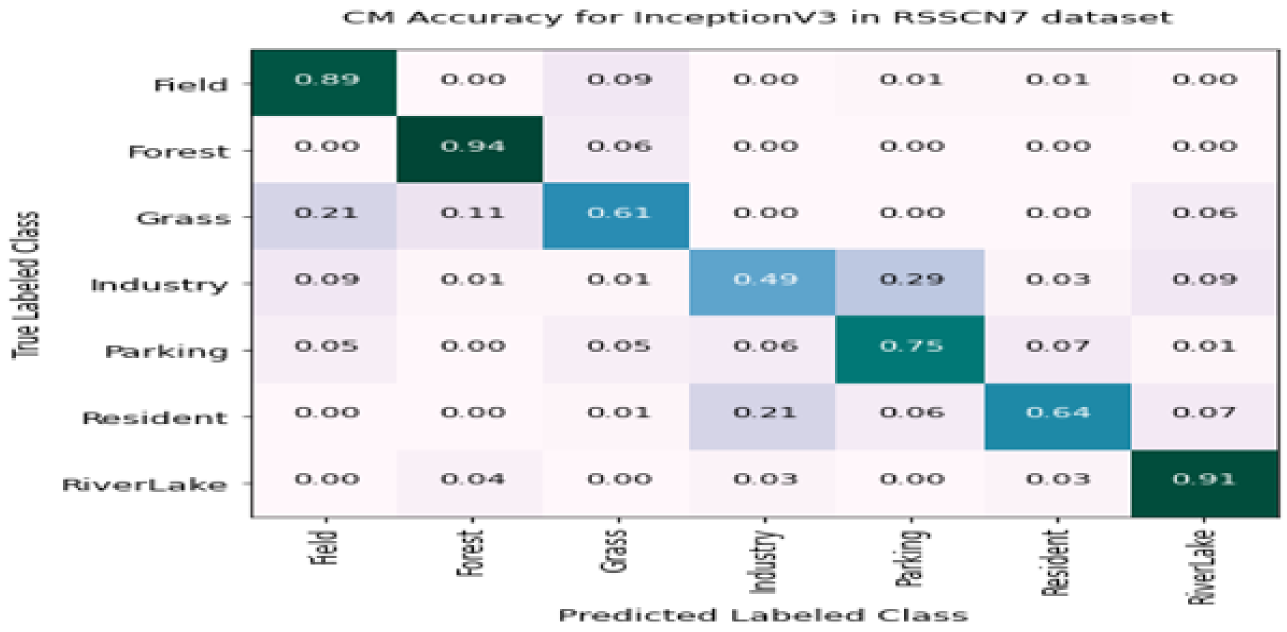


Figure 7.9. CM results for InceptionV3 model in the RSSCN7 dataset

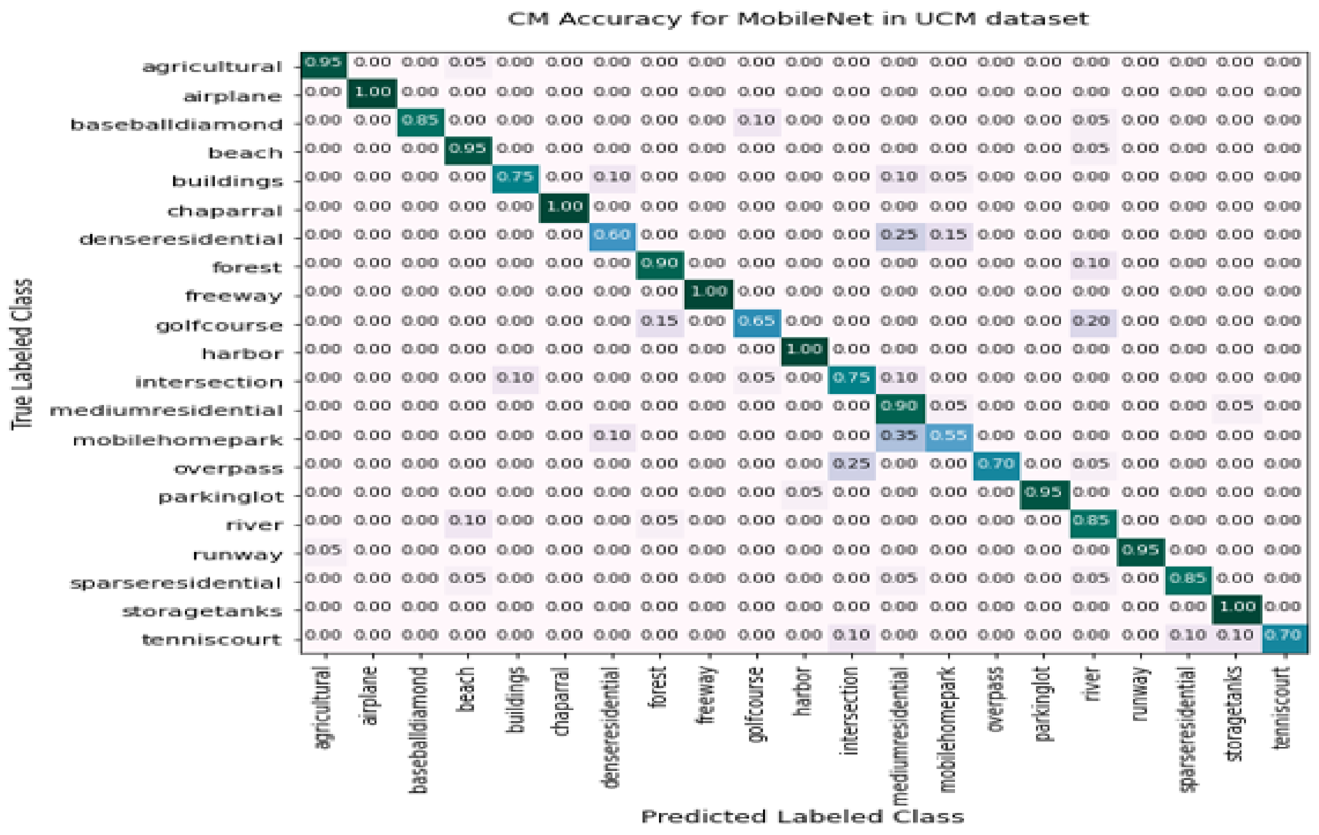


Figure 7.10. CM results for MobileNet model in the UCM

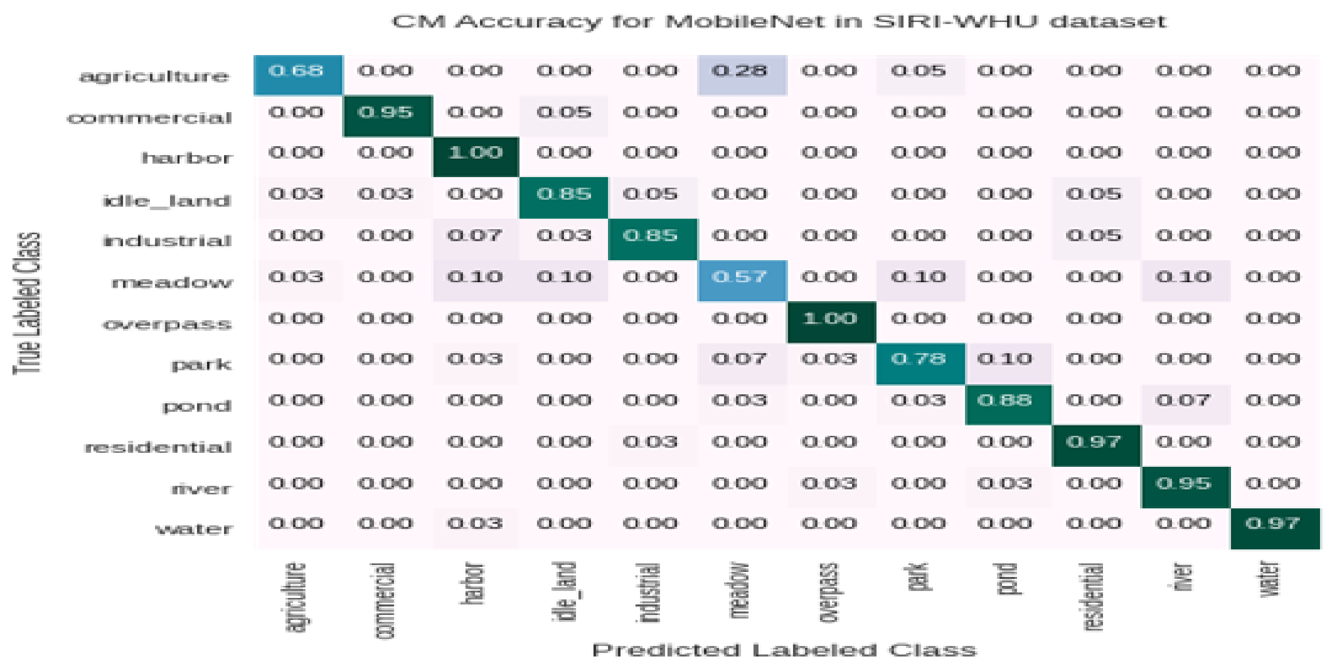
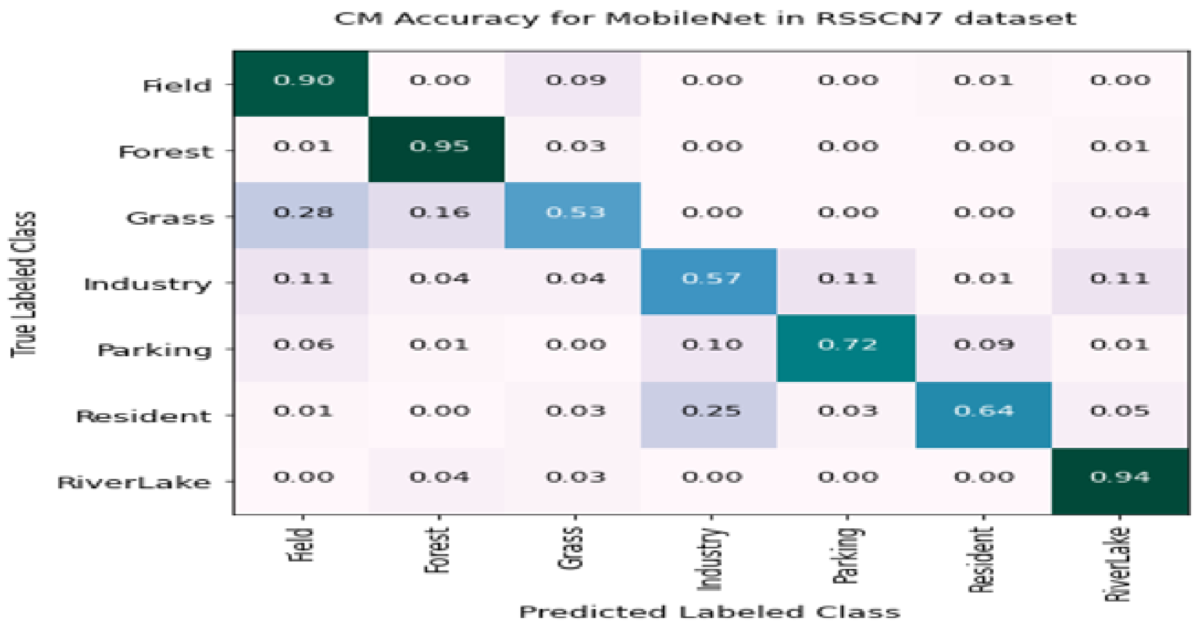


Figure 7.11. CM results for MobileNet model in the SIRI-WHU dataset



	Field	Forest	Grass	Industry	Parking	Resident	RiverLake
Field	0.90	0.00	0.09	0.00	0.00	0.01	0.00
Forest	0.01	0.95	0.03	0.00	0.00	0.00	0.01
Grass	0.28	0.16	0.53	0.00	0.00	0.00	0.04
Industry	0.11	0.04	0.04	0.57	0.11	0.01	0.11
Parking	0.06	0.01	0.00	0.10	0.72	0.09	0.01
Resident	0.01	0.00	0.03	0.25	0.03	0.64	0.05
RiverLake	0.00	0.04	0.03	0.00	0.00	0.00	0.94

Figure 7.12. CM results for MobileNet model in the RSSCN7 dataset

7.3.2 Discussions

This objective aimed the DL models for LCLU classification using RSHIs. Table 7.11 shows the performance of these models as a result of various measurement metrics, with good accuracy results for the classification problem.

To address our objective stated in this task, Table 7.2 through Table 7.10 and Figure 7.1 through Figure 7.12 compare the DL model performances on the UCM, SIRI-WHU, and RSSCN7 datasets. From the results, good class performance has been achieved in precision, recall, F1-score, and CM, though some class performances scored lower in values. The training accuracy increases smoothly in all DL models across all datasets, as shown in Figure 7.1, Figure 7.2 and Figure 7.3. The accuracy performance of each model has been evaluated with 420, 480, and 560 test images of the UCM, SIRI-WHU, and RSSCN7 datasets, respectively. The overall accuracy of each model is summarized in Table 7.11.

MobileNet outperforms the other two DL models, with accuracy of 86.47% and 87.07% for the UCM and SIRI-WHU datasets, respectively. However, the EfficientNetB7 model outperforms all other models in the RSSCN7 dataset, despite the fact that all models trained on the UCM and SIRI-WHU datasets outperformed those trained on the RSSCN7 dataset, as shown in Table 7.11.

Table 7.11. The DL model performances in performance measurement metrics in both datasets

Dataset	DL Models	DL Performances in performance measurement metrics			
		Precision	Recall	F1-score	Accuracy
UCM	EfficientNetB7	86.00	85.00	85.50	86.16
	InceptionV3	87.00	85.00	85.99	86.24
	MobileNet	87.00	85.00	85.99	**86.47**
SIRI-WHU	EfficientNetB7	76.00	76.00	76.00	76.04
	InceptionV3	79.00	79.00	79.00	79.54
	MobileNet	87.00	87.00	**87.00**	**87.08**
RSSCN7	EfficientNetB7	77.00	77.00	**77.00**	**77.14**
	InceptionV3	74.00	75.00	74.50	74.64
	MobileNet	76.00	75.00	75.50	75.00

7.4 Chapter Summarization

In this chapter, we applied the recent DL models to the UCM, SIRI-WHU, and RSSCN7 datasets, which have different properties, using DL hyperparameters for LCLU classification problems. Therefore, we designed the three DL CNN models, namely EfficientNetB7, InceptionV3, and MobileNet, for classifying the LCLU classification using the UCM, SIRI-WHU, and RSSCN7 datasets, which have different properties. Because of how well it worked, we added the EfficientNetB7 DL model to the LCLU classification.

The model performances were evaluated and compared using accuracy, precision, recall, f1-score, and CM metrics. The MobileNet model outperformed the other models in the UCM and SIRI-WHU datasets in terms of accuracy. The nature of the dataset had an effect on the DL performances in the majority of the measurement metrics we used. The better performance results in most metrics have been achieved in the UCM dataset rather than the SIRI-WHU and RSSCN7 datasets.

8. CONCLUSIONS AND RECOMMENDATIONS

8.1 Conclusions

In this research, we present the results of our investigation into the use of DL models for solving the LCLU classification problem in RS imagery datasets. We reviewed and analyzed different primary studies that had been retrieved from reputable databases in order to pinpoint the research gaps. DL methods for LCLU classification using RS are recent hot research areas in the field of ML and AI. As can be seen in Figure 2.4 and Figure 2.5, however, there has not been much research into the use of AI, namely DL approaches, for LCLU classification using RS images. Results from our review indicate that DL approaches are gaining popularity due to their potential to enhance classification system performance. The performance of DL modes has been studied by a number of researchers, such as [50] and [21], because of their advantages over LCLU classification utilizing RS images. So, we achieved our objective of reviewing by citing some of the researchers whose work we looked at and by pointing out where the literature was lacking and what these experts suggested for the future (see Table 2.2).

After completing the review objective, we set four main experimental objectives to address the problem of LCLU classification in RS images. The first experimental task was designing an end-to-end CNN-FE model. After developing this model on the inconsistent UCM dataset, we retrained it on the more stable SIRI-WHU dataset of RS images to see if the dataset had any effect on its performance. To further verify the prospective use of the CNN-FE, we also developed a VGG19 pretrained DL model and tested its performance on both datasets. In this objective, we validated that CNNs are powerful DL techniques for evaluating RS images for LCLU classification systems. Due to the inconsistency of RS images, the results of any modeling done with pretrained networks will likely not be accurate. We evaluate its performance versus that of the VGG19 pretrained model, which was also trained with similar hyperparameters, and prior state-of-the-art studies. Our results, which include how the features of the dataset affect the model and how it could be used in different domains, show that the CNN-FE outperformed the previous works and the VGG-19 pretrained model.

The second experimental task was to build a TL model with the help of pretrained networks and bottleneck feature extraction. Our objective was to reduce training time for LCLU classification in RS images using the TL model. According to the state-of-the-art study by [58], [146], TL can be trained more quickly than other deep CNN models (trained in seconds compared to days when they were trained from scratch). To enhance the speed and accuracy of the model's training, we applied the bottleneck feature extraction technique to extract features from previously trained models. The model's performance is also prominent in all models, i.e., 92.46%, 94.36%, and 99.64% of the accuracy results for Resnet50V2, InceptionV3, and VGG19, respectively. As a result, the improved performance of the TL models for LCLU classification in RS images was our finding. The VGG-19 model, on the other hand, achieves high accuracy in a short period of time.

The third experimental task was to design and compare the CNN-FE, TL, and fine-tuning models for LCLU classification problems using RS images. The TL and fine-tuning models have been trained on the recent EfficientNetB7 pretrained baseline network, whereas the CNN-FE has been trained on the four blocks of CNNs using the UCM dataset. Here, the design of CNN-FE differs from our first experimental task designation by differentiating the batch size and learning rate values. The models are evaluated, and the fine-tuned model produces a higher accuracy performance in less time. In addition to the performance, the training time is another critical evaluation metric that is used to compare the economic advantages of TL and fine-tuning models over the DL models developed from scratch. We found that the TL and fine-turning DL models are good at saving time and important for improving performance.

The fourth experimental task was to apply the recent DL models to the UCM, SIRI-WHU, and RSSCN7 RS imagery datasets with different properties using DL hyperparameters for LCLU classification problems. Therefore, we designed the three DL CNN models, namely EfficientNetB7, InceptionV3, and MobileNet, for classifying the LCLU categories using RSHIs in the UCM and SIRI-WHU datasets. We contributed the EfficientNetB7 DL model to LCLU classification in RSHIs because of its good performance. The three models were evaluated and compared, and profound accuracy and f1-score measure performances have been achieved in MobileNet rather than in other models in the UCM and SIRI-WHU datasets. In the RSSCN7 dataset, EfficientNetB7 has higher accuracy and f1-score measure performance. The nature of the dataset had an effect on the DL performances in the majority of the measurement metrics we used. The better performance results in most metrics have been achieved in the UCM dataset rather than the SIRI-WHU and RSSCN7 datasets.

Our contributions in this study include exploring the potential application of the recent DL models for LCLU classification problem using RS imagery data for the benefits of the RS communities and decision makers, improving the performances of the earlier studies, identifying the better performance of the models in terms of training times, dataset size and dataset properties, identifying the DL hyperparameters and properties of the dataset influence the DL performance in LCLU classification problem, and suggesting the future research direction to the future researchers.

This study's significance values in its application of an intelligent LCLU classification system to the problem of scene identification for the RS communities. Sustainable development in the areas of agricultural and urban planning, environmental protection, and natural resource management would all greatly benefit from the use of the LCLU classification system. We also looked at how the DL models worked and improved their performance, showing that they could be used to classify LCLUs in RS images.

However, the lack of powerful computational resources i.e., robust processor needs and time constraints resulting from the COVID-19 epidemic were the major challenges or limitations we faced during our study. Since we were working with a CPU processor and Google Colab instead of a GPU processor, we had to work with relatively small RS datasets. In addition, both the TL and fine-tuning modes work with small datasets. Developing DL models from scratch is time-consuming and difficult, so the size of the dataset may have an effect on the DL performance. We didn't have enough time to compare typical ML methods with DL methods or test the effects of important DL hyperparameters that can't be learned, like the deeper number of layers, iteration, and batch normalization (mean and variance).

8.2 Recommendations

The research objectives of this thesis have been achieved with constraints and challenges. Therefore, we would like to suggest the following major recommendations for further investigations in the area based on the aforementioned challenges and constraints:

- The hardware processor requirements, such as the CPU, GPU, and VPU, are essential requirements for designing DL models. Because of its extremely fast computation and processing capabilities on large datasets, the GPU is required to improve model performance. VPU in the other way is also required to accelerate the performance of DL models and produce high-quality images with less power consumption by freezing up the CPU and GPU space. So, it might be better to run the experiments again with a powerful GPU or VPU to improve how well the DL models work on larger RSHIs datasets.
- The various dataset properties and the DL hyperparameters could also affect the model's performance. So, the next step to improving DL performance in the domain would be to look into DL optimization techniques for LCLU classification using different datasets.
- The TL and pretrained models are effective in terms of time and resources. But training the TL and pretrained models on the pretrained models may be hard because pretrained networks like "ImageNet" may have been trained on large images with different properties than the RS images. Therefore, we recommend that designing the DL model from scratch is better for validating the LCLU classification system. Moreover, TL is recommended for small datasets, which may be less than thousands of images per class. If the dataset exceeds thousands of images per class, we would like to recommend designing the DL model from scratch.
- Moreover, the developed DL models need more improvements, validations, and comparisons with other traditional ML approaches on large datasets. Therefore, building DL and traditional ML approaches on other large datasets is our future task to compare their performances.

REFERENCES

[1] H. Yang, F., Rottensteiner, “Classification of Land Cover and Land Use based on Convolutional Neural Networks,” *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.,* vol. IV, no. 3, pp. 251–258, 2018, doi: https://doi.org/10.5194/isprs-annals.

[2] Y. Li, H. Zhang, X. Xue, Y. Jiang, and Q. Shen, “Deep learning for remote sensing image classification : A survey,” *WIREs Data Min. Knowl. Discov.*, vol. 8, no. 6, p. e1264, 2018, doi: 10.1002/widm.1264.

[3] R. Stivaktakis, G. Tsagkatakis, and P. Tsakalides, “Deep Learning for Multilabel Land Cover Scene Categorization Using Data Augmentation,” *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 7, pp. 1031–1035, 2019, doi: 10.1109/LGRS.2019.2893306.

[4] M. Carranza-García, J. García-Gutiérrez, and J. C. Riquelme, “A Framework for Evaluating Land Use and Land Cover Classification using Convolutional Neural Networks,” *Remote Sens.*, vol. 11, no. 3, p. 274, 2019, doi: 10.3390/rs11030274.

[5] D. Marmanis, M. Datcu, T. Esch, and U. Stilla, “Deep Learning Earth Observation Classification Using ImageNet Pretrained Networks,” *IEEE Geosci. Remote Sens.*, vol. 13, no. 1, pp. 105–109, 2016, doi: 10.1109/LGRS.2015.2499239.

[6] K. Nogueira, O. A. B. Penatti, and J. A. dos Santos, “Towards Better Exploiting Convolutional Neural Networks for Remote Sensing Scene Classification,” *Pattern Recognit.*, vol. 61, pp. 539–556, 2016, doi: 10.1016/j.patcog.2016.07.001.

[7] M. A. Shafaey, M. A. Salem, H. M. Ebied, M. N. Al-Berry, and M. F. Tolba, “Deep Learning for Satellite Image Classification,” in *Hassanien, A., Tolba, M., Shaalan, K., Azar, A. (eds) Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2018. AISI 2018. Advances in Intelligent Systems and Computing,* 2019, vol. 845, pp. 383–391. doi: 10.1007/978-3-319-99010-1_35.

[8] G. J. Scott, M. R. England, W. A. Starms, R. A. Marcum, and C. H. Davis, “Training Deep Convolutional Neural Networks for Land-Cover Classification of High-Resolution Imagery,” *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 4, pp. 549–553, 2017, doi: 10.1109/LGRS.2017.2657778.

[9] S. Dong, Y. Zhuang, Z. Yang, L. Pang, H. Chen, and T. Long, “Land Cover Classification From VHR Optical Remote Sensing Images by Feature Ensemble Deep Learning

Network,” *IEEE Geosci. Remote Sens. Lett.*, vol. 17, no. 8, pp. 1396–1400, 2020, doi: 10.1109/LGRS.2019.2947022.

[10] K. Karalas, G. Tsagkatakis, M. Zervakis, and P. Tsakalides, “Land Classification Using Remotely Sensed Data: Going Multilabel,” *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 6, pp. 3548–3563, 2016, doi: 10.1109/TGRS.2016.2520203.

[11] B. Zhao, B. Huang, and Y. Zhong, “Transfer Learning With Fully Pretrained Deep Convolution Networks for Land-Use Classification,” *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 9, pp. 1436–1440, 2017, doi: 10.1109/LGRS.2017.2691013.

[12] Z. Huang, C. O. Dumitru, Z. Pan, B. Lei, and M. Datcu, “Classification of Large-Scale High-Resolution SAR Images with Deep Transfer Learning,” *IEEE Geosci. Remote Sens. Lett.*, vol. 18, no. 1, pp. 107–111, 2021, doi: 10.1109/LGRS.2020.2965558.

[13] Y. Xu *et al.*, “Advanced Multi-Sensor Optical Remote Sensing for Urban Land Use and Land Cover Classification : Outcome of the 2018 IEEE GRSS Data,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 12, no. 6, pp. 1709–1724, 2019, doi: 10.1109/JSTARS.2019.2911113.

[14] T. W. Cenggoro, S. M. Isa, G. P. Kusuma, and B. Pardamean, “Classification of Imbalanced Land-Use/Land-Cover Data using Variational Semi-supervised Learning,” in *2017 International Conference on Innovative and Creative Information Technology (ICITech)*, 2017, pp. 1–6. doi: 10.1109/INNOCIT.2017.8319149.

[15] M. Kampffmeyer, A. Salberg, and R. Jenssen, “Urban Land Cover Classification With Missing Data Modalities Using Deep Convolutional,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 11, no. 6, pp. 1758–1768, 2018, doi: 10.1109/JSTARS.2018.2834961.

[16] A. Hamida, A. Benoit, P. Lambert, and C. Amar, “3-D Deep Learning Approach for Remote Sensing Image Classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 8, pp. 4420–4434, 2018, doi: 10.1109/TGRS.2018.2818945.

[17] A. Elshamli, G. W. Taylor, A. Berg, and S. Areibi, “Domain Adaptation Using Representation Learning for the Classification of Remote Sensing Images,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 10, no. 9, pp. 4198–4209, 2017, doi: 10.1109/JSTARS.2017.2711360.

[18] X. Liu *et al.*, “Classifying urban land use by integrating remote sensing and social media data,” *Int. J. Geogr. Inf. Sci.*, vol. 31, no. 8, pp. 1675–1696, 2017, doi:

10.1080/13658816.2017.1324976.

[19] S. Song, H. Yu, Z. Miao, Q. Zhang, Y. Lin, and S. Wang, “Domain Adaptation for Convolutional Neural Networks-Based Remote Sensing,” *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 8, pp. 1324–1328, 2019, doi: 10.1109/LGRS.2019.2896411.

[20] P. Russell, S., & Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Upper Saddle River, New Jersey: Pearson Education, Inc., 2003.

[21] A. M. Abdi, “Land cover and land use classification performance of machine learning algorithms in a boreal landscape using Sentinel-2 data,” *GIScience Remote Sens.*, vol. 57, no. 1, pp. 1–20, 2019, doi: 10.1080/15481603.2019.1650447.

[22] X. X. Zhu *et al.*, “Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources,” *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 4, pp. 8–36, 2017, doi: 10.1109/MGRS.2017.2762307.

[23] K. Ali *et al.*, “Land Usage Analysis: A Machine Learning Approach,” *Int. J. Comput. Appl.*, vol. 141, no. 12, pp. 23–28, 2016, doi: 10.5120/ijca2016909936.

[24] R. P. de Lima and K. Marfurt, “Convolutional Neural Network for Remote - Sensing Scene Classification : Transfer Learning Analysis,” *Remote Sens.*, vol. 12, no. 1, p. 86, 2020, doi: 10.3390/rs12010086.

[25] X. Sun, L. A. N. Liu, C. Li, J. Yin, J. Zhao, and W. E. N. Si, “Classification for Remote Sensing Data With Improved CNN-SVM Method,” *IEEE Access*, vol. 7, pp. 164507–164516, 2019, doi: 10.1109/ACCESS.2019.2952946.

[26] N. Kussul, M. Lavreniuk, S. Skakun, and A. Shelestov, “Deep Learning Classification of Land Cover and Crop Types Using Remote Sensing Data,” *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 5, pp. 778–782, 2017.

[27] E. Ndikumana, D. H.T.Minh, N. Baghdadi, D. Courault, and L. Hossard, “Deep Recurrent Neural Network for Agricultural Classification using multitemporal SAR Sentinel-1 for Camargue , France,” *Remote Sens.*, vol. 10, no. 8, p. 1217, 2018, doi: 10.3390/rs10081217.

[28] M. E. Yuksel, N. S. Basturk, H. Badem, A. Caliskan, and A. Basturk, “Classification of high resolution hyperspectral remote sensing data using deep neural networks,” *J. Intell. Fuzzy Syst.*, vol. 34, no. 4, pp. 2273–2285, 2018, doi: 10.3233/JIFS-171307.

[29] X. X. Zhu *et al.*, “Deep Learning in Remote Sensing: A Review,” *arXiv:1710.03959*,

2017, doi: 10.1109/MGRS.2017.2762307.

[30] Y. Zhong, F. Fei, Y. Liu, B. Zhao, H. Jiao, and L. Zhang, “SatCNN : satellite image dataset classification using agile convolutional neural networks convolutional neural networks,” *Remote Sens. Lett.*, vol. 8, no. 2, pp. 136–145, 2017, doi: 10.1080/2150704X.2016.1235299.

[31] F. P. S. Luus, B. P. Salmon, F. Van Den Bergh, and B. T. J. Maharaj, “Multiview Deep Learning for Land-Use Classification,” *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 12, pp. 2448–2452, 2015, doi: 10.1109/LGRS.2015.2483680.

[32] W. Li, H. Liu, Y. U. Wang, Z. Li, Y. Jia, and G. Gui, “Deep Learning-Based Classification Methods for Remote Sensing Images in Urban Built-up Areas,” *IEEE Access*, vol. XX, no. c, pp. 1–9, 2018, doi: 10.1109/ACCESS.2019.2903127.

[33] J. M. Haut, R. Fernandez-Beltran, M. E. Paoletti, J. Plaza, and A. Plaza, “Remote Sensing Image Superresolution Using Deep Residual Channel Attention,” *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 11, pp. 9277–9289, 2019, doi: 10.1109/TGRS.2019.2924818.

[34] G. Chen, X. Zhang, Q. Wang, F. Dai, Y. Gong, and K. Zhu, “Symmetrical Dense-Shortcut Deep Fully Convolutional Networks for Semantic Segmentation of Very-High-Resolution Remote Sensing Images,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 11, no. 5, pp. 1633–1644, 2018, doi: 10.1109/JSTARS.2018.2810320.

[35] P. Helber, B. Bischke, A. Dengel, and D. Borth, “Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 12, no. 7, pp. 2217–2226, 2019, doi: 10.1109/JSTARS.2019.2918242.

[36] Q. Weng, Z. Mao, J. Lin, and W. Guo, “Land-Use Classification via Extreme Learning Classifier Based on Deep Convolutional Features,” *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 5, pp. 704–708, 2017, doi: 10.1109/LGRS.2017.2672643.

[37] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, “Convolutional Neural Networks for Large-Scale Remote-Sensing Image Classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 2, pp. 645–657, 2017, doi: 10.1109/TGRS.2016.2612821.

[38] D. Ienco, R. Gaetano, C. Dupaquier, and P. Maurel, “Land Cover Classification via Multitemporal Spatial Data by Deep Recurrent Neural Networks,” *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 10, pp. 1685–1689, 2017.

[39] J. Dong, R. Yin, X. Sun, Q. Li, Y. Yang, and X. Qin, “Inpainting of Remote Sensing SST Images with Deep Convolutional Generative Adversarial Network,” *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 2, pp. 173–177, 2019, doi: 10.1109/LGRS.2018.2870880.

[40] Y. Zhan, D. Hu, Y. Wang, and X. Yu, “Semisupervised Hyperspectral Image Classification Based on Generative Adversarial Networks,” *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 2, pp. 212–216, 2018, doi: 10.1109/LGRS.2017.2780890.

[41] Q. Zou, L. Ni, T. Zhang, and Q. Wang, “Deep Learning Based Feature Selection for Remote Sensing Scene Classification,” *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 11, pp. 2321–2325, 2015, doi: 10.1109/LGRS.2015.2475299.

[42] Y. Chen, X. Zhao, and X. Jia, “Spectral – Spatial Classification of Hyperspectral Data Based on Deep Belief Network,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 8, no. 6, pp. 2381–2392, 2015.

[43] M. Rezaee, M. Mahdianpari, Y. Zhang, and B. Salehi, “Deep Convolutional Neural Network for Complex Wetland Classification Using Optical Remote Sensing Imagery,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 11, no. 9, pp. 3030–3039, 2018, doi: 10.1109/JSTARS.2018.2846178.

[44] G. Fu, C. Liu, R. Zhou, T. Sun, and Q. Zhang, “Classification for high resolution remote sensing imagery using a fully convolutional network,” *Remote Sens.*, vol. 9, no. 5, pp. 1–21, 2017, doi: 10.3390/rs9050498.

[45] K. Bhosle and V. Musande, “Evaluation of Deep Learning CNN Model for Land Use Land Cover Classification and Crop Identification Using Hyperspectral Remote Sensing Images,” *J. Indian Soc. Remote Sens.*, vol. 47, no. 11, pp. 1949–1958, 2019, doi: 10.1007/s12524-019-01041-2.

[46] J. Song, S. Gao, Y. Zhu, and C. Ma, “A survey of remote sensing image classification based on CNNs,” *Big Earth Data*, vol. 3, no. 3, pp. 232–254, 2019, doi: 10.1080/20964471.2019.1657720.

[47] M. Castelluccio, G. Poggi, C. Sansone, and L. Verdoliva, “Land Use Classification in Remote Sensing Images by Convolutional Neural Networks,” *ArXiv Prepr.*, no. arXiv: 1508.00092v1, 2015.

[48] X. Liu, Y. Zhou, J. Zhao, R. Yao, B. Liu, and Y. Zheng, “Siamese Convolutional Neural Networks for Remote Sensing Scene Classification,” *IEEE Geosci. Remote Sens. Lett.*,

vol. 16, no. 8, pp. 1200–1204, 2019, doi: 10.1109/LGRS.2019.2894399.

[49] G. Cheng, Z. Li, X. Yao, L. Guo, and Z. Wei, "Remote Sensing Image Scene Classification Using Bag of Convolutional Features," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 10, pp. 1735–1739, 2017, doi: 10.1109/LGRS.2017.2731997.

[50] G. Li, C. Zhang, M. Wang, F. Gao, and X. Zhang, "Scene classification of high-resolution remote sensing image using transfer learning with multi-model feature extraction framework," *Commun. Comput. Inf. Sci.*, vol. 875, pp. 238–251, 2018, doi: 10.1007/978-981-13-1702-6_24.

[51] M. Papadomanolaki, M. Vakalopoulou, S. Zagoruyko, and K. Karantzalos, "Benchmarking Deep Learning Frameworks for the Classification of Very High Resolution Satellite Multispectral Data," *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. 3, no. 7, pp. 83–88, 2016, doi: 10.5194/isprsannals-iii-7-83-2016.

[52] M. A. Kadhim and M. H. Abed, "Convolutional Neural Network for Satellite Image Classification," in *Studies in Computational Intelligence*, vol. 830, Springer, Cham: Springer International Publishing, 2019, pp. 165–178. doi: 10.1007/978-3-030-14132-5.

[53] F. Özyurt, "Efficient Deep Feature Selection for Remote Sensing Image Recognition with Fused Deep Learning Architectures," *J. Supercomput.*, vol. 76, no. 11, pp. 8413–8431, 2019, doi: 10.1007/s11227-019-03106-y.

[54] N. Imamoglu, P. Martínez-Gómez, R. Hamaguchi, K. Sakurada, and R. Nakamura, "Exploring recurrent and feedback CNNs for multi-spectral satellite image classification," *Procedia Comput. Sci.*, vol. 140, no. 2018, pp. 162–169, 2018, doi: 10.1016/j.procs.2018.10.325.

[55] L. Mou and X. Xiang, "RiFCN : Recurrent Network in Fully Convolutional Network for Semantic Segmentation of High Resolution Remote Sensing Images," *ArXiv*, vol. abs/1805.0, no. 1805.02091v1, pp. 1–29, 2018.

[56] A. Das, R. Giri, G. Chourasia, and A. A. Bala, "Classification of Retinal Diseases Using Transfer Learning Approach," in *Proceedings of the 4th International Conference on Communication and Electronics Systems, ICCES 2019*, 2019, no. Icces, pp. 2080–2084. doi: 10.1109/ICCES45898.2019.9002415.

[57] K. Weiss, T. M. Khoshgoftaar, and D. D. Wang, *A survey of transfer learning*, vol. 3, no. 1. Springer International Publishing, 2016. doi: 10.1186/s40537-016-0043-6.

[58] M. Mahdianpari, B. Salehi, M. Rezaee, F. Mohammadimanesh, and Y. Zhang, “Very Deep Convolutional Neural Networks for Complex Land Cover Mapping using Multispectral Remote Sensing Imagery,” *Remote Sens.*, vol. 10, no. 7, p. 1119, 2018, doi: 10.3390/rs10071119.

[59] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010, doi: 10.1109/TKDE.2009.191.

[60] R. P. de Lima and K. Marfurt, “Convolutional neural network for remote-sensing scene classification: Transfer learning analysis,” *Remote Sens.*, vol. 12, no. 1, p. 86, 2020, doi: 10.3390/rs12010086.

[61] H. Wu, B. Liu, W. Su, W. Zhang, and J. Sun, “Deep Filter Banks for Land-Use Scene Classification,” *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 12, pp. 1895–1899, 2016, doi: 10.1109/LGRS.2016.2616440.

[62] Y. Liu and Y. Zhong, “Scene Classification Based on a Deep Random-Scale Stretched Convolutional Neural Network,” *Remote Sens.*, vol. 10, no. 3, p. 444, 2018, doi: 10.3390/rs10030444.

[63] F. Zhang, B. Du, and L. Zhang, “Scene Classification via a Gradient Boosting Random Convolutional Network Framework,” *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 3, pp. 1793–1802, 2016, doi: 10.1109/TGRS.2015.2488681.

[64] X. Yu, X. Wu, C. Luo, and P. Ren, “Deep learning in remote sensing scene classification : a data augmentation enhanced convolutional neural network framework,” *GIScience Remote Sens.*, vol. 54, no. 5, pp. 741–758, 2017, doi: 10.1080/15481603.2017.1323377.

[65] Y. Liu, L. Gross, Z. Li, X. Li, X. Fan, and W. Qi, “Automatic Building Extraction on High-Resolution Remote Sensing Imagery Using Deep Convolutional Encoder-Decoder with Spatial Pyramid Pooling,” *IEEE Access*, vol. 7, pp. 128774–128786, 2019, doi: 10.1109/ACCESS.2019.2940527.

[66] B. Liu, X. Yu, P. Zhang, A. Yu, Q. Fu, and X. Wei, “Supervised Deep Feature Extraction for Hyperspectral Image Classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 4, pp. 1909–1921, 2018, doi: 10.1109/TGRS.2017.2769673.

[67] Q. Liu, R. Hang, H. Song, and Z. Li, “Learning Multiscale Deep Features for High-Resolution Satellite Image Scene Classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 1, pp. 117–126, 2018, doi: 10.1109/TGRS.2017.2743243.

[68] M. Chi, Z. Sun, Y. Qin, J. Shen, and J. A. Benediktsson, "A Novel Methodology to Label Urban Remote Sensing Images Based on Location-Based Social Media Photos," *Proc. IEEE*, vol. 105, no. 10, pp. 1926–1936, 2017, doi: 10.1109/JPROC.2017.2730585.

[69] G. Cheng, J. Han, and X. Lu, "Remote Sensing Image Scene Classification: Benchmark and State of the Art," *Proc. IEEE*, vol. 105, no. 10, pp. 1865–1883, 2017, doi: 10.1109/JPROC.2017.2675998.

[70] Y. Yang and S. Newsam, "Bag-of-Visual-Words and Spatial Extensions for Land-Use Classification," in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2010, pp. 270–279. doi: 10.1145/1869790.1869829.

[71] G. S. Xia *et al.*, "AID: A benchmark data set for performance evaluation of aerial scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3965–3981, 2017, doi: 10.1109/TGRS.2017.2685945.

[72] G. Cheng, C. Yang, X. Yao, L. Guo, and J. Han, "When Deep Learning Meets Metric Learning : Remote Sensing Image Scene Classification via Learning Discriminative CNNs," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 5, pp. 2811–2821, 2018, doi: 10.1109/TGRS.2017.2783902.

[73] H. Li, C. Tao, Z. Wu, J. Chen, J. Gong, and M. Deng, "RSI-CB : A Large-Scale Remote Sensing Image Classification Benchmark via Crowdsource Data", [Online]. Available: https://github.com/lehaifeng/RSI-CB

[74] W. Teng, S. Member, N. Wang, H. Shi, Y. Liu, and J. Wang, "Classifier-Constrained Deep Adversarial Domain Adaptation for Cross-Domain Semisupervised Classification in Remote Sensing Images," *IEEE Geosci. Remote Sens. Lett.*, pp. 1–5, 2019, doi: 10.1109/LGRS.2019.2931305.

[75] B. Zhao, Y. Zhong, G. S. Xia, and L. Zhang, "Dirichlet-Derived Multiple Topic Scene Classification Model for High Spatial Resolution Remote Sensing Imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 4, pp. 2108–2123, 2016, doi: 10.1109/TGRS.2015.2496185.

[76] V. Unnikrishnan, A., Sowmya and P. K. Soman, "Deep learning architectures for land cover classification using red and near-infrared satellite images," *Multimed. Tools Appl.*, vol. 78, no. 13, pp. 18379–18394, 2019, doi: 10.1007/s11042-019-7179-2.

[77] B. Kitchenham, *Guidelines for performing systematic literature reviews in software engineering version 2.3. Technical Report EBSE-2007-01*, vol. Version 2. 2007. doi: 10.1109/ACCESS.2016.2603219.

[78] G. Xu, X. Zhu, and N. Tapper, "Using Convolutional Neural Networks Incorporating Hierarchical Active Learning for Target-Searching in Large-Scale Remote Sensing Images," *Int. J. Remote Sens.*, vol. 41, no. 11, pp. 4057–4079, 2020, doi: 10.1080/01431161.2020.1714774.

[79] W. Zhou, S. Newsam, C. Li, and Z. Shao, "PatternNet : A benchmark dataset for performance evaluation of remote sensing image retrieval," *ISPRS J. Photogramm. Remote Sens.*, vol. 145, pp. 197–209, 2018, doi: 10.1016/j.isprsjprs.2018.01.004.

[80] S. Li, W. Song, L. Fang, Y. Chen, P. Ghamisi, and J. Benediktsson, "Deep Learning for Hyperspectral Image Classification : An Overview," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 9, pp. 6690–6709, 2019, doi: 10.1109/TGRS.2019.2907932.

[81] A. Sharma, X. Liu, X. Yang, and D. Shi, "A patch-based convolutional neural network for remote sensing image classification," *Neural Networks*, vol. 95, pp. 19–28, 2017, doi: 10.1016/j.neunet.2017.07.017.

[82] U. Côté-Allard *et al.*, "Deep Learning for Electromyographic Hand Gesture Signal Classification Using Transfer Learning," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 27, no. 4, pp. 760–771, 2019, doi: 10.1109/TNSRE.2019.2896269.

[83] X. Liu, R. Wang, Z. Cai, Y. Cai, and X. Yin, "Deep Multigrained Cascade Forest for Hyperspectral Image Classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 10, pp. 8169–8183, 2019, doi: 10.1109/TGRS.2019.2918587.

[84] C. Deng, Y. Xue, X. Liu, C. Li, and D. Tao, "Active Transfer Learning Network: A Unified Deep Joint Spectral-Spatial Feature Learning Model for Hyperspectral Image Classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 3, pp. 1741–1754, 2019, doi: 10.1109/TGRS.2018.2868851.

[85] M. Xie, N. Jean, M. Burke, D. Lobell, and S. Ermon, "Transfer learning from deep features for remote sensing and poverty mapping," *ArXiv, arXiv:1510.00098v2*, 2016.

[86] A. Bahri, S. Ghofrani Majelan, S. Mohammadi, M. Noori, and K. Mohammadi, "Remote Sensing Image Classification via Improved Cross-Entropy Loss and Transfer Learning Strategy Based on Deep Convolutional Neural Networks," *IEEE Geosci. Remote Sens.*

Lett., vol. 17, no. 6, pp. 1087–1091, 2020, doi: 10.1109/LGRS.2019.2937872.

[87] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015, doi: 10.1007/s11263-015-0816-y.

[88] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, vol. 25, 2012, pp. 1097–1105.

[89] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-scale Image Recognition,” *arXiv Prepr. arXiv1409.1556v1*, pp. 1–10, 2014.

[90] C. Szegedy *et al.*, “Going Deeper with Convolutions,” *arXiv Prepr. arXiv1409.4842v1*, pp. 1–12, 2014.

[91] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *arXiv Prepr. arXiv1512.03385v1*, pp. 1–12, 2015.

[92] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” *arXiv Prepr. arXiv1704.04861v1*, pp. 1–9, 2017, doi: 10.48550/arXiv.1704.04861.

[93] M. Tan and Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” *arXiv Prepr. arXiv1905.11946v1*, pp. 1–10, 2019.

[94] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015, pp. 1–15.

[95] J. Wang, Y. Zhong, Z. Zheng, A. Ma, and L. Zhang, “RSNet: The Search for Remote Sensing Deep Neural Networks in Recognition Tasks,” *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 3, pp. 2520–2534, 2021, doi: 10.1109/TGRS.2020.3001401.

[96] D. Peng, L. Bruzzone, Y. Zhang, H. Guan, H. DIng, and X. Huang, “SemiCDNet: A Semisupervised Convolutional Neural Network for Change Detection in High Resolution Remote-Sensing Images,” *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 7, pp. 5891–5906, 2021, doi: 10.1109/TGRS.2020.3011913.

[97] X. Zheng, T. Gong, X. Li, and X. Lu, “Generalized Scene Classification from Small-Scale Datasets with Multitask Learning,” *IEEE Trans. Geosci. Remote Sens.*, vol. 60, pp. 1–11, 2022, doi: 10.1109/TGRS.2021.3116147.

[98] W. Huang, Q. Wang, and X. Li, “FEATURE SPARSITY IN CONVOLUTIONAL

NEURAL NETWORKS FOR SCENE CLASSIFICATION OF REMOTE SENSING IMAGE School of Computer Science and Center for OPTical IMagery Analysis and Learning (OPTIMAL),” in *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, 2019, pp. 3017–3020. doi: 10.1109/IGARSS.2019.8898875.

[99] M. M. Alam *et al.*, “Classification of Deep-SAT Images under Label Noise,” *Appl. Artif. Intell.*, vol. 35, no. 14, pp. 1196–1218, 2021, doi: 10.1080/08839514.2021.1975381.

[100] Q. Yuan, Y. Wei, X. Meng, H. Shen, and L. Zhang, “A Multiscale and Multidepth Convolutional Neural Network for Remote Sensing Imagery Pan-Sharpening,” *IEEE J. Sel. Top. Appl. EARTH Obs. Remote Sens.*, vol. 11, no. 3, pp. 978–989, 2018.

[101] Y. Chen, Y. Wang, Y. Gu, X. He, P. Ghamisi, and X. Jia, “Deep Learning Ensemble for Hyperspectral Image Classification,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 12, no. 6, pp. 1882–1897, 2019, doi: 10.1109/JSTARS.2019.2915259.

[102] W. Kang, Y. Xiang, F. Wang, and H. You, “DO-Net : Dual-Output Network for Land Cover Classification From Optical Remote Sensing Images,” *IEEE Geosci. Remote Sens. Lett.*, vol. 19, no. Art no. 8021205, pp. 1–5, 2022, doi: 10.1109/LGRS.2021.3114305.

[103] Q. Sang, Y. Zhuang, S. Dong, G. Wang, and H. Chen, “FRF-Net : Land Cover Classification From Large-Scale VHR Optical Remote Sensing Images,” *IEEE Geosci. Remote Sens. Lett.*, vol. 17, no. 6, pp. 1057–1061, 2020, doi: 10.1109/LGRS.2019.2938555.

[104] B. Li *et al.*, “Further Exploring Convolutional Neural Networks’ Potential for Land-Use Scene Classification,” *IEEE Geosci. Remote Sens. Lett.*, vol. 17, no. 10, pp. 1687–1691, 2020, doi: 10.1109/LGRS.2019.2952660.

[105] P. Du, E. Li, J. Xia, A. Samat, and X. Bai, “Feature and Model Level Fusion of Pretrained CNN for Remote Sensing Scene Classification,” *IEEE J. Sel. Top. Appl. EARTH Obs. Remote Sens.*, vol. 12, no. 8, pp. 2600–2611, 2019, doi: 10.1109/JSTARS.2018.2878037.

[106] N. Liu, L. Wan, Y. Zhang, T. Zhou, H. Huo, and T. Fang, “Exploiting Convolutional Neural Networks With Deeply Local Description for Remote Sensing Image Classification,” *IEEE Access*, vol. 6, pp. 11215–11228, 2018, doi: 10.1109/ACCESS.2018.2798799.

[107] R. Fan, R. Feng, L. Wang, J. Yan, and X. Zhang, “Semi-MCNN: A Semisupervised Multi-CNN Ensemble Learning Method for Urban Land Cover Classification Using Submeter

HRRS Images," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 13, pp. 4973–4987, 2020, doi: 10.1109/JSTARS.2020.3019410.

[108] Y. Zhang, X. Zheng, Y. Yuan, and X. Lu, "Attribute-Cooperated Convolutional Neural Network for Remote Sensing Image Classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 12, pp. 8358–8371, 2020, doi: 10.1109/TGRS.2020.2987338.

[109] Y. Han and J. Wang, "Application of Convolutional Neural Networks in Remote Sensing Image Classification," in *Proceedings - 2019 2nd International Conference on Safety Produce Informatization, IICSPI 2019*, 2019, pp. 279–282. doi: 10.1109/IICSPI48186.2019.9096058.

[110] Z. Shao, K. Yang, and W. Zhou, "Performance Evaluation of Single-Label and Multi-Label Remote Sensing Image Retrieval Using a Dense Labeling Dataset," *Remote Sens.*, vol. 10, no. 6, p. 964, 2018, doi: 10.3390/rs10060964.

[111] Y. Long, Y. Gong, Z. Xiao, and Q. Liu, "Accurate Object Localization in Remote Sensing Images Based on Convolutional Neural Networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 5, pp. 2486–2498, 2017, doi: 10.1109/TGRS.2016.2645610.

[112] G. J. Scott, R. A. Marcum, C. H. Davis, T. W. Nivin, A. U. C. Merced, and L. Use, "Fusion of Deep Convolutional Neural Networks for Land Cover Classification of High-Resolution Imagery," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 9, pp. 1638–1642, 2017, doi: 10.1109/LGRS.2017.2722988.

[113] J. J. Senecal, J. w. Sheppard, and J. A. Shaw, "Efficient Convolutional Neural Networks for Multi-Spectral Image Classification," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, no. 14-19 July 2019, pp. 1–8. doi: 10.1109/IJCNN.2019.8851840.

[114] C. Nagpal and S. R. Dubey, "A Performance Evaluation of Convolutional Neural Networks for Face Anti Spoofing," in *IJCNN 2019. International Joint Conference on Neural Networks*, 2019, no. 14-19 July 2019, pp. 1–8.

[115] A. Shabbir *et al.*, "Satellite and Scene Image Classification Based on Transfer Learning and Fine Tuning of ResNet50," *Math. Probl. Eng.*, vol. 2021, p. 5843816, 2021, doi: 10.1155/2021/5843816.

[116] J. Liang, Y. Deng, and D. Zeng, "A Deep Neural Network Combined CNN and GCN for Remote Sensing Scene Classification," *IEEE J. Sel. Top. Appl. EARTH Obs. Remote*

Sens., vol. 13, pp. 4325–4338, 2020, doi: 10.1109/JSTARS.2020.3011333.

[117] D. Chen, P. Hu, and X. Duan, "Complex Scene Classification of High Resolution Remote Sensing Images Based on DCNN Model," in *2019 10th International Workshop on the Analysis of Multitemporal Remote Sensing Images (MultiTemp)*, 2019, pp. 1–4. doi: 10.1109/Multi-Temp.2019.8866895.

[118] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "FULLY CONVOLUTIONAL NEURAL NETWORKS FOR REMOTE SENSING IMAGE CLASSIFICATION Inria Sophia Antipolis - M ´ editerran ´ ee , TITANE team ; 2 Inria Saclay , TAO team , France Email : emmanuel.maggiori@inria.fr," in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2016, pp. 5071–5074. doi: 10.1109/IGARSS.2016.7730322.

[119] B. Petrovska, E. Zdravevski, P. Lameski, R. Corizzo, I. Štajduhar, and J. Lerga, "Deep Learning for Feature Extraction in Remote Sensing: A Case-Study of Aerial Scene Classification," *Sensors*, vol. 20, no. 14, p. 3906, 2020, doi: 10.3390/s20143906.

[120] B. Hou, J. Li, X. Zhang, S. Wang, and L. Jiao, "Object Detection and Trcacking based on Convolutional Neural Networks for High-Resolution Optical Remote Sensing Video," in *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, 2019, pp. 5433–5436. doi: 10.1109/IGARSS.2019.8898173.

[121] Y. Hu, X. Li, N. Zhou, L. Yang, L. Peng, and S. Xiao, "A Sample Update-Based Convolutional Neural Network Framework for Object Detection in Large-Area Remote Sensing Images," vol. 16, no. 6, pp. 947–951, 2019, doi: 10.1109/LGRS.2018.2889247.

[122] J. Pang, C. Li, J. Shi, Z. Xu, and H. Feng, "R2-CNN: Fast Tiny Object Detection in Large-Scale Remote Sensing Images," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 8, pp. 5512–5524, 2019, doi: 10.1109/TGRS.2019.2899955.

[123] M. Kim *et al.*, "Convolutional Neural Network-Based Land Cover Classification Using 2-D Spectral Reflectance Curve Graphs With Multitemporal Satellite Imagery," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 11, no. 12, pp. 4604–4617, 2018, doi: 10.1109/JSTARS.2018.2880783.

[124] W. Zhang, P. Tang, and L. Zhao, "Remote Sensing Image Scene Classification Using CNN-CapsNet," *Remote Sens.*, vol. 11, no. 5, p. 494, 2019, doi: 10.3390/rs11050494.

[125] P. Lin, M. Sun, C. Kung, and T. Chiueh, "FloatSD : A New Weight Representation and

Associated Update Method for Efficient Convolutional Neural Network Training," *IEEE J. Emerg. Sel. Top. CIRCUITS Syst.*, vol. 9, no. 2, pp. 267–279, 2019, doi: 10.1109/JETCAS.2019.2911999.

[126] M. J. Bosco, G. Wang, and Y. Hategekimana, "Learning Multi-Granularity Neural Network Encoding Image Classification Using DCNNs for Easter Africa Community Countries," *IEEE Access*, vol. 9, no. 2021, pp. 146703–146718, 2021, doi: 10.1109/ACCESS.2021.3122569.

[127] C. Peng, Y. Li, L. Jiao, Y. Chen, and R. Shang, "Densely Based Multi-Scale and Multi-Modal Fully Convolutional Networks for High-Resolution Remote-Sensing Image Semantic Segmentation," *IEEE J. Sel. Top. Appl. EARTH Obs. Remote Sens.*, vol. 12, no. 8, pp. 2612–2626, 2019, doi: 10.1109/JSTARS.2019.2906387.

[128] M. J. Bosco and W. Guoyin, "Deeply Fine-Tune a Convolutional Neural Network in Remote Sensing Image Classification : Easter Africa Countries (EAC)," in *2021 IEEE 2nd International Conference on Pattern Recognition and Machine Learning (PRML)*, 2021, pp. 13–20. doi: 10.1109/PRML52754.2021.9520703.

[129] A. Alem and S. Kumar, "Transfer Learning Models for Land Cover and Land Use Classification in Remote Sensing Image," *Appl. Artif. Intell.*, vol. 60, no. 1, p. 2014192, 2022, doi: 10.1080/08839514.2021.2014192.

[130] X. Yin, W. Chen, X. Wu, and H. Yue, "Fine-tuning and Visualization of Convolutional Neural Networks," in *12th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2017, pp. 1310–1315. doi: 10.1109/ICIEA.2017.8283041.

[131] Y. Li, Y. Zhang, and Z. Zhu, "Learning Deep Networks under Noisy Labels for Remote Sensing Image Scene Classification," in *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, 2019, pp. 3025–3028. doi: 10.1109/IGARSS.2019.8900497.

[132] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout : A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, 2014.

[133] J. Zhang *et al.*, "Training Convolutional Neural Networks with Multi - Size Images and Triplet Loss for Remote Sensing Scene Classification," *Sensors*, vol. 20, no. 4, p. 1188, 2020, doi: 10.3390/s20041188.

[134] K. Simonyan and A. Zisserman, “Very Veep Convolutional Networks for Large-scale Image Recognition,” *arXiv:1409.1556v6*, pp. 1–14, 2015, doi: 10.48550/arXiv.1409.1556.

[135] H. A. Abdu, “Classification accuracy and trend assessments of land cover- land use changes from principal components of land satellite images,” *Int. J. Remote Sens.*, vol. 40, no. 4, pp. 1275–1300, 2019, doi: 10.1080/01431161.2018.1524587.

[136] C. Cao, S. Dragićević, and S. Li, “Land-use change detection with convolutional neural network methods,” *Environ. - MDPI*, vol. 6, no. 2, 2019, doi: 10.3390/environments6020025.

[137] X.-Y. Tong *et al.*, “Land-cover classification with high-resolution remote sensing images using transferable deep models,” *Remote Sens. Environ.*, vol. 237, no. February 2020, p. 111322, 2020, doi: 10.1016/j.rse.2019.111322.

[138] B. Fang, R. Kou, L. Pan, and P. Chen, “Category-sensitive domain adaptation for land cover mapping in aerial scenes,” *Remote Sens.*, vol. 11, no. 22, p. 2631, 2019, doi: 10.3390/rs11222631.

[139] S. Hung, H. Wu, and M.-H. Tseng, “Remote Sensing Scene Classification and Explanation Using RSSCNet and LIME,” *Appl. Sci.*, vol. 10, no. 18, p. 6151, 2020, doi: https://doi.org/10.3390/app10186151.

[140] Y. Yao, H. Zhao, D. Huang, and Q. Tan, “Remote sensing scene classification using multiple pyramid pooling,” *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. Photogramm. Image Anal. Munich Remote Sens. Symp.*, vol. XLII-2/W16, no. 18-20 September,2019, Munich, Germany, 2019.

[141] G. Cheng, X. Xie, J. Han, L. Guo, and G. S. Xia, “Remote Sensing Image Scene Classification Meets Deep Learning: Challenges, Methods, Benchmarks, and Opportunities,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 13, pp. 3735–3756, 2020, doi: 10.1109/JSTARS.2020.3005403.

[142] D. Zhang, Z. Liu, and X. Shi, “Transfer learning on EfficientNet for remote sensing image classification,” in *Proceedings - 2020 5th International Conference on Mechanical, Control and Computer Engineering, ICMCCE 2020*, 2020, pp. 2255–2258. doi: 10.1109/ICMCCE51767.2020.00489.

[143] A. Vali, S. Comai, and M. Matteucci, “Deep Learning for Land Use and Land Cover Classification based on Hyperspectral and Multispectral Earth Observation Data : A

Review,” *Remote Sens.*, vol. 12, no. 15, p. 2495, 2020, doi: 10.3390/rs12152495.

[144] X. Pan, J. Wang, X. Zhang, Y. Mei, L. Shi, and G. Zhong, “A deep-learning model for the amplitude inversion of internal waves based on optical remote-sensing images,” *Int. J. Remote Sens.*, vol. 39, no. 3, pp. 607–618, 2018, doi: 10.1080/01431161.2017.1390269.

[145] W. Zhao, Z. Guo, J. Yue, X. Zhang, and L. Luo, “On combining multiscale deep learning features for the classification of hyperspectral remote sensing imagery,” *Int. J. Remote Sens.*, vol. 36, no. 13, pp. 3368–3379, 2015, doi: 10.1080/2150704X.2015.1062157.

[146] M. Rashid *et al.*, “A Sustainable Deep Learning Framework for Object Recognition using Multi-layers Deep Features Fusion and Selection,” *Sustain.*, vol. 12, no. 12, p. 5037, 2020, doi: 10.3390/su12125037.

[147] B. G. Weinstein, S. Marconi, S. Bohlman, A. Zare, and E. White, “Individual tree-crown detection in rgb imagery using semi-supervised deep learning neural networks,” *Remote Sens.*, vol. 11, no. 11, p. 1309, 2019, doi: 10.3390/rs11111309.

[148] M. Rostami, S. Kolouri, E. Eaton, and K. Kim, “Deep transfer learning for few-shot SAR image classification,” *Remote Sens.*, vol. 11, no. 11, p. 1374, 2019, doi: 10.3390/rs11111374.

[149] M. Y. Zou and Y. Zhong, “Transfer Learning for Classification of Optical Satellite,” *Sens. Imaging*, vol. 19, no. 1, pp. 1–13, 2018, doi: 10.1007/s11220-018-0191-1.

[150] S. Kumar, M. Talib, Naman, and P. Verma, “Covid Detection from X-RAY and CT Scans using Transfer Learning – A Study,” in *Proceedings of the International Conference on Artificial Intelligence and Smart Systems (ICAIS-2021)*, 2021, pp. 85–92. doi: 10.1109/ICAIS50930.2021.9395784.

[151] Z. Chen, T. Zhang, and C. Ouyang, “End-to-end airplane detection using transfer learning in remote sensing images,” *Remote Sens.*, vol. 10, no. 1, pp. 1–15, 2018, doi: 10.3390/rs10010139.

[152] Y. Zhu *et al.*, “Heterogeneous Transfer Learning for Image Classification,” *Proc. Twenty-Fifth AAAI Conf. Artif. Intell. Heterog.*, pp. 1304–1309, 2011.

[153] X. Zhang, Y. Guo, and X. Zhang, “Deep convolutional neural network structure design for remote sensing image scene classification based on transfer learning,” in *IOP Conf. Series: Earth and Environmental Science 569 (2020) 012046 IOP*, 2020, p. 012046. doi: 10.1088/1755-1315/569/1/012046.

[154] H. Astola, L. Seitsonen, E. Halme, M. Molinier, and A. Lönnqvist, “Deep neural networks with transfer learning for forest variable estimation using sentinel-2 imagery in boreal forest,” *Remote Sens.*, vol. 13, no. 12, p. 2392, 2021, doi: 10.3390/rs13122392.

[155] M. Zhu, Y. Xu, S. Ma, S. Li, H. Ma, and Y. Han, “Effective airplane detection in remote sensing images based on multilayer feature fusion and improved nonmaximal suppression algorithm,” *Remote Sens.*, vol. 11, no. 9, p. 1062, 2019, doi: 10.3390/rs11091062.

[156] Y. Qian, W. Zhou, W. Yu, L. Han, W. Li, and W. Zhao, “Integrating backdating and transfer learning in an object-based framework for high resolution image classification and change analysis,” *Remote Sens.*, vol. 12, no. 24, p. 4094, 2020, doi: 10.3390/rs12244094.

[157] M. Liu, B. Fu, D. Fan, P. Zuo, S. Xie, and H. He, “Study on transfer learning ability for classifying marsh vegetation with multi-sensor images using DeepLabV3+ and HRNet deep learning algorithms,” *Int. J. Appl. Earth Obs. Geoinf.*, vol. 103, p. 102531, 2021.

[158] R. Naushad and T. Kaur, “Deep Transfer Learning for Land Use Land Cover Classification : A Comparative Study,” *arXiv:2110.02580v2, Online available at https://arxiv.org/abs/2110.02580*, 2021.

[159] A. Alem and S. Kumar, “Deep Learning Methods for Land Cover and Land Use Classification in Remote Sensing: A Review,” in *ICRITO 2020 - IEEE 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)*, 2020, no. Amity University, Noida, India. June 4-5, 2020, pp. 903–908. doi: 10.1109/ICRITO48877.2020.9197824.

[160] V. Risojevic and V. Stojnic, “The Role of Pre-Training in High-Resolution Remote Sensing Scene Classification,” *arXiv:2111.03690v1*, vol. 14, no. 8, pp. 1–10, 2021.

[161] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[162] J. Xiao, J. Wang, S. Cao, and B. Li, “Application of a Novel and Improved VGG-19 Network in the Detection of Workers Wearing Masks,” *J. Phys. Conf. Ser.*, vol. 1518, no. 2020, p. 012041, 2020, doi: 10.1088/1742-6596/1518/1/012041.

[163] M. Mateen, J. Wen, Nasrullah, S. Song, and Z. Huang, “Fundus Image Classification Using VGG-19 Architecture with PCA and SVD,” *Symmetry (Basel).*, vol. 11, no. 1, p. 1,

2019, doi: 10.3390/sym11010001.

[164] C. Szegedy, V. Vanhoucke, S. Ioffe, and J. Shlens, “Rethinking the Inception Architecture for Computer Vision,” *arXiv:1512.00567v3*, 2015.

[165] V. Nair and G. E. Hinton, “Rectified Linear Units Improve Restricted Boltzmann Machines,” in *Proceedings of the 27th International Conference on International Conference on Machine Learning, Haifa, Israel,21–24 June 2010*, 2010, pp. 807–814. doi: 10.1123/jab.2016-0355.

[166] E. A. Alshari and B. W. Gawali, “Development of classification system for LULC using remote sensing and GIS,” *Glob. Transitions Proc.*, vol. 2, no. 1, pp. 8–17, 2021, doi: 10.1016/j.gltp.2021.01.002.

[167] G. Cheng, Z. Li, J. Han, X. Yao, and L. Guo, “Exploring Hierarchical Convolutional Features for Hyperspectral Image Classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 11, pp. 6712–6722, 2018, doi: 10.1109/TGRS.2018.2841823.

[168] J. Han, D. Zhang, G. Cheng, L. Guo, and J. Ren, “Object Detection in Optical Remote Sensing Images based on Weakly Supervised Learning and High-Level Feature Learning,” *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 6, pp. 3325–3337, 2015, doi: 10.1109/TGRS.2014.2374218.

[169] O. Fink, Q. Wang, M. Svensén, P. Dersin, W.-J. Lee, and M. Ducoffe, “Potential, Challenges and Future Directions for Deep Learning in Prognostics and Health Management Applications,” *Eng. Appl. Artif. Intell.*, vol. 92, no. May, p. 103678, 2020, doi: 10.1016/j.engappai.2020.103678.

[170] U. Zahid *et al.*, “BrainNet : Optimal Deep Learning Feature Fusion for Brain Tumor Classification,” *Comput. Intell. Neurosci.*, vol. 2022, 2022, doi: 10.1155/2022/1465173.

[171] B. Yang, S. Hu, Q. Guo, and D. Hong, “Multisource Domain Transfer Learning Based on Spectral Projections for Hyperspectral Image Classification,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 15, pp. 3730–3739, 2022, doi: 10.1109/JSTARS.2022.3173676.

[172] H. Alhichri, A. S. Alswayed, Y. Bazi, N. Ammour, and N. A. Alajlan, “Classification of Remote Sensing Images Using EfficientNet-B3 CNN Model with Attention,” *IEEE Access*, vol. 9, no. 2021, pp. 14078–14094, 2021, doi: 10.1109/ACCESS.2021.3051085.

[173] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, “Fully convolutional neural

networks for remote sensing image classification," in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2016, pp. 5071–5074. doi: 10.1109/IGARSS.2016.7730322.

[174] Z. Zeng, X. Chen, and Z. Song, "MGFN: A Multi-Granularity Fusion Convolutional Neural Network for Remote Sensing Scene Classification," *IEEE Access*, vol. 9, pp. 76038–76046, 2021, doi: 10.1109/ACCESS.2021.3081922.

[175] Y. Hu, Q. Zhang, Y. Zhang, and H. Yan, "A deep Convolution Neural Network Method for Land Cover Mapping: A Case Study of Qinhuangdao, China," *Remote Sens.*, vol. 10, no. 12, pp. 1–17, 2018, doi: 10.3390/rs10122053.

[176] B. Zhao, Y. Zhong, L. Zhang, and B. Huang, "The fisher Kernel coding framework for high spatial resolution scene classification," *Remote Sens.*, vol. 8, no. 2, p. 157, 2016, doi: 10.3390/rs8020157.

[177] Q. Zhu, Y. Zhong, B. Zhao, G. Xia, and L. Zhang, "Bag-of-visual-words scene classifier combining local and global features for high spatial resolution imagery," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 6, pp. 747–751, 2016, doi: 10.1109/LGRS.2015.2513443.

APPENDICES

Appendix I: Journal and Conference Publications

Journal Papers:

1. A. Alem and S. Kumar (2022). Deep Learning Models Performance Evaluations for Remote Sensed Image Classification, IEEE Access, vol. 10, pp. 111784-111793, DOI: 10.1109/ACCESS.2022.3215264 (**SCIE, IF: 3.476, best quartile- Q1**)
2. A. Alem and S. Kumar (2022). End-to-end Convolutional Neural Network Feature Extraction for Remote Sensed Images Classification, Applied Artificial Intelligence, vol. 36 (1), art. no. 2137650, DOI: 10.1080/08839514.2022.2137650 (**SCIE, IF: 2.777, best quartile- Q2**)
3. A. Alem and S. Kumar (2021). Transfer Learning Models for Land Cover and Land Use Classification in Remote Sensing Image, Applied Artificial Intelligence, vol. 36 (1), art. no. 2014192, DOI: 10.1080/08839514.2021.2014192 (**SCIE, IF: 2.777, best quartile- Q2**)

Published and Presented Conference Papers:

1. A. Alem and S. Kumar (2022), "Deep Learning Models for Remote Sensed Hyperspectral Image Classification," 2022 13th International Conference on Computing, Communication and network Technologies (ICCCNT), Kharagpur, India, Oct 3-5, 2022, pp. 1-7, doi: 10.1109/ICCCNT54827.2022.9984282. (**IEEE Explore Scopus Indexed**)
2. A. Alem and S. Kumar, "Deep Learning Methods for Land Cover and Land Use Classification in Remote Sensing: A Review," 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2020, pp. 903-908, doi: 10.1109/ICRITO48877.2020.9197824. (**IEEE Explore Scopus Indexed**)
3. Abebaw Alem, Shailender Kumar, Transfer Learning for Land Use Classification in Remote Sensing. In Jamia Teachers' Association Multidisciplinary International Conference (JTACON-2020), New Delhi, India, February 16-18, 2020. (**Presented**)

Appendix II: List of ML and DL Tools and Package Libraries for applications

No	Python Libraries	Type of library	Descriptions
1	Apache Singa	ML, DL, NLP, Image processing	Supporting scalable and distributed training for healthcare applications using ML algorithms
2	H2o	ML framework	Used for statistical, ML and AI algorithms
3	HDF5	Data manipulation	Enabling the storage of huge amounts of numerical data and manipulating the data easily from NumPy
4	Keras, TensorFlow, Theano, Caffe, Torch	DL	Providing fast and easy scientific computing of numerical data with deep neural networks, effectively handling mathematical expressions such as matrix values, modeling for language and vision applications
5	Matplotlib, Seaborn, Bokeh, Plotly, NetworkX, Basemap,d3py, ggplot, prettyplotlib	Visualization	Visualizing the data from Python quickly Plotting 2D graphs in various formats such as bar charts, plots, histograms, error charts, power spectra, and scatter plots across platforms using a few lines of code
6	MLlib	ML	Encompassing sets of ML algorithms like classification and clustering
7	NumPy	Numerical Operations	Supporting the scientific computing that is high-level mathematical functions over large, multi-dimensional arrays and matrices
8	NumPy, SciPy, matplotlib, OpenCV, scikit-learn, scikit-image, ilastik	Image Processing	Providing a set of algorithms for image processing, supporting geometric transformations, segmentation, filtering, color space manipulation, morphology, analysis, and feature detection
9	Pandas	Data Analysis	Offering high-performance operations and data structures for time series and numerical tables manipulation
10	PyBrain	Neural Network	Providing algorithms for reinforcement learning, neural networks, unsupervised learning, and evolution to analyze large-scale data
11	RankLib	ML	Providing a set of learning to rank algorithms and supporting the evaluation using retrieval metrics
12	Scikit- Learn, Shogun, Pattern, PyLearn2, PyMC	ML Algorithms	Providing ML techniques (such as classification, clustering, and regression), Interoperating with the numerical and scientific libraries (such as NumPy and SciPy)
13	Shogun	ML	Providing different data structures and algorithms for ML problems
14	Statsmodel	Statistical Analysis	Conducting statistical data exploration and statistical tests Performing statistical computations such as descriptive statistics and providing classes and functions to estimate different statistical models
15	SymPy	Statistical Applications	Supporting symbolic mathematics and modeling the full-featured Computer Algebra System (CAS)