

GEOSPATIAL ARTIFICIAL INTELLIGENCE FOR LAND-COVER CHARACTERIZATION  
BASED ON REMOTE-SENSING DATA ANALYSIS

A Dissertation

by

ANDONG MA

Submitted to the Graduate and Professional School of  
Texas A&M University  
in partial fulfillment of the requirements for the degree of  
DOCTOR OF PHILOSOPHY

Chair of Committee, Anthony M. Filippi  
Committee Members, Inci Güneralp  
Burak Güneralp  
Zhangyang Wang  
Head of Department, David Cairns

December 2021

Major Subject: Geography

Copyright 2021 Andong Ma

## ABSTRACT

Remote sensing (RS), a critical technology for large-scale-monitoring Earth-observing systems (EOS), plays an important role in Earth science and other related fields where physical, biological, and chemical properties of the Earth can be characterized in a non-contact manner. One of the most widely used techniques to analyze land-cover information utilizing RS images is pixel-level classification, where each individual pixel will be classified with a semantic label by employing machine learning (ML) algorithms. In recent decades, with the tremendous developments in hardware and software, computational capabilities have improved dramatically, which has facilitated the development of deep-learning (DL) algorithms derived from traditional ML. Among various deep learning models, recurrent neural networks (RNNs), which are able to process sequential inputs by utilizing a series of internal state, attracts more attention for the purpose of handling multi-temporal RS image analysis due to their intrinsic recurrent structure. However, their application on single-image classification still needs more investigations, especially from the perspective of sequential feature extraction. To address this limitation, we propose a similarity measurements-based sequential feature extraction method for single RS image classification using long short-term memory (LSTM), a special class of RNN. For a given pixel the proposed framework utilizes the spectral information of those pixels collected from the whole image instead of the individual spectral vector of its own. And its classification performance on two standard datasets demonstrates its effectiveness compared with other benchmark algorithms.

However, the computational time cost of the aforementioned approach is a critical issue as all pixels in that single RS image need to be considered during similarity measurement for every pixel. That brings more difficulties, especially for processing large-scale RS image. Therefore, building upon the previous work, we improve that model by adding segmentation map as an additional criterion for shrinking searching range from the whole image to selected segments. Within such a segmentation map, homogeneous pixels will be aggregated into adjacent segments. Thus, the similarity measurement will be split into two phases, including segment-level similarity calcu-



lation and pixel-level similarity calculation. Experimental results obtained from three benchmark hyperspectral RS images and large-scale satellite images illustrate that the proposed approaches achieve promising classification performance with lower computational time cost.

Besides classification, another application of deep learning is object detection where interest of object is allowed to identify and localized in an image or video. In this study, our focus is to detect fallen trees from large-scale aerial photos. Therefore, we develop a framework to automatically create image and annotation dataset which can be utilized as input data for deep learning based object detection model. Quantitative and qualitative classification results illustrate that, for isolated fallen trees, our model achieves promising results considering recall index. However, many false positive detection results are generated as well. Then we further investigate the influence of those fallen trees located on the boundary of extracted sub-image and find that using geographical coordinates to calculate accuracy metrics can achieve better results. Furthermore, we also explore the issue of overlapping fallen trees and observe that overlapping fallen tree introduce more detection errors which can be alleviated to some extent by considering fallen tree clusters. However, detecting overlapping fallen trees is still a challenging task, especially for those large scale datasets collected from the real world.

## DEDICATION

To my dear daughter, my wife, and my family.

## ACKNOWLEDGMENTS

First and foremost, I would like to express my deep sense of gratitude and thanks to my advisor, Dr. Anthony M. Filippi, for his invaluable advice, patience, and continuous support during my Ph.D. study. His rigorous attitude towards research always motivate me to seek the truth behind the observation. And his optimistic character encourages me a lot when I am having difficulties in my research and even in my daily life. I can not believe that this dissertation will be finished without his constant support and timely advice. It is a great honor for me to be his student and work with him over the five years.

I would like to thank Dr. Inci Güneralp and Dr. Burak Güneralp for all your continuous support on my dissertation research by providing opportunities exploring the applications of those machine learning and deep learning algorithms on real-world problem.

I would like to thank Dr. Zhangyang Wang not only for your machine learning course which guilds me in this area, but also for your generous advice on the my dissertation research. Every time I communicate with you, I am always inspired by your creative insight and unique thoughts.

And I am extremely thankful to all my friends for the support during my Ph.D. life. I would like to thank Da Huo, Zhengcong Yin, Jinwoo Park, Mengqu Han, Xingchen Chen, and Xiao Li for their wonderful friendship that makes my life in the United States amazing.

Finally, I would like to thank my dear daughter, my wife and all my family members for your love. This is for you.

## CONTRIBUTORS AND FUNDING SOURCES

### **Contributors**

This work was supported by a dissertation committee consisting of Professor Anthony M. Filippi, Inci Güneralp, and Burak Güneralp of the Department of Geography and Professor Zhangyang Wang of the Department of Computer Science and Engineering.

The ground reference dataset utilized in Chapter 3 was provided by Brendan Lawrence. The digitized fallen tree dataset utilized in Chapter 4 was provided by Mengqu Han.

All other work conducted for the dissertation was completed by the student independently.

### **Funding Sources**

Graduate study was supported by research projects from NASA, NSF, and Microsoft.

## NOMENCLATURE

1D	One-Dimensional
2D	Two-Dimensional
3D	Three-Dimensional
AA	Average Accuracy
ALS	Airborne Laser Scanning
ANN	Artificial Neural Network
AVIRIS	Airborne Visible/Infrared Imaging Spectrometer
BM	Block Matching
CasRNN	Cascaded Recurrent Neural Network
CC	Cloud Cover
CNN	Convolutional Neural Network
DOS	Disk Operating System
CRF	Conditional Random Field
CRNN	Convolutional Recurrent Neural Network
DBN	Deep Belief Network
DEM	Digital Elevation Model
DL	Deep Learning
DTM	Digital terrain mode
EMR	Electromagnetic Radiation
EOS	Earth Observing Systems
ETM+	Enhanced Thematic Mapper Plus
EU	Euclidean Distance

Fast R-CNN	Fast Region-based Convolutional Neural Network
Faster R-CNN	Faster Region-based Convolutional Neural Network
FN	False Negative
FNEA	Fractal Net Evolution Approach
FP	False Positive
GAN	Generative Adversarial Network
GLCM	Grey-Level Co-occurrence Matrix
GRU	Gated Recurrent Unit
HMM	Hidden Markov Mode
HPRC	High Performance Research Computing
HSI	Hyperspectral Remote-sensing Image
IOU	Intersection Over Union
IPD	Image Patch Distance
ISODATA	Iterative Self-Organizing Data Analysis Technique
k-NN	K-Nearest Neighbor
LDA	Linear Discriminant Analysis
LiDAR	Light Detection And Ranging
LSTM	Long Short-Term Memory
MANERR	Mission-Aransas National Estuarine Research Reserve
NDVI	Normalized Difference Vegetation Index
NLP	Natural Language Processing
OA	Overall Accuracy
OBIA	Object-Based Image Analysis
PC	Principal Component
PCA	Principal Component Analysis

PM	Pixel Matching
Pretanh	Parametric Rectified Hyperbolic Tangent Function
RBF	Radial Basis Function
RCNN	Recurrent Convolutional Neural Network
R-CNN	Region-based Convolutional Neural Network
RF	Random forest
RNN	Recurrent Neural Network
RODIS	Reflective Optics System Imaging Spectrometer
RPCA	Randomized Principal Component Analysis
RPN	Region Proposal Network
RS	Remote Sensing
SAE	Stacked Autoencoder
SAM	Spectral Angle Mapper
SAR	Synthetic Aperture Radar
SRTM	Shuttle Radar Topography Mission
SSFE	Single-image Sequential Feature Extraction
SSM	Spatial Similarity Measurement
SVM	Support Vector Machine
TM	Thematic Mapper
TP	True Positive
TSVM	Transductive Support Vector Machine
UAV	Unmanned Aerial Vehicle

## TABLE OF CONTENTS

	Page
ABSTRACT .....	ii
DEDICATION .....	iv
ACKNOWLEDGMENTS .....	v
CONTRIBUTORS AND FUNDING SOURCES .....	vi
NOMENCLATURE .....	vii
TABLE OF CONTENTS .....	x
LIST OF FIGURES .....	xiii
LIST OF TABLES.....	xvi
1. INTRODUCTION.....	1
2. HYPERSPECTRAL IMAGE CLASSIFICATION USING SIMILARITY MEASUREMENTS- BASED DEEP RECURRENT NEURAL NETWORKS .....	4
2.1 Introduction.....	4
2.2 Background: RNN and LSTM .....	9
2.2.1 RNN.....	9
2.2.2 LSTM .....	10
2.3 Spatial Similarity Measurements in LSTM .....	12
2.3.1 Pixel Matching.....	12
2.3.2 Block Matching .....	13
2.3.3 Sequential Feature Extraction.....	15
2.4 Experimental Setup, Results, and Discussion.....	16
2.4.1 Datasets .....	16
2.4.2 Experimental Design .....	16
2.4.3 Classification Results: Pavia University Image .....	19
2.4.4 Classification Results: Salinas Image .....	20
2.4.5 Parameter Sensitivity Analysis .....	23
2.5 Conclusions.....	30
3. FAST SEQUENTIAL FEATURE EXTRACTION FOR RECURRENT NEURAL NETWORK- BASED HYPERSPECTRAL IMAGE CLASSIFICATION .....	32



3.1	Introduction.....	32
3.2	Related work .....	38
3.2.1	RNN and LSTM .....	38
3.2.2	Object-based Segmentation .....	40
3.3	Improved single-image sequential feature extraction .....	42
3.3.1	Segment Feature Extraction.....	43
3.3.2	Segments-based Similarity Measurements .....	44
3.3.3	Improved Single-image Sequential Feature Extraction .....	46
3.4	Experiments .....	48
3.4.1	Datasets .....	48
3.4.2	Experimental Setup.....	54
3.4.3	Classification Performance Comparisons .....	59
3.4.4	Parameter Analysis .....	62
3.4.4.1	Number of training samples .....	68
3.4.4.2	Sequence length .....	68
3.4.4.3	Scale parameter in FNEA .....	70
3.4.4.4	Computational time cost for SSFE .....	70
3.5	Conclusion.....	71
4.	<b>CHARACTERIZATION OF LAND COVER INFORMATION USING SINGLE-IMAGE BASED RECURRENT NEURAL NETWORK AND LANDSAT IMAGERY: A CASE STUDY IN NORTHERN INDIA.....</b>	<b>73</b>
4.1	Introduction.....	73
4.2	Material .....	77
4.2.1	Study site and field data .....	77
4.2.2	Landsat image acquisition and preprocessing .....	77
4.2.3	Ground reference data generation .....	80
4.3	Single-image-based sequential feature extraction and classification .....	80
4.4	Experiments .....	82
4.4.1	Experimental setup .....	82
4.4.2	Classification results.....	83
4.5	Conclusion.....	87
5.	<b>FALLEN TREE DETECTION USING HIGH-RESOLUTION AERIAL PHOTOS AND DEEP LEARNING IN A COASTAL RIVER AND ITS FLOODPLAIN .....</b>	<b>90</b>
5.1	Introduction.....	90
5.2	Background: CNN and Faster R-CNN.....	94
5.2.1	CNN.....	94
5.2.2	Faster R-CNN .....	95
5.3	Study area and Data .....	96
5.4	Methodology .....	97
5.4.1	Fallen tree digitization .....	99
5.4.2	Automatic image and annotation generation for deep learning-based object detection model.....	100

5.5	Experiments .....	107
5.5.1	Experimental setup .....	107
5.5.2	Experimental results on testing dataset .....	110
5.5.3	Analysis on different areas with different overlapping level .....	112
5.6	Conclusion.....	117
6.	CONCLUSION.....	122
	REFERENCES .....	124

## LIST OF FIGURES

FIGURE	Page
2.1 Architecture of the proposed methods .....	9
2.2 The basic structure of an unrolled long short-term memory LSTM unit .....	11
2.3 False-color image composites and their corresponding ground-reference data. <b>(a)</b> false-color composite of Pavia University image bands (R: band 55, G: band 33, and B: band 13); <b>(b)</b> ground-reference data (with class legend) for the Pavia University image; <b>(c)</b> false-color composite of Salinas image bands (R: band 57, G: band 27, and G: band 17); <b>(d)</b> ground-reference data (with class legend) for the Salinas image	17
2.4 Classification maps for the Pavia University image from the fifth trial: <b>(a)</b> ground-reference map; <b>(b)</b> SVM, with OA = 80.04%; <b>(c)</b> 1DCNN, with OA = 78.32%; <b>(d)</b> 1DLSTM, with OA = 83.72%; <b>(e)</b> LSTM_PM_EU, with OA = 83.81%; <b>(f)</b> LSTM_PM_SAM, with OA = 86.78%; <b>(g)</b> LSTM_BM_EU, with OA = 93.18%; and <b>(h)</b> LSTM_BM_SAM, with OA = 96.01%. The red-rectangle and red-circle annotations represent sample areas of interest, discussed in the text .....	22
2.5 Classification maps of Salinas image from the fifth trial: <b>(a)</b> ground-reference map; <b>(b)</b> SVM, with OA = 83.38%; <b>(c)</b> 1DCNN, with OA = 87.00%; <b>(d)</b> 1DLSTM, with OA = 86.85%; <b>(e)</b> LSTM_PM_EU, with OA = 86.21%; <b>(f)</b> LSTM_PM_SAM, with OA = 87.13%; <b>(g)</b> LSTM_BM_EU, with OA = 90.02%; and <b>(h)</b> LSTM_BM_SAM, with OA = 90.72%. The red-circle and red-rectangle annotations represent sample areas of interest, discussed in the text .....	25
2.6 Analysis of different sequential feature lengths. <b>(a)</b> OAs based on the Pavia University image data; and <b>(b)</b> OAs based on the Salinas image data.....	27
2.7 Analysis of different window sizes. <b>(a)</b> OAs based on the Pavia University image data; and <b>(b)</b> OAs based on the Salinas image data .....	29
3.1 Analysis of different window sizes. <b>(a)</b> OAs based on the Pavia University image data; and <b>(b)</b> OAs based on the Salinas image data .....	35
3.2 Architecture of proposed SSFE-based LSTM HSI classification framework .....	42
3.3 The proposed three SSFE methods : <b>(a)</b> Local segment-based SSFE; <b>(b)</b> Non-local segment-based SSFE; and <b>(c)</b> mixed segments-based SSFE .....	45

3.4	Pavia University hyperspectral image: (a) False-color composite (R: band 55, G: band 33, and B: band 13); (b) Ground-reference data .....	52
3.5	Salinas hyperspectral image: (a) False-color composite (R: band 57, G: band 27, and B: band 17); (b) Ground-reference data .....	53
3.6	Indian Pines hyperspectral image: (a) False-color composite (R: band 57, G: band 27, and B: band 18); (b) Ground-reference data .....	53
3.7	Classification results from the PU image, from the eighth trial: (a) SVM (82.65%); (b) 1D_CNN (84.98%); (c) 1D_LSTM (83.52%); (d) BM_LSTM (95.94%); (e) LS <sup>3</sup> FE_LSTM (PM) (93.39%); (f) LS <sup>3</sup> FE_LSTM (BM) (95.24%); (g) NS <sup>3</sup> FE_LSTM (PM) (86.31%); (h) NS <sup>3</sup> FE_LSTM (BM) (90.36%); (i) MS <sup>3</sup> FE_LSTM (PM) (93.83%); and (j) MS <sup>3</sup> FE_LSTM (BM) (97.28%) .....	56
3.8	Classification results from the Salinas image, from the eighth trial: (a) SVM (85.23%); (b) 1D_CNN (85.98%); (c) 1D_LSTM (85.17%); (d) BM_LSTM (91.94%); (e) LS <sup>3</sup> FE_LSTM (PM) (94.17%); (f) LS <sup>3</sup> FE_LSTM (BM) (95.29%); (g) NS <sup>3</sup> FE_LSTM (PM) (92.92%); (h) NS <sup>3</sup> FE_LSTM (BM) (92.95%); (i) MS <sup>3</sup> FE_LSTM (PM) (94.91%); and (j) MS <sup>3</sup> FE_LSTM (BM) (96.85%) .....	57
3.9	Classification results from the INP image, from the eighth trial: (a) SVM (86.00%); (b) 1D_CNN (79.71%); (c) 1D_LSTM (76.46%); (d) BM_LSTM (89.91%); (e) LS <sup>3</sup> FE_LSTM (PM) (93.43%); (f) LS <sup>3</sup> FE_LSTM (BM) (94.58%); (g) NS <sup>3</sup> FE_LSTM (PM) (86.57%); (h) NS <sup>3</sup> FE_LSTM (BM) (87.16%); (i) MS <sup>3</sup> FE_LSTM (PM) (92.60%); and (j) MS <sup>3</sup> FE_LSTM (BM) (93.79%) .....	58
3.10	Classification accuracies of different models using different numbers of training samples per class on three experimental datasets: (a) PU image; (b) Salinas image; and (c) INP image .....	65
3.11	Classification accuracies of proposed SSFE-based models using different sequence lengths on three experimental datasets: (a) PU image; (b) Salinas image; and (c) INP image .....	66
3.12	Classification accuracies of proposed SSFE-based models using segmentation map with different scale parameters on three experimental datasets: (a) PU image; (b) Salinas image; and (c) INP image .....	67
4.1	The study area. Base image is collected from Landsat 8 where false color composite is applied (R: Band 5, G: Band 4, and B: Band 3) .....	78
4.2	Single-image-based sequential feature extraction and classification framework.....	81

4.3	Classification results obtained from 2019-pair image. (a1), (a2), and (a3) are false color composite of 2018-pair image collected from three different areas. (b1), (b2) and (b3) are training samples. (c1), (c2) and (c3) are classification results collected from 2DCNN model. (d1), (d2) and (d3) are classification maps generated from SSFE-base LSTM approach .....	85
5.1	The architecture of Faster R-CNN. “Conv” is convolutional layer, “FC” represents fully connected layer. “Bbox_pred” is position offset of predicted bounding box, and “Cls_prob” denotes category probability of predicted bounding box .....	96
5.2	Study area for this research .....	98
5.3	Architecture of proposed framework. Red polygons are digitized fallen trees. Yellow rectangles are the minimum bounding boxes of fallen trees. Blue rectangles are object detection results .....	99
5.4	Examples of fallen tree digitization. Red polygons are digitized fallen trees .....	100
5.5	Example of geographical coordinates for fallen tree bounding box and aerial photo ..	102
5.6	Calculation of IOU between two bounding boxes. Red rectangle is ground reference bounding box, and blue rectangle is detected bounding box .....	109
5.7	Some object detection results. Red rectangle is ground reference bounding box, and blue rectangle is detected bounding box .....	111
5.8	Some object detection results collected from two datasets with different overlapping degrees .....	114
5.9	Example of creating clusters based on overlapped fallen trees. Red rectangles represent the original polygons, and yellow rectangles denote merged polygons. Merged bounding boxes are highlighted in blue rectangles .....	115
5.10	Recall and precision using different IOU parameters .....	118
5.11	Recall results using different IOU parameters .....	119
5.12	Precision results using different IOU parameters .....	120

## LIST OF TABLES

TABLE	Page
2.1 Class codes for Pavia University and Salinas images .....	17
2.2 Parameter settings for 1D-CNN and 1D-LSTM, where the convolutional layer is represented as “Conv(number of kernels)-(kernel size)”, maxpooling layer is performed as “Maxpooling-(kernel size)”, and LSTM layer is denoted as “LSTM-(kernel size)” .....	19
2.3 Comparison of different classification accuracy results for the Pavia University image (%), where LSTM_PEU equals LSTM_PM_EU, LSTM_PSAM equals LSTM_PM_SAM, LSTM_BEU equals LSTM_BM_EU, and LSTM_BSAM equals LSTM_BM_SAM. 21	21
2.4 Comparison of different classification results for the Salinas image (%), where LSTM_PEU equals LSTM_PM_EU, LSTM_PSAM equals LSTM_PM_SAM, LSTM_BEU equals LSTM_BM_EU, and LSTM_BSAM equals LSTM_BM_SAM.....	24
2.5 Average training time (min) of LSTM models.....	28
3.1 Class Codes and Sample Sizes for Pavia University Image .....	51
3.2 Class Codes and Sample Sizes for Salinas Image .....	54
3.3 Class Codes and Sample Sizes for Indian Pines Image .....	55
3.4 Classification results (in units of %) in Pavia University Image .....	61
3.5 Classification results (in units of %) in Salinas Image .....	63
3.6 Classification results (in units of %) in Indian Pines Image.....	64
3.7 Numbers of segments obtained using different scale parameter values, for each image	68
3.8 Average computational time cost (seconds) for sequential feature extraction from single line (row) .....	69
4.1 Information of acquired images. “TM” is Thematic Mapper, and “OLI” represents Operational Land Imager .....	79
4.2 Number of pixels for ground reference data .....	80
4.3 Classification results (in units of %) in the 1991-pair image.....	84

4.4	Classification results (in units of %) in the 1993-pair image.....	84
4.5	Classification results (in units of %) in the 1996-pair image.....	86
4.6	Classification results (in units of %) in the 1998-pair image.....	86
4.7	Classification results (in units of %) in the 2009-pair image.....	86
4.8	Classification results (in units of %) in the 2018-pair image.....	87
4.9	Classification accuracies of different models using different parameters .....	88
5.1	Object detection result on sub-image level using the entire testing dataset .....	110
5.2	Object detection result on geographical coordinate level using the entire testing dataset.....	112
5.3	Object detection results on two different datasets. "A" represents the dataset with more isolated fallen trees. "B" is the dataset with more overlapping fallen trees. "IMG" denotes the accuracy evaluation on sub-image coordinates. "GEO" is the accuracy evaluation on geographical coordinates .....	113
5.4	Accuracy assessment on the dataset with more overlapping fallen trees using clusters created from detected results.....	117

## 1. INTRODUCTION

Remote sensing (RS), a critical technology for large-scale-monitoring Earth-observing systems (EOS), plays an important role in Earth science and other related fields where physical, biological, and chemical properties of the Earth can be characterized in a non-contact manner. Utilizing advanced geospatial technology is a common strategy to analysis RS images by taking advantaged of great computational capability. Geospatial artificial intelligence (GeoAI) integrates data mining, machine learning, deep learning, and high-performance computing to obtain knowledge from geospatial datasets [1]. Due to the intrinsic similarity between conventional digital images and videos exploited in computer vision, RS is attracting more attention regarding the utilization of deep-learning models, which were proposed in the computer science community and are considered to be a key component of GeoAI. Various applications of deep learning in remote sensing data analysis have been proposed with promising performance compared with other traditional machine learning models.

One of the most widely used techniques to analyze land-cover information utilizing RS images is pixel-level classification, where each individual pixel will be classified with a semantic label by employing machine learning (ML) algorithms. In recent decades, with the tremendous developments in hardware and software, computational capabilities have improved dramatically, which has facilitated the development of deep-learning algorithms derived from traditional ML. Among various deep learning models, recurrent neural networks (RNNs), which are able to process sequential inputs by utilizing a series of internal state, attracts more attention for the purpose of handling multi-temporal RS image analysis due to their intrinsic recurrent structure. However, their application on single-image classification still needs more investigations, especially from the perspective of sequential feature extraction. There are two main strategies to utilize RNNs: 1) utilizing the spectral feature vector of the target pixel on its own; and 2) extracting pixels that are similar to the target pixel and construct its sequential feature. To address this limitation, we propose a similarity measurements-based sequential feature extraction method for single RS image classification using



long short-term memory (LSTM), a special class of RNN, based on the aforementioned second strategy. For a given pixel the proposed framework utilizes the spectral information of those pixels collected from the whole image instead of the individual spectral vector of its own. The main idea is for a target pixel, we calculate similarity using all other pixels and only select first several similar pixels to construct sequential feature. And its classification performance on two standard datasets demonstrates its effectiveness compared with other benchmark algorithms.

However, the computational time cost of the aforementioned approach is a critical issue as all pixels in that single RS image need to be considered during similarity measurement for every pixel. That brings more difficulties, especially for processing large-scale RS image. Therefore, building upon the previous work, we improve that model by adding segmentation map as an additional criteria for shrinking searching range from the whole image to selected segments. Specifically, two-phase similarity measurements are designed. Similarity measurements are first applied at the segments level in order to select the similar segments from the whole image; and within those selected similar segments, similar pixels are then selected to construct sequential features in the same manner as what we proposed in previous work. Three different sequential feature extraction strategies are developed, where local and non-local segments are considered in a separate and combined manner, respectively. Experimental results obtained from three benchmark hyperspectral RS images and large-scale satellite images illustrate that the proposed approaches achieve promising classification performance with lower computational time cost.

Besides classification, another application of deep learning is object detection, where interest of object is allowed to be identified and localized in an image or video. In this study, our focus is to detect fallen trees from large-scale aerial photos. Therefore, we develop a framework to automatically create image and annotation datasets that can be utilized as input data for deep learning-based object-detection models. Quantitative and qualitative classification results illustrate that, for isolated fallen trees, our model achieves promising results, considering a recall index. However, many false positive detection results are generated as well. Then we further investigate the influence of those fallen trees located on the boundary of extracted sub-image and find that using geographical

coordinates to calculate accuracy metrics can achieve better results. Furthermore, we also explore the issue of overlapping fallen trees and observe that overlapping fallen tree introduce more detection errors wwhich can be alleviated to some extent by considering fallen-tree clusters. However, detecting overlapping fallen trees is still a challenging task, especially for large-scale datasets collected from the real world.

## 2. HYPERSPECTRAL IMAGE CLASSIFICATION USING SIMILARITY MEASUREMENTS-BASED DEEP RECURRENT NEURAL NETWORKS<sup>1</sup>

### 2.1 Introduction

Hyperspectral remote-sensing images (HSI) can entail both abundant spectral and spatial information, which generally provides enhanced capability of distinguishing different objects from one another, relative to multispectral images, and play an important role in a variety of research domains, such as precision agriculture [2], land-use monitoring [3][4], change detection [5][6], and environment measurements [7]. For such subfields, classification is a critical technology, where each pixel in an HSI will be classified with a pre-defined label, and encouraging achievements have been produced from different methods.

Based on the acquisition of training samples, HSI classification frameworks can be divided into three types: unsupervised, semi-supervised, and supervised classification. Considering the availability of training samples and classification performances, supervised methods are investigated the most and are typically selected as benchmark algorithms, such as k-nearest-neighbor [8][9][10], support vector machines (SVM) [11][12][13], sparse representation [14][15][16][17], and artificial neural networks (ANNs) [18][19]. Since acquiring sufficient labeled samples in the field can be quite difficult and time-consuming, semi-supervised methods [20][21][22][23][24] and tensor-based methods [25][26][27], which can obtain satisfying results only based on a limited number of training samples, also warrant many investigations. To overcome the shortcoming of limited training samples, researchers have also found that spatial information can be employed as crucial complementary and supportive features in spectral-spatial feature-combination frameworks to improve classification performance [28][29][30]. Various spatial feature-extraction methods have been proposed for HSI classification; however, such kinds of spectral-spatial features still refer to low-level features [31], and the classification performances arising from spectral-spatial methods

---

<sup>1</sup>Reprinted with permission from “Hyperspectral Image Classification Using Similarity Measurements-Based Deep Recurrent Neural Networks ” by Andong Ma, Anthony M. Filippi, Zhangyang Wang, and Zhengcong Yin, 2019. *Remote Sensing*, 11(2), 194, ©2019 MDPI.

can be easily affected by the curse of dimensionality problem and the difficulty in discriminating among some classes. Therefore, methods for extracting more robust and representative features from the HSI image itself still need to be investigated.

In the most recent decade, deep learning (DL) has demonstrated marked effectiveness and robustness on large datasets. Considering the inherent deep architecture, which can be viewed as an efficient feature-extraction model, from low-level feature to the highly-abstract level, DL algorithms are always employed as end-to-end classifiers for both pixel- and image-based classification tasks in remote-sensing communities [32]. Chen et al. proposed a stacked autoencoder (SAE)-based HSI classification framework [33], and it was the first paper on DL applications in HSI processing. During unsupervised feature extraction, spatial information was incorporated in order to enhance the classification performance. The authors have also investigated the performance of deep belief network (DBN), another unsupervised deep feature learning method for HSI classification [34]. Additionally, even considering the relatively limited availability of ground-reference or other training data, supervised DL algorithms have also been well-explored and have posted generally impressive results. Convolutional neural networks (CNNs) can be viewed as a milestone in the DL epoch since its first successful application in target detection [35]. CNNs are similar to conventional ANNs in that they are all composed of neurons, activation functions, and loss functions for optimization. The most significant development pertaining to CNNs is that with CNNs, 2-dimensional (2D) images can be fed directly into the network architecture via the input layer by introducing convolutional layers, instead of transforming 2D images to one-dimensional (1D) vectors. Such a property makes image-based processing more efficient and straightforward, and applying a convolutional layer can utilize spatial contextual information with a specific receptive domain. However, regarding pixel-based processing of HSIs, CNNs cannot be employed directly due to its 2D filter-processing characteristic, and preprocessing is needed, including patch extraction. Hu. et al. [36] applied a 1D CNN (1DCNN) for pixel-based HSI classification, where a single pixel is considered as a 2D image whose height is equal to 1. Makantasis et al. [37] proposed a 2D CNN (2DCNN)-based HSI classification method where randomized principal com-

ponent analysis (R-PCA) was conducted as a dimensionality reduction to reduce the number of 2D convolutional filters. Li et al. [38] investigated the effectiveness of a three-dimensional (3D) CNN (3DCNN)-based HSI classification technique, where spectral and spatial features were simultaneously exploited without any dimensionality reduction pre-processing.

Recurrent neural networks (RNNs) [39], another novel DL architecture with outstanding adaptability for handling sequence data has recently achieved promising performance, particularly for natural language processing (NLP) [40][41]. Since multi-temporal information can be obtained conveniently with the development of modern satellite remote-sensing technology, RNNs and an improved type of RNN, referred to as long short-term memory (LSTM) [42], were explored to extract temporal features from multiple images. Ienco et al. [43] utilized RNN and LSTM to perform land-cover classification on multi-temporal satellite images. In [44], Sharma et al. proposed a patch-based RNN framework incorporating both spectral and spatial information within a local window to classify Landsat 8 images. Furthermore, single-image-based RNN methods are also applied to HSIs. Mou et al. [45] proposed a novel RNN-based HSI classification algorithm by using a parametric rectified hyperbolic tangent function (*PRetanh*). In this framework, each individual pixel in the HSI can be regarded as one sequential feature for the RNN input layer. Wu et al. [46] investigated the combination of CNN and RNN layers on the spectral feature domain and employed the convolutional RNN (CRNN) model for HSI classification. The utilization of a CNN can extract patch-level local invariant information among spectral bands, which provides spatial contextual features for the following RNN layers. Shi et al. [47] proposed another strategy to design the sequential data in RNN model instead of taking spectral vector from all bands as one sequential data, but taking advantage of spatial neighbors. For this method, local spectral-spatial features were first extracted by exploiting a 3DCNN on a local image patch, and then sequences were built based on an eight-directional construction.

Although the aforementioned RNN-based DL models have significantly contributed to HSI processing efforts, there are still some critical problems that need to be addressed. The first issue is the limitation of training sample. Acquiring sufficient labeled training data for HSI classifica-

tion is often difficult and time consuming. Moreover, satisfactory DL-based classification accuracy has always relied upon very large sets of training samples. Therefore, obtaining convincing HSI classification results by utilizing limited training data for DL models is a challenging task. However, unlabeled samples, which are relatively easier to acquire than labeled samples, have already been investigated for HSI classification purposes under semi-supervised classification frameworks [20][21][22][23]. Such investigations illustrate the potential effectiveness of unlabeled data for such a purpose. Another critical issue involves the construction of sequential data for the RNN model. In [45][46], the respective authors analyzed the HSI from the perspective of a sequential point, meaning that each pixel is considered to be a data sequence since all pixels in the HSI are sampled densely from the entire spectrum, and they are expected to have dependencies between different bands. Nonetheless, such dependencies still need to be explored in order to more fully exploit the integrity of the full spectral signature. In order to distinguish different classes, it is frequently advantageous to utilize the information encapsulated within the entire reflectance spectrum, as is the case with many conventional classification methods. Furthermore, exploiting the spectral feature directly in the RNN model will introduce more parameters that need to be computed and optimized in the training step.

In this paper, we propose a novel LSTM-based HSI classification framework with spatial similarity measurements (SSM), which is inspired by [48], where the LSTM model and the spatial location are combined simultaneously. First, the sequential feature for each pixel is constructed by selecting candidates from the whole image based on the similarity between each candidate and the target pixel. This selection method mainly relies upon two different similarity measurements where spectral and spatial information are considered, namely pixel-matching-based (PM) spatial similarity measurements and block-matching-based (BM) spatial similarity measurements, respectively. LSTM entails the significant capability of handling sequential data, and it achieves outstanding performance in NLP. The proposed similarity-measuring strategies provide an innovative framework to extract sequential features for HSI classification by employing all pixels in the entire HSI, regardless of whether the pixel candidates to be a given sequential feature are la-

beled or not. The proposed sequential feature effectively encodes the dependency of the target pixel with regard to its contexts, where the pixel-level spectral similarities and the patch-, or block-level contextual similarities are naturally encoded, respectively. More specifically, LSTM assumes that closer “time steps” (which denote selected pixels/blocks with higher similarity) in general have stronger feature sharing, while also allowing for longer-term dependency that can account for non-local similarity and long-tail effects. The motivation for extending one pixel to its sequential features is essentially to find a new feature embedding of the pixel, which makes reference to other similar pixels. The action of re-ordering those pixels in terms of their similarity measures is to ensure that all obtained sequential features admit “comparable” formats in terms of monotonically-decreasing similarity to the original pixel (i.e., the target pixel, or the given pixel of interest). In this framework, the influence of unlabeled data in an HSI is enhanced compared with conventional supervised-learning methods due to our proposed spatial selection, where any pixel can be selected as a candidate to construct sequential features. In view of global searching based on the whole image, more supportive pixels are incorporated with a wider receptive domain. In addition, spatial contextual information, which has already been utilized to reduce the “salt-and-pepper” phenomenon in remote-sensing/HSI classification [49], is also investigated here in the BM-based framework. Similarity measurements of two pixels is implemented by using the neighboring points of two pixels instead of their spectral feature vectors. Compared with the PM-based method, the BM-based scheme can obtain more typical sequential feature representation combining both spectral and spatial features together. In summary, this is the first study to propose such methods operating over the entire image. This is important because these new, novel methods can incorporate additional information collected throughout the whole image by exploiting unlabeled pixels, instead of utilizing only the limited prior information in the form of labeled pixels. Figure 2.1 illustrates the framework of our proposed method.

The organization of the remainder of this section is as follows: Section 5.2 provides a brief introduction to the original RNN and LSTM models. Section 5.3 describes the proposed LSTM classification framework based on pixel matching and block matching. Experimental results are

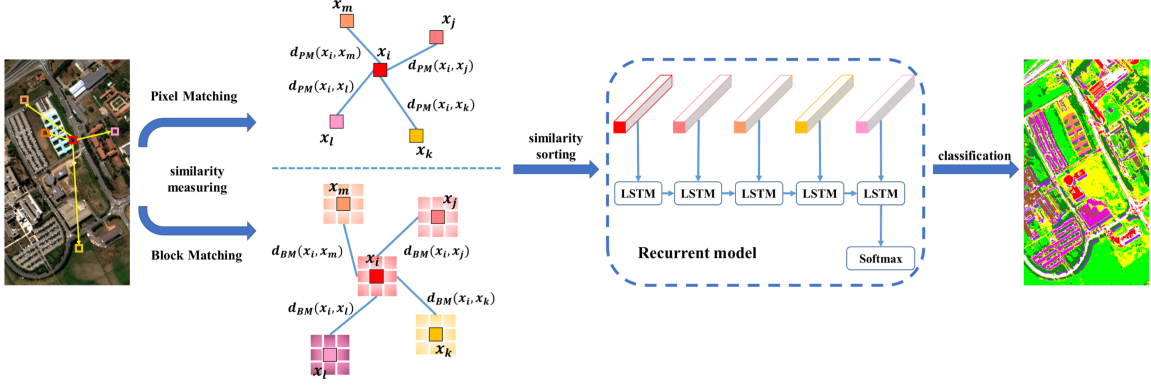


Figure 2.1: Architecture of the proposed methods

discussed in Section 5.4. Section 5.5 presents the conclusions.

## 2.2 Background: RNN and LSTM

### 2.2.1 RNN

The recurrent neural network (RNN) has shown great capability in time-sequence data processing, including NLP [50] and speech recognition [41]. A significant characteristic of a time sequence is that there is typically a strong relationship between a given sample and the previous samples. In the hidden Markov model (HMM), which is a widely-utilized sequence model in language processing, the probability of a specific state depends only on its previous state, instead of all previous states. Let  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t]$  be the sequence data where  $t$  is the label of state.  $\mathbf{x}_1$  represents the data at the first state, and  $\mathbf{x}_t$  represents the data at the  $t$  state. The Markov assumption can be formulated as

$$P(\mathbf{x}_t | \mathbf{x}_1, \dots, \mathbf{x}_{t-1}) = P(\mathbf{x}_t | \mathbf{x}_{t-1}), \quad (2.1)$$

where  $P(\cdot)$  expresses the conditional probability. Compared with HMM, RNN is quite similar with HMM since the computation on the current state relies on the previous state. In contrast to conventional ANNs, the RNN has a circular processing on the sequential data, which means that such processing will be applied on each data instance in the sequence, and the result at each



state relies on the previous state. This circular processing also represents the parameter sharing. Parameter sharing is a prevalent method to control the number of parameters in a DL scheme. Still given  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t]$  as sequential data, the hidden state  $\mathbf{s}_t$  can be represented as

$$\mathbf{s}_t = f_s(\mathbf{W}_{xs}\mathbf{x}_t + \mathbf{W}_{ss}\mathbf{s}_{t-1} + \mathbf{b}_t), \quad (2.2)$$

where  $\mathbf{W}_{xs}$  is the weight matrix from input data to the hidden state, and  $\mathbf{W}_{ss}$  is the weight matrix from the current state to the next state, respectively.  $\mathbf{b}_t$  is the bias variable.  $\mathbf{s}_t$  denotes the hidden state at time step  $t$ , and  $f_s(\cdot)$  represents the nonlinear activation function. The calculation of the output at state  $t$  is quite similar with Equation (3.2) as follows:

$$\mathbf{y}_t = f_y(\mathbf{W}_{sy}\mathbf{s}_t + \mathbf{b}_y), \quad (2.3)$$

where  $\mathbf{W}_{sy}$  is the weight matrix from the hidden state to the output.  $\mathbf{b}_y$  is the bias, and  $f_y(\cdot)$  is the nonlinear activation function.

The hidden state  $\mathbf{s}_t$  can be viewed as the memory of the RNN model, as it is calculated based on the previous state through forward propagation. Meanwhile, sequential data in the previous states are taken into consideration as well. In such forward propagation, some parameters, including three different weight matrices  $\mathbf{W}_{xs}$ ,  $\mathbf{W}_{ss}$ , and  $\mathbf{W}_{sy}$ , are shared across all steps, which is quite different from a traditional neural network. The parameter-sharing scheme reduces the number of trainable parameters, and makes the total computation more efficient.

### 2.2.2 LSTM

In Equation (3.2), the calculation of the hidden state depends on the previous state. However, with the increase in the length of the sequence data, gradient vanishing and gradient exploding will be introduced in this recurrent model due to the forward and backward propagations of weight matrices. To address this issue, long short-term memory was developed with a more sophisticated recurrent neuron. In LSTM, each recurrent neuron can be regarded as a cell state. Similar to the conventional RNN, LSTM also employs the previous state as the input to the current

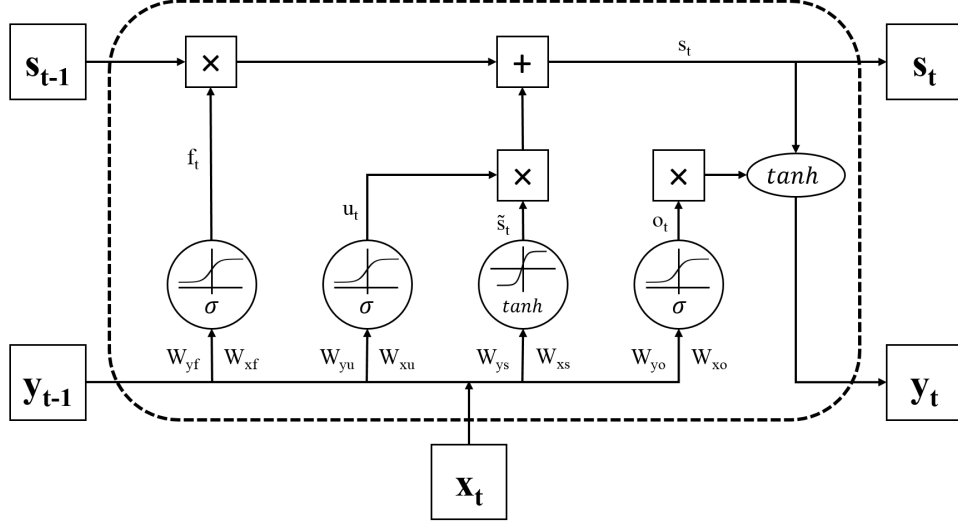


Figure 2.2: The basic structure of an unrolled long short-term memory LSTM unit

state. However, with LSTM, there are three gates, including forget gate, update gate, and output gate, to control the update of the current neuron. Figure 2.2 illustrates the basic structure of the LSTM recurrent unit.

The first part of LSTM is the forget gate which determines if the previous state will be retained or not. Still given  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t]$  as sequential data,  $\mathbf{y}_t$  and  $\mathbf{s}_t$  are the output and the hidden state at step  $t$ , respectively. The common computation for forget gate  $\mathbf{f}_t$  is as follows:

$$\mathbf{f}_t = \sigma_f(\mathbf{W}_{yf}\mathbf{y}_{t-1} + \mathbf{W}_{xf}\mathbf{x}_t + \mathbf{b}_f), \quad (2.4)$$

where the  $\mathbf{W}_{(\cdot)}$  terms denote the weight matrices, and the  $\mathbf{b}_{(\cdot)}$  term is the bias variable.  $\sigma(\cdot)$  is the logistic sigmoid function. The following step is to compute the update gate  $\mathbf{u}_t$  and a new candidate state value  $\tilde{\mathbf{s}}_t$ :

$$\mathbf{u}_t = \sigma_u(\mathbf{W}_{yu}\mathbf{y}_{t-1} + \mathbf{W}_{xu}\mathbf{x}_t + \mathbf{b}_u), \quad (2.5)$$

$$\tilde{\mathbf{s}}_t = \tanh(\mathbf{W}_{ys}\mathbf{y}_{t-1} + \mathbf{W}_{xs}\mathbf{x}_t + \mathbf{b}_s), \quad (2.6)$$

where  $\tanh(\cdot)$  is the hyperbolic tangent function. Then, the new hidden step  $\mathbf{s}_t$  can be updated by

using the aforementioned equations

$$\mathbf{s}_t = \mathbf{s}_{t-1} \times \mathbf{f}_t + \tilde{\mathbf{s}}_t \times \mathbf{u}_t. \quad (2.7)$$

Finally, the output gate and the output of the current neuron yields:

$$\mathbf{o}_t = \sigma_o(\mathbf{W}_{y_o}\mathbf{y}_{t-1} + \mathbf{W}_{x_o}\mathbf{x}_t + \mathbf{b}_o), \quad (2.8)$$

$$\mathbf{y}_t = \mathbf{o}_t \times \tanh(\mathbf{s}_t). \quad (2.9)$$

### 2.3 Spatial Similarity Measurements in LSTM

For the LSTM model, the sequential feature is a critical issue when training LSTM since determining the representative feature will improve classification performance and reduce the training-time cost. In this section, the spatial similarity measurement-based LSTM model will be introduced as a method to construct sequential features. First, two different strategies utilized in SSM will be discussed, named PM-based and BM-based schemes. For each of them, when computing the similarity between pixels, two distance measurements are investigated, which are Euclidean distance (EU) and spectral angle mapper (SAM). Furthermore, we will introduce the way of constructing sequence as the input of LSTM model.

#### 2.3.1 Pixel Matching

Measuring the similarity in the pixel data vectors between different pixels is a common technology in many HSI analysis applications, such as endmember-based analysis [51], manifold learning [52, 53], and graph-based semi-supervised learning [23]. Since HSIs have abundant spectral information, typically entailing hundreds of bands, spectral features collected from all bands have the most discriminative capability to distinguish different ground objects or materials encompassed within the given image, and have been most widely utilized in HSI classification [54]. In the pixel-matching scheme, pairwise spectral similarity measurements are applied for all pixels. Suppose we have HSI data  $\mathbf{X} \in \mathbb{R}^{L \times C \times B}$  with  $L$  rows,  $C$  columns, and  $B$  bands.  $\mathbf{X}$  can be rewritten

as  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N] \in \mathbb{R}^{B \times N}$  with row-major order, where  $N$  is the total number of pixels in the HSI, which equals  $L * C$ . For any pixels  $\mathbf{x}_i$  in  $\mathbf{X}$ , the distances measured between  $\mathbf{x}_i$  and all pixels of  $\mathbf{X}$  will be computed as follows:

$$d_{\text{PM}}(\mathbf{x}_i, \mathbf{X}) = [d(\mathbf{x}_i, \mathbf{x}_1), \dots, d(\mathbf{x}_i, \mathbf{x}_j), \dots, d(\mathbf{x}_i, \mathbf{x}_N)], \quad (2.10)$$

where  $d(\mathbf{x}_i, \mathbf{x}_j)$  denotes the distance between  $\mathbf{x}_i$  and  $\mathbf{x}_j$ , and  $d(\cdot)$  is the distance calculation function. There are multiple methods available to compute pairwise distance. In this paper, Euclidean distance and the spectral angle mapper (SAM) are utilized in the pixel-matching scheme. Euclidean distance is a well-known distance measurement, and its definition is given as follows:

$$d_{\text{EU}}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(x_i^1 - x_j^1)^2 + (x_i^2 - x_j^2)^2 + \dots + (x_i^B - x_j^B)^2} = \|\mathbf{x}_i - \mathbf{x}_j\|. \quad (2.11)$$

The other distance measure adopted in this study is SAM, which is investigated well in endmember-based HSI classification. It can be defined as follows:

$$d_{\text{SAM}}(\mathbf{x}_i, \mathbf{x}_j) = \cos^{-1}\left(\frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}\right). \quad (2.12)$$

In the following text of this paper, we use  $d_{(\cdot)}()$  to represent either the EU or SAM distance calculation function in the pixel-wise measurement. Therefore, Equation (3.10) can be rewritten as follows:

$$d_{\text{PM}}(\mathbf{x}_i, \mathbf{X}) = [d_{(\cdot)}(\mathbf{x}_i, \mathbf{x}_1), \dots, d_{(\cdot)}(\mathbf{x}_i, \mathbf{x}_j), \dots, d_{(\cdot)}(\mathbf{x}_i, \mathbf{x}_N)]. \quad (2.13)$$

### 2.3.2 Block Matching

Although spectral features provide rich, significant information that facilitates discrimination of different ground objects in HSI, classification accuracy based on utilization of spectral features alone is not always satisfactory due to the ‘‘salt-and-pepper’’ phenomenon [49]. Given Tobler’s

First Law of Geography [55], incorporation of spatial contextual information has attracted increasing attention in the research literature in recent years and has exhibited the capability to reduce “salt-and-pepper” noise and improve classification performance, including yielding smoother classification maps. In the current study, we utilize the image patch distance (IPD), proposed in [56], for similarity measurement that considers local spatial information. The IPD method was developed based on the Hausdorff distance [57], also referred to as the Pompeiu–Hausdorff distance. Instead of using spectral features to measure the pairwise similarity between two pixels, spatial neighbors within the respective local window of such two pixels will also be employed. Let  $w$  be the local window size, and the  $s_i$  represents the block neighborhood, where pixel  $x_i$  is centered within the  $w \times w$  spatial window. All block sets  $\mathbf{S}$  can be defined as follows:

$$\mathbf{S} = [s_1, s_2, \dots, s_N]. \quad (2.14)$$

Given  $\forall s_i, s_j \in \mathbf{S}$ , we first calculate the distances between one arbitrary pixel  $x_m$  from  $s_i$  and  $s_j$  and then select the minimum distance as follows:

$$d_{min}(x_m, s_j) = \min_{x_n \in s_j} d_{(\cdot)}(x_m, x_n), \quad (2.15)$$

where  $d(\cdot)$  is the distance-measuring function. Correspondingly, the minimum distance between one arbitrary pixel  $x_m$  from  $s_j$  and  $s_i$  can be computed in the following manner:

$$d_{min}(x_m, s_i) = \min_{x_n \in s_i} d_{(\cdot)}(x_m, x_n). \quad (2.16)$$

Therefore, the definition of block matching between  $s_i$  and  $s_j$  is:

$$\begin{aligned} d_{BM}(s_i, s_j) &= \sum_{m=1}^{w^2} (\max(d_{min}(x_m, s_i), d_{min}(x_m, s_j))) \\ &= \sum_{m=1}^{w^2} (\max(\min_{x_n \in s_j} d_{(\cdot)}(x_m, x_n), \min_{x_n \in s_i} d_{(\cdot)}(x_m, x_n))). \end{aligned} \quad (2.17)$$

Finally, the ultimate BM measurements are defined as:

$$\begin{aligned} d_{\text{BM}}(\mathbf{x}_i, \mathbf{X}) &= d_{\text{BM}}(\mathbf{s}_i, \mathbf{X}) \\ &= [d_{\text{BM}}(\mathbf{s}_i, \mathbf{s}_1), \dots, d_{\text{BM}}(\mathbf{s}_i, \mathbf{s}_j), \dots, d_{\text{BM}}(\mathbf{s}_i, \mathbf{s}_N)]. \end{aligned} \quad (2.18)$$

### 2.3.3 Sequential Feature Extraction

After measuring the pairwise similarity among pixels in the whole image, the sequential feature for each pixel is extracted so that it can be fed into the RNN model directly. Based on the aforementioned two matching schemes, given one pixel  $\mathbf{x}_i \in \mathbf{X}$ , its corresponding matching vector is

$$d_M(\mathbf{x}_i, \mathbf{X}) = [d_M(\mathbf{x}_i, \mathbf{x}_1), \dots, d_M(\mathbf{x}_i, \mathbf{x}_j), \dots, d_M(\mathbf{x}_i, \mathbf{x}_N)], \quad (2.19)$$

where  $d_M(\cdot)$  denotes the pairwise matching function introduced in previous sections. Note that the order of  $d_M(\mathbf{x}_i, \mathbf{X})$  is determined only based on its location in the image. To characterize a representative sequential feature,  $d_M(\cdot)$  is reordered based on the degree of similarity, and then the corresponding pixels are selected as the sequential representation. With the definition given in Equation (3.19), we assume we have one pixel  $\mathbf{x}_i \in \mathbf{X}$ , its reordered  $d_M(\mathbf{x}_i, \mathbf{X})$ , named  $d_{sf}(\mathbf{x}_i, \mathbf{X})$ , is built as follows:

$$d_{sf}(\mathbf{x}_i, \mathbf{X}) = [d_{sf}^{i1}, \dots, d_{sf}^{ij}, \dots, d_{sf}^{iN}], \quad (2.20)$$

where  $d_{sf}^{ij}$  denotes the distance-measuring result between  $\mathbf{x}_i$  and  $\mathbf{x}_j$ .  $d_{sf}(\mathbf{x}_i, \mathbf{X})$  is the ascending sort of  $d_M(\mathbf{x}_i, \mathbf{X})$ . More specifically,  $d_{sf}^1$  is the minimum value among  $d_{sf}(\mathbf{x}_i, \mathbf{X})$ , and  $d_{sf}^2$  is the second-smallest value. During ascending sorting, pixels most similar to  $\mathbf{x}_i$  among all pixels in the whole image will be selected as the sequential representation of  $\mathbf{x}_i$ . Note that not all candidates in  $d_{sf}(\mathbf{x}_i, \mathbf{X})$  will be considered, and the parameter  $l$  is defined to control how many candidates can

be selected or to determine the length of such sequence. The first  $l$  pixels with distance-measuring results from  $d_{sf}^1$  to  $d_{sf}^l$  will be selected, and the first elements of this sequence is  $\mathbf{x}_i$  itself since  $d_M(\mathbf{x}_i, \mathbf{x}_i)$  equals zero, which is the minimum value in  $d_M(\mathbf{x}_i, \mathbf{X})$ . Given the sequence length  $l$ , the final sequential feature of  $\mathbf{x}_i$  can be defined:

$$d_{sf}(\mathbf{x}_i) = [\mathbf{x}_{sf}^{i1}, \dots, \mathbf{x}_{sf}^{ij}, \dots, \mathbf{x}_{sf}^{il}], \quad (2.21)$$

where  $\mathbf{x}_{sf}^{ij}$  represents  $\mathbf{x}_j$ , whose distance-measuring result is located at the  $j$ th place.

## 2.4 Experimental Setup, Results, and Discussion

### 2.4.1 Datasets

In this study, two benchmark HSI datasets were utilized, including Pavia University, and Salinas images, as displayed in Figure 2.3 and Table 2.1. The Pavia University image was collected by the Reflective Optics System Imaging Spectrometer (ROSIS) sensor. It consists of 102 spectral bands, with a spectral range from 430 nm to 860 nm. The image spatial resolution is 1.3 m, and the total image size is  $610 \times 340$  pixels. For the Pavia University image extent, nine (9) classes were considered in the classification experiments. The Salinas image was acquired via the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS), and the image contains  $512 \text{ lines} \times 217 \text{ samples}$ , with the spatial resolution of 3.7 m. After removing 20 noise and water-absorption bands, 204 spectral bands remained for subsequent analysis. The ground-reference data for the Salinas image entails 16 classes.

### 2.4.2 Experimental Design

To evaluate the performance of the proposed SSM-based LSTM methods, three algorithms, including SVM, 1DCNN [36], and 1DLSTM [45], are investigated as baseline algorithms. For SVM, the radial basis function (RBF) is utilized as kernel function, and the parameters of SVM are acquired by cross validation. For the following two deep-learning algorithms, they are 1D-based architectures, where spectral features are fed into the classifier directly. For the 1DCNN, two convolutional layers, two max pooling layers, and one fully-connected (FC) layer are selected due

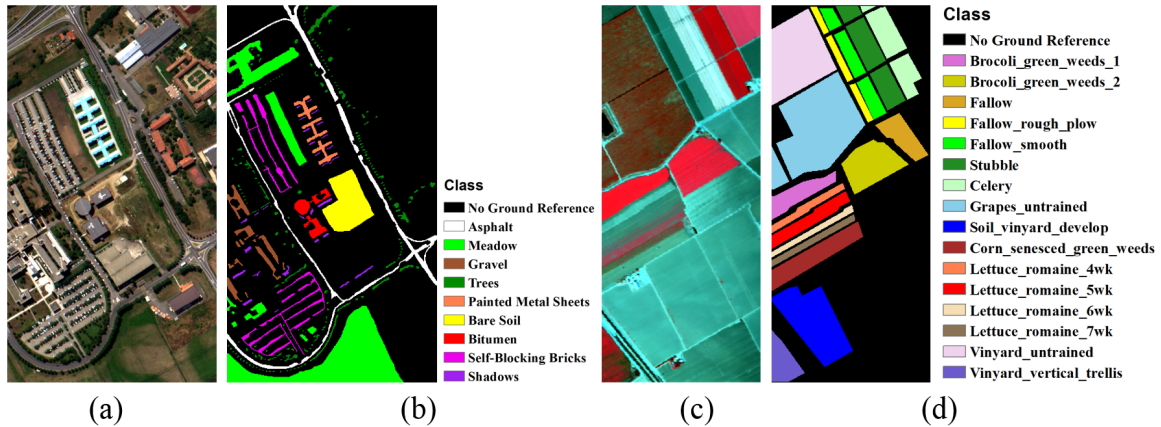


Figure 2.3: False-color image composites and their corresponding ground-reference data. **(a)** false-color composite of Pavia University image bands (R: band 55, G: band 33, and B: band 13); **(b)** ground-reference data (with class legend) for the Pavia University image; **(c)** false-color composite of Salinas image bands (R: band 57, G: band 27, and G: band 17); **(d)** ground-reference data (with class legend) for the Salinas image

Pavia University Image		Salinas image	
Class No.	Name	Class No.	Name
1	Asphalt	1	Brocoli_green_weeds_1
2	Meadow	2	Brocoli_green_weeds_2
3	Gravel	3	Fallow
4	Trees	4	Fallow_rough_plow
5	Painted Metal Sheets	5	Fallow_smooth
6	Bare Soil	6	Stubble
7	Bitumen	7	Celery
8	Self-Blocking Bricks	8	Grapes_untrained
9	Shadows	9	Soil_vinyard_develop
		10	Corn_senesced_green_weeds
		11	Lettuce_romaine_4wk
		12	Lettuce_romaine_5wk
		13	Lettuce_romaine_6wk
		14	Lettuce_romaine_7wk
		15	Vinyard_untrained
		16	Vinyard_vertical_trellis

Table 2.1: Class codes for Pavia University and Salinas images



to the limited available training samples. For the 1DLSTM architecture, three LSTM layers and one FC layer are adopted. Different model structures are implemented based on different images, and the specific parameter settings for the Pavia University and Salinas images are summarized in Table ??, where the convolutional layer is represented as “Conv(number of kernels)-(kernel size)”, maxpooling layer is performed as “Maxpooling-(kernel size)”, and LSTM layer is denoted as “LSTM-(kernel size)”. Regarding the proposed methods, both pixel-matching and block-matching are investigated, and, for each matching scheme, EU and SAM are employed as distance measurements. Therefore, there are four different LSTM-based classification frameworks investigated here, named LSTM\_PM\_EU, LSTM\_PM\_SAM, LSTM\_BM\_EU, and LSTM\_BM\_SAM. For the LSTM structure, we use four recurrent layers and two fully-connected layers, and the length of the sequential feature is 20. The size of the recurrent layers are 32, 64, 128, and 256, respectively, and the size of first fully-connected layer is 50. The second fully-connected layer is applied for the purpose of classification, and its length equals the number of classes. During training of the recurrent model, the batch size is set to 20, and the number of epochs is 500.

In order to evaluate classification performance quantitatively, all ground-reference data for each image is randomly split into training and testing sample sets. In our experiments, we randomly select 200 samples per class as training data, with the remaining ground-reference data used as testing data. Ten replications of the experiments with such random selections were performed, and all classification accuracies were averaged across the ten replications. Furthermore, three other quantitative indicators were also adopted for the evaluation, including overall accuracy (OA), average accuracy (AA), and the Kappa coefficient (Kappa) [58]. The pixel-matching and block-matching experiments were implemented on the Texas A&M High Performance Research Computing (HPRC) system, and the remaining experiments, such as training LSTM models and classification accuracy assessments, were carried out on a local workstation with a 3.2GHz Intel(R) core i7-8700 Central Processing Unit (CPU), and a NVIDIA(R) GeForce GTX 1070 graphics card.

Pavia University Image		Salinas Image	
1DCNN	1DLSTM	1DCNN	1DLSTM
Conv(10)-8	LSTM-32	Conv(8)-12	LSTM-32
Maxpooling-2	LSTM-64	Maxpooling-2	LSTM-64
Conv(10)-8	LSTM-128	Conv(8)-12	LSTM-128
Maxpooling-2		Maxpooling-2	
FC layer-9		FC layer-16	

Table 2.2: Parameter settings for 1D-CNN and 1D-LSTM, where the convolutional layer is represented as “Conv(number of kernels)-(kernel size)”, maxpooling layer is performed as “Maxpooling-(kernel size)”, and LSTM layer is denoted as “LSTM-(kernel size)”

### 2.4.3 Classification Results: Pavia University Image

The first set of experiments is conducted on the Pavia University image. The quantitative results are shown in Table 3.2, where values in bold are the highest class-specific accuracies and the standard deviations are also presented, which are calculated based on ten OAs obtained from the aforementioned ten (10) experimental replications. The classified images are displayed in Figure 2.4 for qualitative analysis, which are obtained from the fifth trial. As shown in Table 3.2, the block-matching-based method LSTM\_BM\_SAM achieved best performance, with 96.20% OA, 94.65% AA, and 94.91% Kappa. For the first three benchmark algorithms, the highest OA (i.e., 84.45%) is obtained from 1DCNN. Regarding our newly-proposed pixel-matching-based LSTM frameworks, the OA of LSTM\_PM\_SAM is 84.56%, exhibiting limited improvement relative to SVM, 1DCNN, and 1DLSTM, and the classification performance of LSTM\_PM\_EU even decreases relative to 1DCNN and 1DLSTM. However, after incorporating spatial information via similarity measurements, LSTM\_BM\_EU and LSTM\_BM\_SAM obtain marked improvements over all non-block-matching methods, with 95.96% and 96.20% OA, respectively. Within each matching method, the performance of SAM is always better than that of the Euclidean distance. Regarding the AA of each class, class 7 (Bitumen, Red) is more difficult to discriminate compared with other classes

due to the mixed spectral features. The proposed LSTM\_BM\_SAM improves the original SVM OA by more than 35%.

From the classification maps shown in Figure 2.4, marked improvements in classification performance are visually apparent. In Figure 2.4b–f, “salt-and-pepper” noise is still obvious due to the lack of incorporation of spatial contextual information in the classification. Within the red-rectangle annotation, many class 2 (Meadow, Bright Green) pixels are misclassified as class 6 (Bare Soil, Yellow), and class 3 (Gravel, Brown), as shown in Figure 2.4b–f. However, the classification maps derived from LSTM\_BM\_EU and LSTM\_BM\_SAM (Figure 2.4g,h) are spatially smooth and generally correctly classified, where most discrete, spurious/misclassified points are eliminated if they are located within an otherwise homogeneous area. Therefore, combining spatial contextual information can yield marked alleviation of image misclassification. Similar to what is observed within the red-rectangle annotation, more accurate and homogeneous classification results can be achieved within the red-circle annotation as well. Such results demonstrate the validity and capability of combining spatial and spectral features together when measuring the similarity between two pixels, and the effectiveness of constructing a sequential feature for a specific pixel based on such similarity between that target pixel itself and candidates from the whole image.

#### **2.4.4 Classification Results: Salinas Image**

For the Salinas image, the results are similar to those attained and described in Section 2.4.3. The quantitative results are shown in Table 3.3, where, again, values in bold are the highest class-specific accuracies. The OAs of the SAM-based method are lower than those associated with its corresponding Euclidean distance-based method, and the performance of the block-matching strategy is always better (more accurate) than that of the pixel-matching scheme, where spatial contextual information is ignored. The best classification performance is still obtained from LSTM\_BM\_SAM, with OA = 90.63%, AA = 93.95%, and Kappa accuracy = 89.55%. Class 15 (Vineyard\_untrained, Violet) is the class with the lowest accuracy due to the high spectral and thematic similarity between this vineyard class and other grape fields, and the best classification result for this class is acquired from 1DCNN (among SVM, 1DCNN, and 1DLSTM), with 59.34% accu-

Class No.	SVM	1DCNN	1DLSTM	LSTM_PEU	LSTM_PSAM	LSTM_BEU	LSTM_BSAM
1	97.52±0.21	96.53±0.57	95.76±0.65	85.51±3.06	94.06±0.72	98.76±0.44	<b>98.78±0.39</b>
2	95.77±0.30	94.78±1.55	94.13±0.93	94.11±1.40	96.01±1.08	<b>99.08±0.25</b>	98.84±0.29
3	65.57±3.41	68.93±3.65	64.29±3.02	70.41±2.27	66.61±3.68	<b>90.22±2.28</b>	89.22±2.63
4	71.27±7.38	76.62±7.92	75.71±4.95	62.69±6.28	81.58±4.77	92.97±2.19	<b>94.70±1.71</b>
5	95.50±1.55	98.50±0.86	97.73±1.31	98.51±0.67	95.57±2.80	98.99±1.18	<b>99.42±0.64</b>
6	59.51±6.74	63.62±11.03	61.12±7.62	69.71±8.18	64.39±8.23	88.65±4.93	<b>90.65±3.92</b>
7	52.10±0.93	66.87±4.86	65.74±3.62	63.76±4.45	56.69±4.42	<b>89.72±4.25</b>	88.47±6.60
8	84.27±1.13	83.54±1.73	81.48±1.70	82.37±1.59	80.50±1.01	<b>93.32±2.46</b>	93.27±1.28
9	99.92±0.11	99.65±0.31	99.72±0.35	93.74±2.05	95.99±5.06	<b>99.09±0.62</b>	98.49±1.75
OA	82.12±1.79	84.45±3.01	83.41±2.66	82.70±1.73	84.56±2.41	95.96±1.01	<b>96.20±0.57</b>
AA	80.16±0.96	83.23±1.71	81.74±1.47	80.09±1.19	81.27±1.72	94.53±0.98	<b>94.65±0.64</b>
Kappa	76.98±2.12	79.79±3.53	78.42±3.13	77.40±2.05	79.86±2.89	94.60±1.31	<b>94.91±0.75</b>

Table 2.3: Comparison of different classification accuracy results for the Pavia University image (%), where LSTM\_PEU equals LSTM\_PM\_EU, LSTM\_PSAM equals LSTM\_PM\_SAM, LSTM\_BEU equals LSTM\_BM\_EU, and LSTM\_BSAM equals LSTM\_BM\_SAM

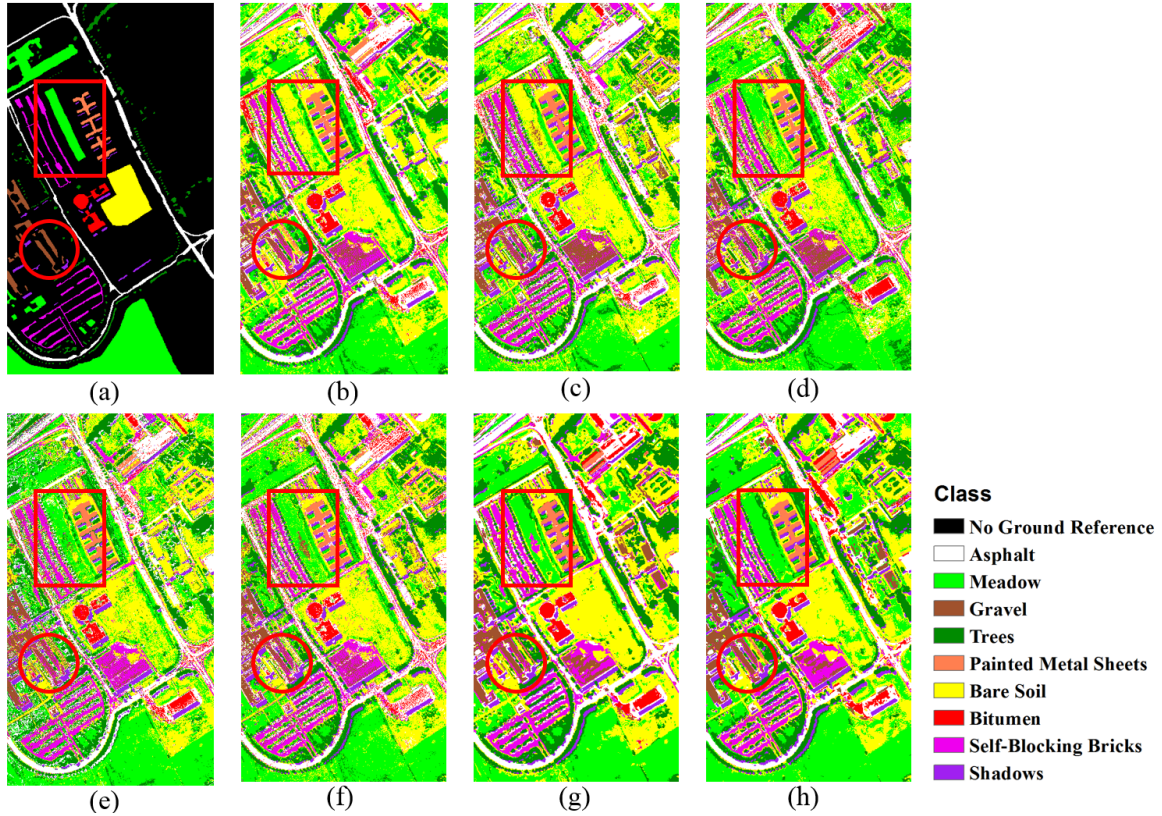


Figure 2.4: Classification maps for the Pavia University image from the fifth trial: **(a)** ground-reference map; **(b)** SVM, with OA = 80.04%; **(c)** 1DCNN, with OA = 78.32%; **(d)** 1DLSTM, with OA = 83.72%; **(e)** LSTM\_PM\_EU, with OA = 83.81%; **(f)** LSTM\_PM\_SAM, with OA = 86.78%; **(g)** LSTM\_BM\_EU, with OA = 93.18%; and **(h)** LSTM\_BM\_SAM, with OA = 96.01%. The red-rectangle and red-circle annotations represent sample areas of interest, discussed in the text

racy. However, LSTM\_BM\_EU and LSTM\_BM\_SAM markedly improve classification accuracy by utilizing spatial features, where class 15 accuracies increase by 7.99% and 10.51%, respectively, compared with the 1DCNN result.

Manual interpretation of the classification maps shown in Figure 2.5 enables us to determine why class 15 (Vineyard\_untrained) entails the lowest OA. Note that many class 15 (Vineyard\_untrained, Violet) pixels are misclassified as class 8 (Grapes\_untrained, Baby Blue). Pixel-matching-based methods, including LSTM\_PM\_EU and LSTM\_PM\_SAM, still yield much discrete noise within the red-circle annotation in Figure 2.5. However, LSTM\_BM\_EU and LSTM\_BM\_SAM produce more homogeneous and smoothed classification results for the Grapes\_untrained class, especially within the red-circle annotation, illustrated in Figure 2.5g,h. Within the red-rectangle annotation, we can see that it is difficult to classify class 10 (Corn\_senesced\_green\_weeds, Brown), for example, and many pixels are misclassified in Figure 2.5b–f. However, such misclassification is markedly minimized when applying block-matching-based methods, i.e., LSTM\_BM\_EU (Figure 2.5g) and LSTM\_BM\_SAM (Figure 2.5h). Such improvement illuminates the advantages of utilizing spatial contextual information when measuring the pixel-wise distances, especially when it is challenging to discriminate between two classes with very similar spectral features.

#### 2.4.5 Parameter Sensitivity Analysis

The influence of different parameter values associated with our proposed methods is investigated in this section, including the length of sequential feature  $l$ , and the size of the local window  $w$ , utilizing block-matching-based methods. The effect of varying the value of  $l$  is tested on LSTM\_PM\_EU, LSTM\_PM\_SAM, LSTM\_BM\_EU, and LSTM\_BM\_SAM, and the effect of varying the value of  $w$  is tested using LSTM\_BM\_EU and LSTM\_BM\_SAM.

For the first parameter,  $l$ , five different lengths (10, 20, 30, 40, and 50) are investigated, while the window size utilized in LSTM\_BM\_EU and LSTM\_BM\_SAM is fixed at 5. The results are shown in Figure 2.6. Regarding the Pavia University data, the best performances for those four proposed methods are obtained with different sequential feature lengths (Figure 2.6a). Sequential feature lengths of 20 and 10 result in the highest classification OAs for LSTM\_PM\_EU

Class No.	SVM	1DCNN	1DLSTM	LSTM_PEU	LSTM_PSAM	LSTM_BEU	LSTM_BSAM
1	96.84±1.18	99.12±1.52	94.18±10.04	98.61±3.93	99.40±0.89	95.78±11.30	<b>99.86±0.22</b>
2	98.79±0.13	98.79±0.69	98.69±0.78	98.75±1.08	99.39±0.28	<b>99.60±0.16</b>	99.34±0.51
3	85.11±1.38	95.53±1.32	88.68±7.32	92.89±3.68	95.70±1.07	96.11±1.25	<b>96.24±1.10</b>
4	97.44±0.18	97.94±0.67	97.20±1.02	98.33±0.74	98.45±0.68	<b>98.66±0.95</b>	97.90±1.70
5	95.03±0.85	97.20±2.76	98.07±1.65	96.26±7.50	97.63±1.57	98.75±1.65	<b>98.83±0.68</b>
6	99.79±0.11	99.77±0.21	98.98±1.16	99.64±0.44	99.45±0.87	99.69±0.36	<b>99.71±0.33</b>
7	98.63±0.44	99.35±0.62	98.81±0.91	99.37±0.53	99.11±0.76	<b>99.40±0.43</b>	99.38±0.51
8	76.70±1.31	83.99±4.02	76.24±5.90	76.97±2.91	76.50±3.48	83.04±1.37	<b>85.66±3.00</b>
9	99.12±0.04	99.02±0.29	98.03±1.46	98.78±0.28	98.66±0.33	99.20±0.41	<b>99.43±0.27</b>
10	81.91±1.58	85.89±2.09	84.90±1.94	85.94±4.19	88.65±1.20	<b>94.16±1.58</b>	91.00±2.13
11	69.51±1.00	82.23±6.71	81.23±13.79	76.81±8.06	83.24±4.33	<b>86.00±3.27</b>	83.49±4.40
12	93.33±0.34	96.82±1.05	87.58±9.57	97.09±1.33	97.26±1.03	<b>98.31±1.36</b>	98.26±1.67
13	92.67±0.65	94.15±2.62	90.12±4.05	96.07±2.23	95.27±2.27	95.89±2.78	<b>97.33±1.98</b>
14	89.68±2.19	89.80±3.78	84.77±12.61	87.66±5.95	88.47±5.34	<b>92.50±3.95</b>	90.75±3.77
15	56.78±2.39	59.34±12.38	56.88±8.13	60.79±6.72	63.62±5.58	67.33±2.82	<b>69.85±3.21</b>
16	95.59±1.18	98.06±0.44	93.84±1.59	96.53±2.14	96.63±1.12	<b>98.03±1.07</b>	96.19±3.79
OA	84.75±0.62	85.99±4.14	84.07±2.90	86.35±2.07	87.53±0.95	89.90±0.43	<b>90.63±0.61</b>
AA	89.18±0.23	92.31±0.96	89.26±3.02	91.28±1.40	92.34±0.56	93.90±0.71	<b>93.95±0.55</b>
Kappa	76.98±2.12	84.45±4.49	82.26±3.19	84.78±2.26	86.07±1.05	88.72±0.48	<b>89.55±0.68</b>

Table 2.4: Comparison of different classification results for the Salinas image (%), where LSTM\_PEU equals LSTM\_PM\_EU, LSTM\_PSAM equals LSTM\_PM\_SAM, LSTM\_BEU equals LSTM\_BM\_EU, and LSTM\_BSAM equals LSTM\_BM\_SAM

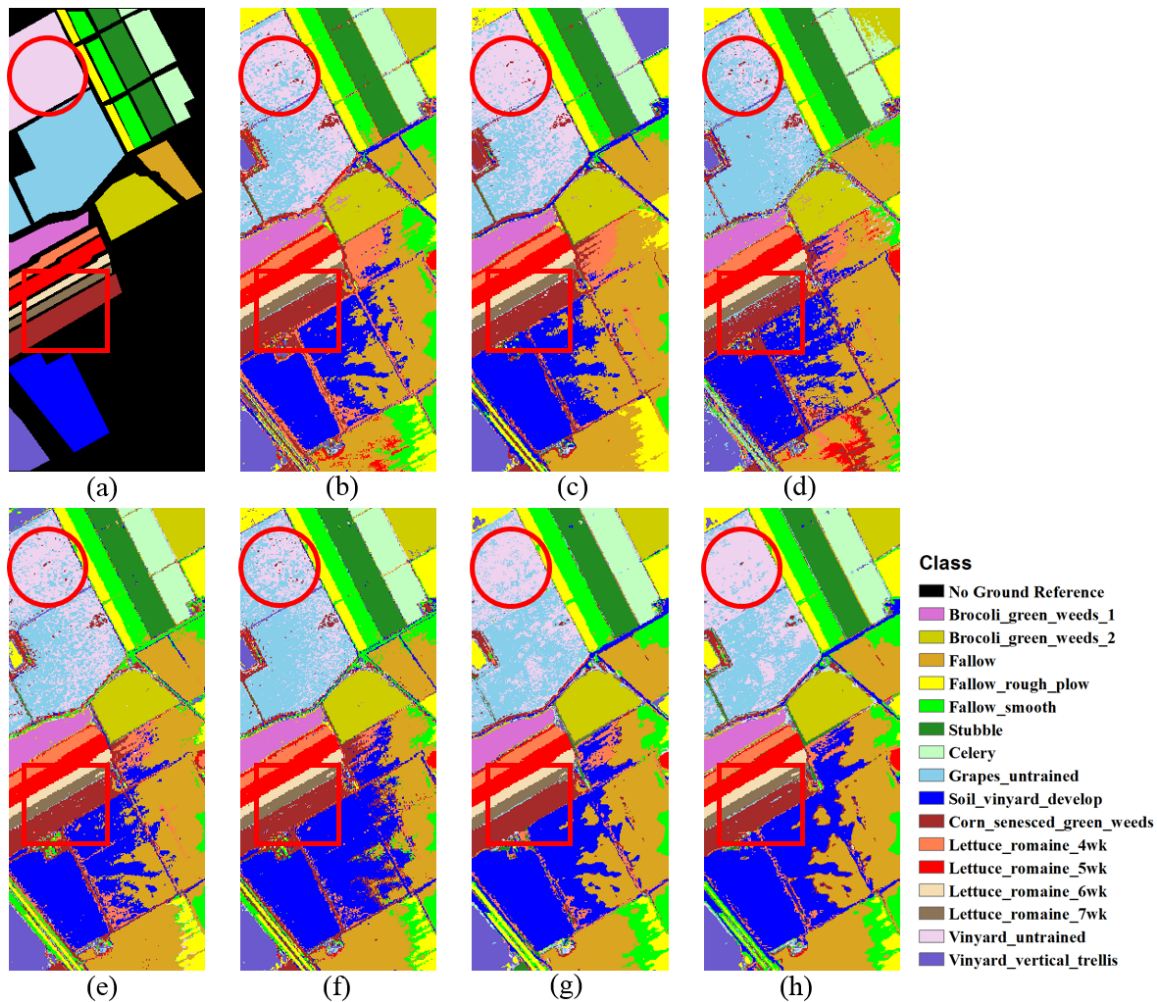


Figure 2.5: Classification maps of Salinas image from the fifth trial: (a) ground-reference map; (b) SVM, with OA = 83.38%; (c) 1DCNN, with OA = 87.00%; (d) 1DLSTM, with OA = 86.85%; (e) LSTM\_PM\_EU, with OA = 86.21%; (f) LSTM\_PM\_SAM, with OA = 87.13%; (g) LSTM\_BM\_EU, with OA = 90.02%; and (h) LSTM\_BM\_SAM, with OA = 90.72%. The red-circle and red-rectangle annotations represent sample areas of interest, discussed in the text



(82.70%) and LSTM\_PM\_SAM (86.68%), respectively. For the two block-matching-based methods, the highest OA for LSTM\_BM\_EU is 96.54% (when  $l$  is 50), and setting  $l$  to 20 yields the highest-accuracy result for LSTM\_BM\_SAM. For the pixel-matching-based algorithms, the classification performance of the Euclidean-distance measure is always better than that of SAM. Furthermore, selecting a smaller sequential length (i.e., 10 or 20) is suitable for these two methods. Regarding the block-matching-based methods, Euclidean distance performs better than SAM, except for  $l$  is 20. Smaller sequential lengths used with SAM result in higher OAs, but such lengths are not suitable when using the Euclidean distance measure. Nevertheless, the difference in the resultant classification accuracies when employing these two distance measurements in the block-matching scheme is smaller than what it is in pixel-matching scheme. Moreover, the standard deviations for those four methods, across the five sequential lengths, are 1.0020, 1.4280, 0.5342, and 0.4805, for LSTM\_PM\_EU, LSTM\_PM\_SAM, LSTM\_BM\_EU, and LSTM\_BM\_SAM, respectively, which illustrates that, for the Pavia University data, the block-matching scheme is less sensitive than the pixel-matching method to the sequential-length parameter value.

For the Salinas data, the best choices for  $l$  vary depending on the algorithm. As shown in Figure 2.6b, the length of 40 yields highest accuracies in LSTM\_PM\_EU and LSTM\_BM\_SAM. LSTM\_PM\_SAM achieves best OA when utilizing 20 as the sequential length. Length of 30 results in the highest OA in LSTM\_BM\_EU. Different from what we observed from Figure 2.6a, within the pixel-matching-based schemes, SAM always performs better than Euclidean distance except  $l = 20$ . Additionally, smaller length provides a better performance for LSTM\_PM\_SAM ( $l = 20$ ) but not applicable in LSTM\_PM\_EU. For the blocking-matching schemes, SAM is the more robust distance measurement since it performs better at four lengths (10, 20, 40, 50), and obtains the highest accuracy with  $l = 40$ . Euclidean distance only yields better result compared with SAM at the length of 30 and that is the highest OA among all lengths. For the standard deviations of those four methods, they are 0.3658, 0.4739, 0.9334, and 0.8140, respectively, which exhibits that pixel-matching-based methods is less sensitive than block-matching-based ones. However, due to the higher OAs obtained by LSTM\_BM\_EU and LSTM\_BM\_SAM, block-matching schemes are

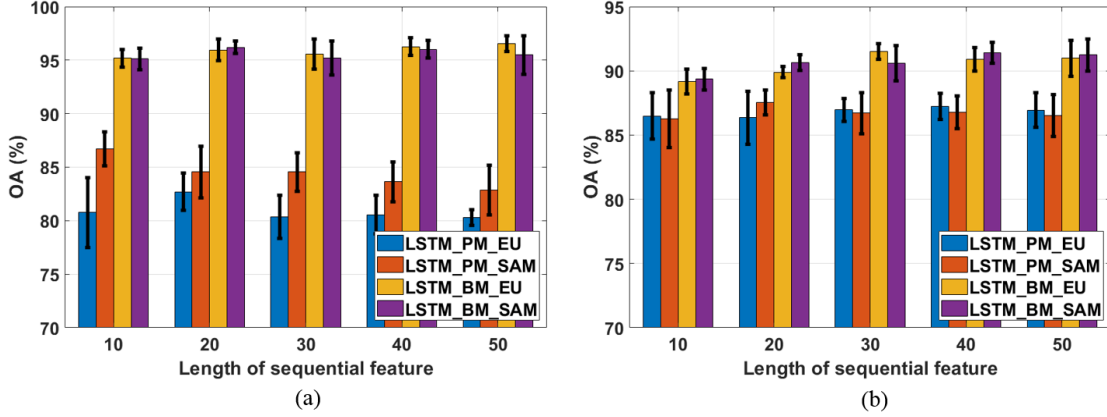


Figure 2.6: Analysis of different sequential feature lengths. (a) OAs based on the Pavia University image data; and (b) OAs based on the Salinas image data

still the applicable methods to classify the Salinas data.

Another investigation of parameter  $l$  is its influence on the training time of the LSTM model since a larger  $l$  will introduce more parameters to be learned in the LSTM model and will result in more processing time. The training times of different approaches are given in Table ???. Those training times are the average values obtained from the 10 replications. Different methods with the same  $l$  have similar training times for both the Pavia University and Salinas images. However, the training time differs when a different  $l$  is applied within one LSTM model. As an example, consider the application of the LSTM\_BM\_SAM to the Pavia University image; the training time is 32.00 min when  $l$  is 10. The training time increases along with utilization of larger  $l$ , and it reaches 142.07 min, which is more than four times the minimum training time consumption. Fortunately, BM-based methods are less sensitive compared with PM-based methods regarding the selection of different  $l$ , which can be obtained from Figure 2.6. To balance the computation time cost and classification performance, choosing a smaller  $l$  (e.g., 10 or 20) is an appropriate strategy for our proposed methods, even though PM-based methods are relatively more sensitive with respect to parameter  $l$ .

For the second parameter,  $w$ , four different window sizes ( $5 \times 5$ ,  $7 \times 7$ ,  $9 \times 9$ ,  $11 \times 11$ ) and two methods (LSTM\_BM\_EU and LSTM\_BM\_SAM) are chosen for the comparison experiments,

LSTM Parameter	Pavia University Image					
	LSTM_PM_EU	LSTM_PM_SAM	LSTM_BM_EU	LSTM_BM_SAM	LSTM_PM_EU	LSTM_BM_SAM
Sequential Feature Length						
10	30.88	34.01	29.11	32.00		
20	51.14	53.25	49.92	53.73		
30	83.39	87.43	77.52	81.63		
40	112.49	115.03	105.31	110.75		
50	145.29	144.98	132.31	142.07		
	Salinas Image					
Sequential Feature Length						
10	29.08	34.19	28.47	28.00		
20	46.27	49.02	44.96	47.64		
30	76.94	85.61	75.83	75.41		
40	110.57	117.78	101.10	102.07		
50	124.06	147.91	128.13	128.90		

Table 2.5: Average training time (min) of LSTM models

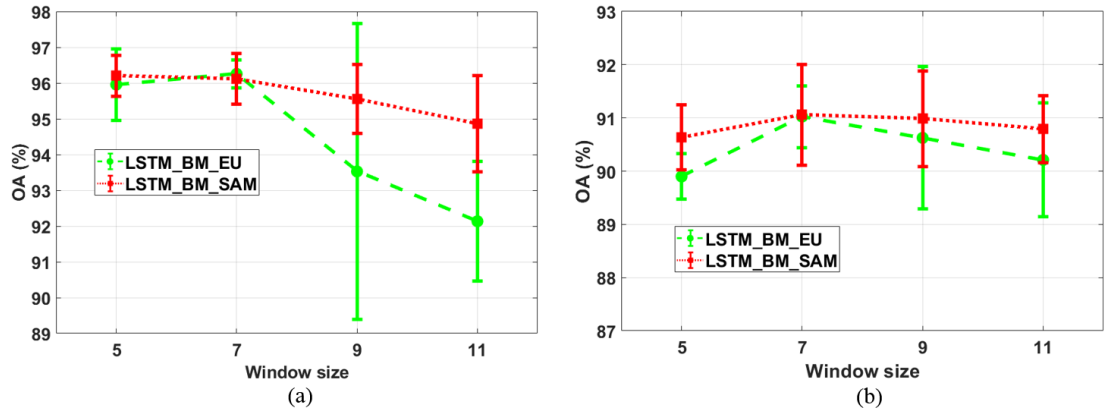


Figure 2.7: Analysis of different window sizes. (a) OAs based on the Pavia University image data; and (b) OAs based on the Salinas image data

where  $l$  is fixed at 20. These results are shown in Figure 2.7. For the Pavia University data-based results (Figure 2.7a), the overall classification accuracy of LSTM\_BM\_SAM is higher than that of LSTM\_BM\_EU. LSTM\_BM\_EU obtains a higher OA only when  $w$  is set at 7, and that is also the best accuracy across all four window sizes. LSTM\_BM\_SAM achieves the best performance with a window size of 5 (the smallest window size considered), and the OA decreases as the window size increases. Regarding the Salinas data-based results, given in Figure 2.7b, the classification accuracies for LSTM\_BM\_SAM are still higher than those for LSTM\_BM\_EU, and both of those two methods realize the most accurate results with a window size of 5. Compared with parameter  $l$ , the optimal value for  $w$  is easier to determine in order to achieve a more accurate classification result. It is predictable that incorporating local spatial contextual information is likely to help better measure the similarity between two pixels, yielding improved classification performance. However, with increasing window size, too many neighboring pixels are included in the calculation, resulting in over-smoothed classification maps, and class spatial boundaries are not preserved. As a consequence, selection of a relatively small window size should introduce sufficient—though not too much—spatial information, leading to higher classification accuracies.

## 2.5 Conclusions

In this paper, we propose a novel LSTM HSI classification framework, where unlabeled data are well-exploited in order to construct sequential features from a single HSI. Instead of using spectral features as the sequential data structure of LSTM, similar pixels collected from the entire image are used to construct the respective sequential features. Specifically, when constructing a sequential feature, the similarity between a target pixel and all other pixels in the image is considered. To better depict the similarity between two pixels, two similarity-measuring strategies—pixel-matching and block-matching—are adopted here, where individual spectral features are utilized in the pixel-matching-based schemes, and both spatial and spectral information are employed in block-matching-based schemes. Such schemes take full advantage of unlabeled data in the HSI, as labeled data are almost always limited in nature and difficult to acquire for HSI classification. Moreover, block-matching-based schemes also consider spatial contextual information in the classification process, and it is demonstrated in this research that such schemes are effective in increasing HSI classification. Our proposed methods produce markedly more accurate results when operating on two well-known, extensively-studied HSI datasets compared with other selected baseline algorithms. Particularly regarding the Pavia University image, the LSTM\_BM\_SAM achieves the best classification performance, with 96.20% OA, which is 11.75% higher than the best result obtained by the three benchmark algorithms, which in this case was 1DCNN, with 84.45% OA. Furthermore, that OA is also higher than those from other three proposed methods (LSTM\_PM\_EU, LSTM\_PM\_SAM, and LSTM\_BM\_EU), with an OA increase of 13.50%, 11.64%, and 0.24%, relative to those respective methods. Additionally, in these experiments, BM-based methods always yield better results compared with their corresponding PM-based methods, which demonstrates the effectiveness and capability of the utilization of spatial contextual information.

Regarding the proposed block-matching method, fixed window sizes are applied for classification. In the future, we will explore adaptive window-size applications, intended to eliminate the phenomenon of over-smoothing in the classified images and to preserve the respective boundaries between different classes. In addition, measuring pixel-wise similarity from the entire HSI more

efficiently still needs to be investigated in future research.

The proposed methods in this study combine similarity measurements and recurrent neural networks, and, although in the present study we focus on encoding spatial contextual information, future work may involve implementing these methods in a temporal context (i.e., in a true multi-temporal remote-sensing context).

### 3. FAST SEQUENTIAL FEATURE EXTRACTION FOR RECURRENT NEURAL NETWORK-BASED HYPERSPECTRAL IMAGE CLASSIFICATION<sup>1</sup>

#### 3.1 Introduction

Remote sensing (RS), a critical technology for large-scale-monitoring Earth observing systems (EOS), plays an important role in Earth science and other related fields where physical, biological, and chemical properties of the Earth can be characterized in a non-contact manner. Hyperspectral remote-sensing images (HSIs), which are collected from a specialized sensor where the spectral width of each band is relatively narrow compared with other optical RS systems, can provide approximately continuous spectra that can be utilized to identify subtle differences among objects with similar spectral reflectances and have been introduced successfully in various sub-fields [59, 60, 61, 62].

For those aforementioned application domains, classification is a widely-used technology for processing and analyzing HSIs, where each pixel is assigned to a predefined semantic label. Supervised classification models are the most common, heavily-investigated methods in the hyperspectral remote-sensing community, including  $k$ -nearest neighbor ( $k$ -NN) [9, 8], random forests (RFs) [63], neural networks [18, 19] support vector machines (SVMs) [11, 13] and sparse representation [14, 64, 65]. Although those algorithms have been demonstrated to have strong capability for HSI classification, obtaining enough training samples for classification purposes is still a challenging task due to the difficulty of labeling using either field data or visual/manual interpretation. Accordingly, how to obtain acceptable or high classification performance with limited training samples is a key issue, which has attracted increasing attention over the past decade.

An intuitive means of addressing this small-sample problem is to use abundant unlabeled data, which can be easily acquired from the HSI. One popular approach is semi-supervised learning,

---

<sup>1</sup>Reprinted with permission from “Fast Sequential Feature Extraction for Recurrent Neural Network-based Hyperspectral Image Classification ” by Andong Ma, Anthony M Filippi, Zhangyang Wang, Zhengcong Yin, Da Huo, Xiao Li, and Burak Güneralp, 2020. *IEEE Transactions on Geoscience and Remote Sensing*, 59(7), Page 5920-5937, ©2020 IEEE.

such as graph-based semi-supervised learning [66, 23], and transductive SVM (TSVM) [21, 67], where limited training samples and abundant unlabeled samples are utilized simultaneously for classification model training. Similarly, to better characterize the capability of unlabeled samples in model training, active learning is another powerful tool to address the small-sample problem, where more informative samples are selected for training [68, 69, 70]. With the help of those informative samples collected from active learning, the classification performance is likely to be boosted in an iterative manner with very limited training samples.

The other shared strategy to address the small-sample problem is not only focusing on the increments in the number of training samples, but to also enhance the capability of feature representation for the limited number of available training samples. Apart from the conventional spectral features, spatial features, which are always taken as the complementary information along with spectral features, can also be obtained simultaneously from HSIs due to the intrinsic properties of images. Given Tobler's First Law of Geography [55], neighboring pixels with similar spectra are more likely to belong to the same class than those located further away. Thus, a multitude of spectral-spatial feature-combination and classification methods have been proposed, including multi-kernel methods [71, 72, 73], multi-feature fusion-based methods [74, 75, 76], and segmentation-based methods [77, 78]. By utilizing spatial features for HSI classification, classification accuracy can be enhanced via minimization of "salt-and-pepper" noise. However, the aforementioned spectral-spatial features are still classified as "shallow," or "low-level," features, and the resulting spectral-spatial features can frequently introduce the curse of dimensionality and an over-smoothing problem, especially for those pixels around the boundaries between different classes. As a result, extracting more representative features constitutes a HSI classification problem that warrants further investigation.

Over the past decade, deep learning (DL), a sub-field of artificial intelligence where "deeper" and highly-abstracted features are extracted and encoded, relative to conventional methods, has attracted increasing attention in the fields of computer vision and image processing. Such algorithms have also been introduced successfully in the remote sensing and HSI communities. Chen



*et al.*[33] proposed a stacked autoencoder (SAE)-based HSI classification scheme, and it is considered to have been the first attempt of applying deep learning to the problem of HSI classification. Afterwards, a great number of DL-based models have been investigated and proposed, including, deep belief networks (DBNs), one-dimensional convolutional neural networks (1D\_CNNs) [36], 2D\_CNNs [37], 3D\_CNNs [38], recurrent neural networks (RNNs) [45], and generative adversarial networks (GANs) [79].

Among the aforementioned well-known DL algorithms, RNN, which has distinguished adaptability and capability of handling sequence data, has been drawing increasing attention for remote sensing, including HSI, classification. Unlike CNNs and other neural network-based models, RNNs are able to process sequential inputs by having a recurrent hidden state whose activation at each step depends on that of the previous step. The manner in which RNN-based models have been utilized in the RS community has been to classify multi-temporal remote-sensing images, given the intrinsic sequential features that exist in such images collected at different times. Ienco *et al.*[43] investigated the utilization of long short-term memory (LSTM), which is an improved type of RNN originally proposed in [42], on the classification of very high spatial-resolution imagery with limited temporal depth. Fig. 3.1 illustrates the fundamental neuron structure of LSTM. In [80], Minh *et al.* proposed LSTM- and gated recurrent unit (GRU)-based classification models for vegetation-quality mapping using synthetic aperture radar (SAR) imagery. However, for the HSI classification task, how to extract such sequential features from an individual image is a critical issue since an individual image itself does not contain multi-temporal features.

To address this issue, many researchers have explored and proposed different solutions. Mou *et al.*[45] developed a deep RNN-based HSI classification framework where a spectral feature vector collected from all bands for each given pixel was employed as its sequential feature for the RNN input layer. Wu *et al.*[46] combined CNN and RNN to extract more discriminative features for classification. The dependencies between spectral bands were captured jointly by utilizing both CNN and RNN. Hang *et al.*[81] proposed a cascaded RNN (casRNN) model to encode the redundant and complementary information of spectral features obtained from a HSI. Instead of

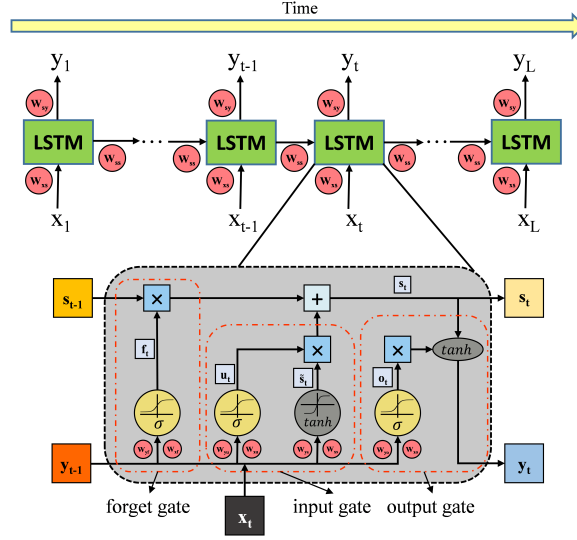


Figure 3.1: Analysis of different window sizes. (a) OAs based on the Pavia University image data; and (b) OAs based on the Salinas image data

utilizing the whole spectral-feature vector as a single sequence for an RNN input layer, this model employed a two-layer RNN. The first-layer RNN was applied to several sub-sequences split from the original spectral vector to eliminate redundancies between adjacent bands, and the second-layer RNN was used for encoding the complementary information between nonadjacent bands. Shi *et al.*[47] also proposed a CNN-RNN-based HSI classification method. Instead of applying a CNN on 1D spectral-feature vectors, as in [46], a 3D\_CNN was utilized to extract spatial-spectral features from each 3D cube, which was generated based on each single pixel and its neighboring pixels. Then sequential features in eight directions were constructed, which were regarded as input features for an RNN model. Zhang *et al.*[82] proposed a novel strategy to employ surrounding pixels within a local window for sequential feature construction, where similarities between those neighboring pixels and the central target pixel were calculated, and neighboring pixels were re-ordered based on calculated similarities in a descending manner.

It is therefore evident that two main strategies are exploited to deal with sequential feature extraction from an individual image. The first implementation is to utilize the spectral feature vector of the target pixel on its own since the spectral (band-to-band) correlation and variability

are similar to the temporal variability of a sequential signal [45]. The other strategy is to extract pixels that are similar to the target pixel and construct its sequential feature. In our previous work [83], to better characterize the sequential feature for each pixel, a similar-pixels selection is implemented across the whole HSI image, where all pixels in an individual HSI are utilized to calculate similarities between them and the target pixel. Then only the first several “similar” pixels are selected to construct the sequence by re-organizing the spectral vectors of those selected “similar” pixels in a descending manner based on their similarities. Even though experimental results illustrate that a promising classification performance is achieved with that proposed method compared with other standard algorithms, the computation time required remains a critical issue, especially for large-scale images (i.e., in this context, images containing a large number of pixels) since all pixels in a HSI need to be considered during similarity measurements involving each target pixel. Thus, the issue of how to develop a more time-efficient sequential feature-construction method within RNN-based models still needs more investigation for a single-HSI classification purpose.

In this article, building upon our previous work [83], we propose an improved single-image-based sequential feature extraction for a LSTM-based HSI classification model. Specifically, similar pixels for one target pixel will not be selected in the range of the whole HSI, but rather from similar segments generated by an object-based segmentation map, which will reduce searching time cost for similar-pixels selection. As a very important image-processing technology, segmentation has been investigated and applied in remote-sensing (including HSI) classification due to its effective capability of extracting relatively homogeneous areas, or segments, that will markedly alleviate “salt-and-pepper” noise compared with pixel-based classification methods [84, 85, 86]. Such relatively homogeneous segments provide an excellent data source that helps to reduce the searching range from the whole-image pixel set to the segments set. For our proposed methods, two-phase similarity measurements are designed: 1) similarity measurements are first applied at the segments level in order to select the similar segments from the whole HSI; and 2) within those selected similar segments, similar pixels are then selected to construct sequential features in the

same manner as what we proposed in [83]. When measuring similarities between different segments, not only spectral features, but also geometry and shape features derived from segments themselves are taken into consideration. Moreover, different similarity-matching schemes, such as pixel-matching (PM) and block-matching (BM) [83], are still under evaluation as part of our proposed methods in the present study in order to investigate the influence of spatial-contextual information on sequential-feature construction and classification performance. Compared with [83], the search range is shrunk from pixel-wise comparisons to segment-wise comparisons first, and pixel-wise comparisons only occur within similar segments. For the specific segment that contains the target pixel, more similar pixels will likely be included in the similarity calculation. In contrast to using a square local window with a fixed kernel size, use of a segment for constructing a sequential feature will translate into better delineations of boundaries, and hence, characterizations of physical ground objects.

The main contributions of this article are summarized as follows:

1) We propose two-phase similarity measurements by utilizing object-based segmentation to facilitate choosing similar pixels from a HSI in a much more time-efficient manner, building upon preliminary research findings reported in [87]. To our knowledge, this is the first time that a segmentation map has been incorporated into two-phase similarity calculations in order to select similar pixels across an entire HSI and build sequential features for the LSTM model. Benefiting from this innovative approach, the algorithms proposed here achieve tremendous improvements in terms of computational time cost compared with our previous work [83]. 2) In the first phase, during the similar-segments selection, both the current segment that contains the target pixel, as well as other similar segments that do not contain the target pixel are investigated in the present study. Often, the local segment provides more “similar” pixels and more robust spatial features than non-local segments, but few studies provide insights on or investigate the effects of non-local spatial-contextual information on similarity calculations and HSI classification.

3) The influence of different scales for object-based segmentation on similar-segments selection is also investigated in this article. Large segmentation-scale parameter values yield a smaller

number of segments, which aids in reducing the computational time cost for both similarity measurements and sequential-feature construction. However, those larger segments will tend to be less homogeneous, which means that some dissimilar pixels may be incorporated into the similarity calculations. The current research informs on the nature of this relationship between segment size and classification accuracy in the context of the proposed methods.

The rest of this article is organized as follows: Section 5.2 introduces some background pertaining to our research, including objected-based segmentation and LSTM methods. Section 5.3 describes the proposed sequential feature extraction and LSTM classification framework. In Section 5.4, experimental results using three benchmark HSIs are discussed. Finally, conclusions and potential suggestions are presented in Section 5.5 for future work.

## 3.2 Related work

### 3.2.1 RNN and LSTM

Compared with traditional neural networks (NN) with a feed-forward structure, the RNN is able to encode sequential inputs via a series of hidden states whose activation at each step depends on its previous step. Basically, a “recurrent” architecture is applied to each element within a sequence in a recurrent manner, where its output at one state will be fed into the computation of its next state. Such a design allows the RNN to encode previous states that persist in the internal dependencies between different states. Given  $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_L]$  as the sequential data where  $L$  denotes the sequential length of  $\mathbf{x}$ . At time step of  $t$ , hidden state  $\mathbf{s}_t$  and its output  $\mathbf{y}_t$  can be calculated as

$$\mathbf{s}_t = f_s(\mathbf{W}_{ss}\mathbf{s}_{t-1} + \mathbf{W}_{xs}\mathbf{x}_t + \mathbf{b}_t) \quad (3.1)$$

$$\mathbf{y}_t = f_y(\mathbf{W}_{sy}\mathbf{s}_t + \mathbf{b}_y) \quad (3.2)$$

where  $\mathbf{W}_{ss}$ ,  $\mathbf{W}_{xs}$ , and  $\mathbf{W}_{sy}$  represent weight matrices that are utilized for calculations from the previous state to its current state, from the input to the current hidden state, and from the current hidden state to the output, respectively.  $\mathbf{b}_t$  and  $\mathbf{b}_y$  denote bias variables.  $f_s(\cdot)$  and  $f_y(\cdot)$  are the

activation functions.

Although promising performance has been achieved by using a RNN, its shortcomings are still noticeable, and they will limit its further application, especially if the length of the input sequence is large. Due to the simple structure of the recurrent neuron, vanishing or exploding gradients may be introduced during back propagation [88]. Long short-term memory (LSTM), an improved type of RNN, is capable of learning long-term dependencies by developing more sophisticated recurrent neurons. Similar to the conventional RNN, LSTM also incorporates the previous state to update the current state. The improvement comes from adding three "gates" including a forget gate, an input gate, and an output gate. These three "gates" can regulate information flow into and out of the current state and control the updates of the recurrent neuron. Still let  $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_L]$  be the sequential input of LSTM. At time step  $t$ , the computations of forget gate  $\mathbf{f}_t$  and input gate  $\mathbf{i}_t$  are designed as follows:

$$\mathbf{f}_t = \sigma_f(\mathbf{W}_{yf}\mathbf{y}_{t-1} + \mathbf{W}_{xf}\mathbf{x}_t + \mathbf{b}_f) \quad (3.3)$$

$$\mathbf{i}_t = \sigma_i(\mathbf{W}_{yu}\mathbf{y}_{t-1} + \mathbf{W}_{xi}\mathbf{x}_t + \mathbf{b}_i) \quad (3.4)$$

where  $\mathbf{W}_{(\cdot)}$  represents weight matrices where the different subscripts represent different data-flow directions in the LSTM neuron, and  $\mathbf{b}_{(\cdot)}$  are bias variables where the different subscripts denote different gates to which they belong.  $\mathbf{y}_{t-1}$  denotes the output at time step  $t - 1$ .  $\sigma_{(\cdot)}$  are sigmoid functions where the different subscripts represent different gates to which they belong. For further details on notation, see [83]. Then new hidden state  $\mathbf{c}_t$  is updated as follows:

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_{ys}\mathbf{y}_{t-1} + \mathbf{W}_{xs}\mathbf{x}_t + \mathbf{b}_s) \quad (3.5)$$

$$\mathbf{c}_t = \mathbf{c}_{t-1} \odot \mathbf{f}_t + \tilde{\mathbf{c}}_t \odot \mathbf{i}_t \quad (3.6)$$

where  $\tanh$  is the hyperbolic tangent function,  $\odot$  stands for the element-wise product, and  $\tilde{\mathbf{c}}_t$  denotes the updated component of the hidden state. Finally, output gate  $\mathbf{o}_t$  and the output of the

current recurrent neuron  $\mathbf{y}_t$  are obtained by

$$\mathbf{o}_t = \sigma_o(\mathbf{W}_{y_o}\mathbf{y}_{t-1} + \mathbf{W}_{x_o}\mathbf{x}_t + \mathbf{b}_o) \quad (3.7)$$

$$\mathbf{y}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \quad (3.8)$$

### 3.2.2 Object-based Segmentation

In remote-sensing image and HSI classification, object-based classification is a critical technique due to its promising capability of minimizing or eliminating discrete misclassification noise compared with traditional pixel-based classification methods. As a preprocessing step, segmentation can be adopted in order to segment a whole image into multiple adjacent segments that encapsulate groups of homogeneous pixels. In this article, fractal net evolution approach (FNEA) segmentation, a well-known segmentation method which was proposed by Baatz and Schäpe [89], is utilized to construct a segmentation map of a HSI. FNEA was proposed based on a region-growing algorithm where both spectral and geometric information were integrated. The main objective of FNEA is to minimize the heterogeneity of all segments during iterative computation. The key point of FNEA is to measure the heterogeneity of two segments before and after merging them to minimize overall heterogeneity. In this context, merging two segments means combining two segments into a single larger segment. Specifically, for one segment  $s$  in a segmentation map, its heterogeneity  $h$  is formulated as follows:

$$h = w_{spec}h_{spec} + w_{shp}h_{shp} \quad (3.9)$$

where  $w_{spec}$  and  $w_{shp}$  denote weighting variables for spectral feature and shape feature, respectively, and the sum of those two weights is equal to 1.  $h_{spec}$  represents spectral heterogeneity, and  $h_{shp}$  is shape heterogeneity. To measure the spectral heterogeneity of a single segment  $s$ , it is characterized as follows:

$$h_{spec} = \sum_{k=1}^b w_k \sigma_k \quad (3.10)$$

where  $b$  is the number of spectral bands in a HSI,  $w_k$  and  $\sigma_k$  represent the weight and the standard deviation of band  $k$ , respectively. Given two adjacent segments  $s_i$  and  $s_j$  in a segment map, the spectral heterogeneity of a merged segment is defined as follows:

$$h_{spec} = \sum_{k=1}^b w_k (n_m \sigma_k^m - (n_i \sigma_k^i + n_j \sigma_k^j)) \quad (3.11)$$

where  $n_m$  is the number of pixels in a merged segment,  $\sigma_k^m$  represents the standard deviation of band  $k$  in that merged segment,  $n_i$  and  $n_j$  denote the number of pixels in those two segments  $s_i$  and  $s_j$ , and  $\sigma_k^i$  and  $\sigma_k^j$  are the standard deviations of band  $k$  in segments  $s_i$  and  $s_j$ .

The definition of shape heterogeneity,  $h_{shp}$ , is formulated as follows:

$$h_{shp} = w_{compact} h_{compact} + (1 - w_{compact}) h_{smooth} \quad (3.12)$$

where  $w_{compact}$  is a weighting parameter for compactness heterogeneity, and  $h_{compact}$  and  $h_{smooth}$  denote compactness heterogeneity and smoothness heterogeneity, respectively. Following the similar definitions in (3.10) and (3.11), the formulations for compactness and smoothness heterogeneity are provided by considering single segments and merged segments separately as well. Regarding a single segment  $s$ , its  $h_{compact}$  and  $h_{smooth}$  are defined as follows:

$$h_{compact} = \frac{p}{\sqrt{n}} \quad (3.13)$$

$$h_{smooth} = \frac{p}{\sqrt{r}} \quad (3.14)$$

where  $p$  is the perimeter of  $s$ ,  $n$  is the number of pixels in  $s$ , and  $r$  represents the perimeter of the bounding box of  $s$ . By considering two adjacent segments,  $s_i$  and  $s_j$ , that are under merging consideration, the definitions of  $h_{compact}$  and  $h_{smooth}$  are computed by

$$h_{compact} = n_m \frac{p_m}{\sqrt{n_m}} - (n_i \frac{p_i}{\sqrt{n_i}} + n_j \frac{p_j}{\sqrt{n_j}}) \quad (3.15)$$



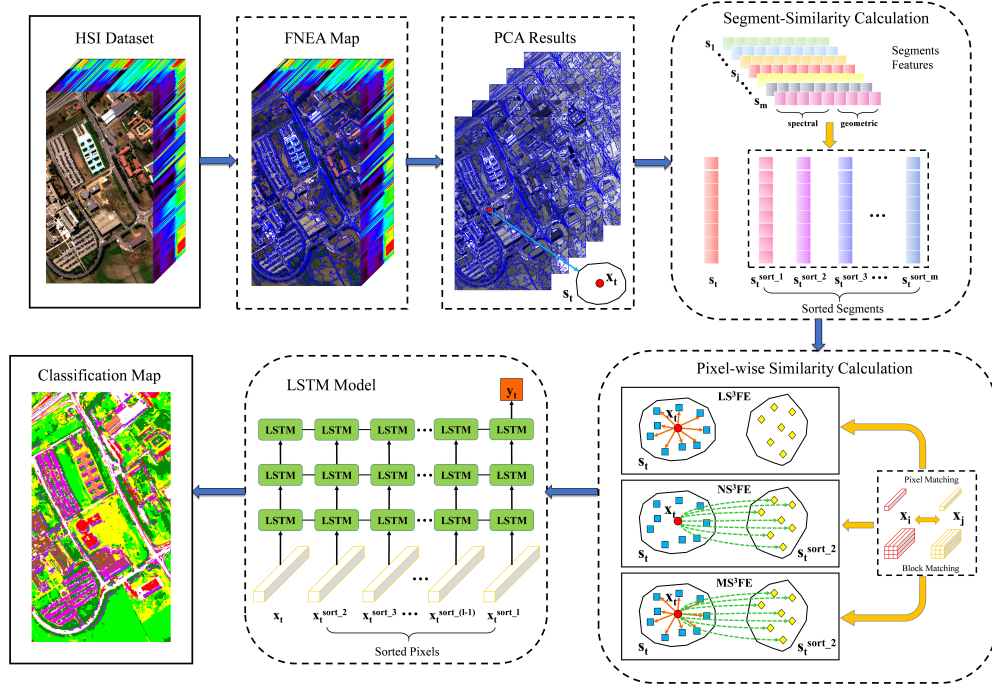


Figure 3.2: Architecture of proposed SSFE-based LSTM HSI classification framework

$$h_{smooth} = n_m \frac{p_m}{\sqrt{r_m}} - (n_i \frac{p_i}{\sqrt{r_i}} + n_j \frac{p_j}{\sqrt{r_j}}) \quad (3.16)$$

where  $n_m$  is the number of pixels in a merged segment,  $p_m$  is the perimeter of that merged segment,  $r_m$  is the perimeter of the bounding box of the merged segment,  $n_i$  and  $n_j$  are the pixel number of  $s_i$  and  $s_j$ , and  $r_i$  and  $r_j$  represent the perimeters of the bounding boxes of  $s_i$  and  $s_j$ , respectively.

### 3.3 Improved single-image sequential feature extraction

In [83], a single-image sequential feature extraction for a LSTM model for HSI classification was proposed. The main idea is to extract similar pixels from the whole HSI by calculating pixel-wise similarities and construct the sequential feature by re-ordering the spectral vectors of those selected similar pixels in a descending manner. However, such computation on global similarity measurements is exhaustive and very time-consuming, as such global searching and calculation is repeated for all pixels in a HSI. In this section, an improved single-image sequential feature extraction (SSFE) method is introduced. The method exploits an object-based segmentation map, where similar segments are first selected, and then similar pixels that will be utilized to construct sequen-

tial features are chosen only from those similar segments. More specifically, we have designed three different novel strategies to build sequential features which are discussed in this section, referred to as local segment-based SSFE (LS<sup>3</sup>FE), non-local segment-based SSFE (NS<sup>3</sup>FE), and mixed segments-based SSFE (MS<sup>3</sup>FE). Fig. 3.2 provides a basic workflow for our proposed framework.

### 3.3.1 Segment Feature Extraction

Suppose we have a HSI image  $\mathcal{X} \in \mathbb{R}^{r \times c \times b}$  with  $r$  rows,  $c$  columns, and  $b$  bands and a segment set collected from segmentation map  $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_{m-1}, \mathbf{s}_m\}$  where  $m$  denotes the number of segments. Originally, for the  $i$ th segment  $\mathbf{s}_i \in \mathbb{R}^{p_i \times b}$ , where  $p_i$  is the number of pixels comprising  $\mathbf{s}_i$ , and  $b$  represents the spectral bands. However, to better characterize the properties of segments, especially during similarity measurements between different segments, we propose in this article that both spectral features and geometry features are employed and incorporated simultaneously. For spectral features, a common approach is to calculate mean and standard deviation values, band-by-band. Due to the high dimensionality of a HSI, extracting mean and standard deviation values across all bands will introduce more feature layers than that of the original HSI. Therefore, we implement dimensionality reduction first to reduce the high dimensionality of segments-based spectral features. In this article, principal component analysis (PCA) is utilized for dimensionality reduction, and only the first several principal components (PCs) are chosen as new spectral features of segments. The other important features characterizing those segments are the geometry features, which have been widely exploited in object-based segmentation and classification. Those geometry features extracted from segment boundaries directly do not rely on the spectral features of the segment and will provide more representative spatial features in terms of the shape and spatial extent of those segments. Regarding the new spectral features generated from first several PCs, they include mean values of each PC image, standard deviation of each PC image, and brightness derived from the original image bands [90]. The geometry features consist of area, border length, the ratio between length and width, number of pixels, asymmetry, compactness, density, main direction, and rectangular fit [90]. After obtaining such new spectral and geometry features, for the

$i$ th segment  $\mathbf{s}_i$ , its feature representation can be formulated as follows:

$$\mathbf{s}_i = \text{concat}(\mathbf{s}_i^{spec}, \mathbf{s}_i^{geo}) \quad (3.17)$$

where  $\text{concat}(\cdot)$  denotes a concatenation process, and  $\mathbf{s}_i^{b_{spec}} \in \mathcal{R}^{b_1}$  and  $\mathbf{s}_i^{b_{geo}} \in \mathcal{R}^{b_2}$  represent extracted spectral and geometry features, respectively. Also,  $b_1$  and  $b_2$  are the number of new spectral and geometry features, respectively. In the remaining parts of this article, we use  $\mathbf{s}_i$  to represent the combined spectral and geometry features of the  $i$ th segment.

### 3.3.2 Segments-based Similarity Measurements

Once we obtain the representing features of the segments, the next step is to calculate the similarities between different segments. Given one target pixel  $\mathbf{x}_t \in \mathbf{s}_t$  where  $\mathbf{s}_t$  is the corresponding segment that contains  $\mathbf{x}_t$ , we measure the similarities by calculating distances between the current segment  $\mathbf{s}_t$  and other segments in order to select the most similar segments. The distances between  $\mathbf{s}_t$  and all segments in  $\mathcal{S}$  are defined as follows:

$$d_{seg}(\mathbf{s}_t, \mathcal{S}) = [d(\mathbf{s}_t, \mathbf{s}_1), \dots, d(\mathbf{s}_t, \mathbf{s}_j), \dots, d(\mathbf{s}_t, \mathbf{s}_m)] \quad (3.18)$$

where  $d(\mathbf{s}_t, \mathbf{s}_j)$  represents the distance between segments  $\mathbf{s}_t$  and  $\mathbf{s}_j$  and  $d(\cdot)$ , denotes the distance calculation function. Measuring similarity between different segments can be regarded as measuring the distance between two feature vectors of those two segments. In this article, Euclidean distance (EU) is employed as the distance calculation function to assess the segment-wise similarity. The definition of EU between two segment feature vectors  $\mathbf{s}_t$  and  $\mathbf{s}_j$  is given as follows:

$$d_{EU}(\mathbf{s}_t, \mathbf{s}_j) = \sqrt{\sum_{k=1}^{b_s} (s_t^k - s_j^k)^2} \quad (3.19)$$

where  $b_s = b_1 + b_2$  is the number of extracted spectral-geometry features defined in Section 3.3.1.

In order to extract similar segments of  $\mathbf{s}_t$ , the calculated distance list  $d_{seg}(\mathbf{s}_t, \mathcal{S})$  needs to be

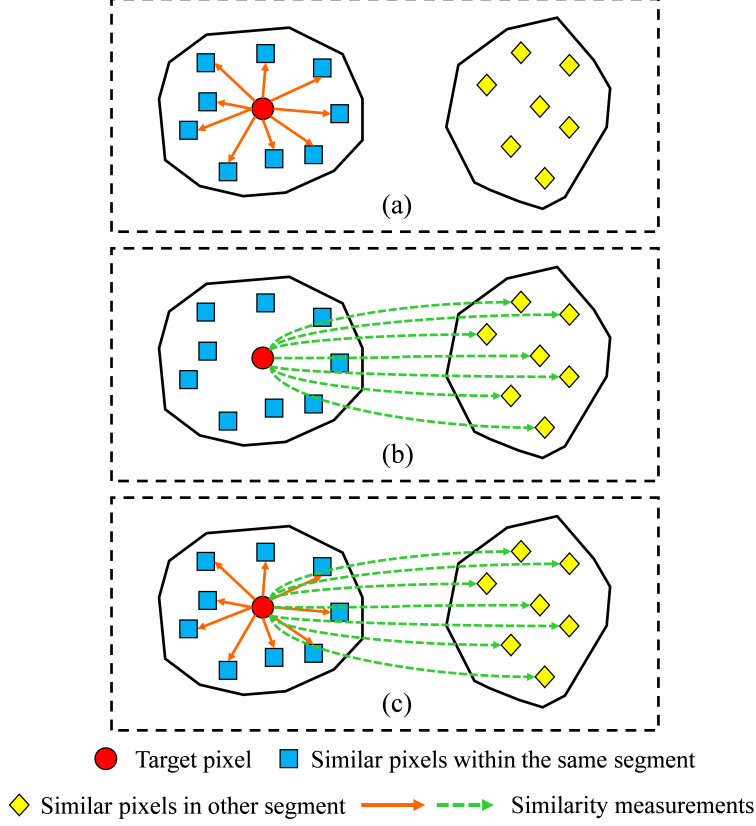


Figure 3.3: The proposed three SSFE methods : (a) Local segment-based SSFE; (b) Non-local segment-based SSFE; and (c) mixed segments-based SSFE

sorted in an ascending manner due to the intrinsic property of EU distance, which is formulated as follows:

$$\begin{aligned}
 d_{seg\_sort}(\mathbf{s}_t, \mathcal{S}) &= \text{sort}(d_{seg}(\mathbf{s}_t, \mathcal{S})) \\
 &= [d_{\text{EU}}(\mathbf{s}_t, \mathbf{s}_t^{sort\_1}), \dots, d_{\text{EU}}(\mathbf{s}_t, \mathbf{s}_t^{sort\_m})]
 \end{aligned}
 \tag{3.20}$$

where  $\text{sort}(\cdot)$  represents a sorting function,  $\mathbf{s}_t^{sort\_1}$  denotes the most similar segment compared with  $\mathbf{s}_t$ , and  $\mathbf{s}_t^{sort\_m}$  is the least similar segment. Based on (3.20), the sorted segments set, which is composed based on similarities, will be built simultaneously and can be defined as follows:

$$\mathbf{s}_t^{seg\_sort} = [\mathbf{s}_t^{sort\_1}, \dots, \mathbf{s}_t^{sort\_m}]
 \tag{3.21}$$

### 3.3.3 Improved Single-image Sequential Feature Extraction

In this section, our ultimate goal is to extract a sequential feature representation  $\mathcal{F} \in \mathbb{R}^{p \times l \times b}$  for all pixels in a HSI, where  $p = r \times c$  denotes the number of pixels in the whole HSI, and  $l$  represents the sequence length. Once similar segments  $\mathbf{s}_t^{seg\_sort}$  relative to the target segment  $\mathbf{s}_t$  are obtained, the next step is to extract pixels that are similar to target pixel  $\mathbf{x}_t$  from those similar segments for sequential feature construction. That comprises a significant difference between our previous work [83] and this article; the shrinkage of the similar-pixel searching domain from all pixels in the HSI to those similar segments is the primary innovation of the present work. As the pixel-wise similarity calculation is implemented segment-by-segment, let  $\mathbf{s}_t$  be an example segment for searching for similar pixels. The distance between  $\mathbf{x}_t$  and all pixels within  $\mathbf{s}_t$  can be defined as follows:

$$d_{px}(\mathbf{x}_t, \mathbf{s}_t) = [d(\mathbf{x}_t, \mathbf{x}_1), \dots, d(\mathbf{x}_t, \mathbf{x}_j), \dots, d(\mathbf{x}_t, \mathbf{x}_{n\_px})] \quad (3.22)$$

where  $n\_px$  denotes the number of pixels in  $\mathbf{s}_t$  and  $d(\cdot)$  represents the distance calculation function. Regarding the distance calculation and extraction of similar pixels, based on [83], pixel matching (PM) and block matching (BM) are still utilized here as the matching schemes for pixel-wise similarity measurements. Within the PM scheme, spectral angle mapping (SAM) is adopted in this article. Given  $\forall \mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}$ , its SAM distance is formulated as follows:

$$d_{SAM}(\mathbf{x}_i, \mathbf{x}_j) = \cos^{-1}\left(\frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}\right). \quad (3.23)$$

For the BM scheme, the image patch distance (IPD) proposed in [56] is also utilized as the similarity calculation function whose inputs are image patches centered by two target pixels instead of individual spectral vectors. To maintain consistency, we still use the SAM scheme when calculating IPD between different blocks for BM-based methods. Once a particular distance calculation function is applied, the sorted  $d_{px}(\mathbf{x}_t, \mathbf{s}_t)$  and sorted pixel sets  $\mathbf{x}_t^{px\_sort}$  are characterized in

a manner similar to that described in (3.20) and (3.21)

$$\begin{aligned} d_{px\_sort}(\mathbf{x}_t, \mathbf{s}_t) &= \text{sort}(d_{px}(\mathbf{x}_t, \mathbf{s}_t)) \\ &= [d(\mathbf{x}_t, \mathbf{x}_t^{sort_{-1}}), \dots, d(\mathbf{x}_t, \mathbf{x}_t^{sort_{-n\_px}})] \end{aligned} \quad (3.24)$$

$$\mathbf{x}_t^{px\_sort} = [\mathbf{x}_t^{sort_{-1}}, \dots, \mathbf{x}_t^{sort_{-n\_px}}] \quad (3.25)$$

where  $\mathbf{x}_t^{sort_{-1}}$  represents the most similar pixel to  $\mathbf{x}_t$  in  $\mathbf{s}_t$  and  $\mathbf{x}_t^{sort_{-n\_px}}$  is the least similar pixel.

Another key point pertaining to our proposed method is how many similar segments and what kind of segments will be incorporated to calculate pixel-wise similarities. Here we propose three different strategies, graphically illustrated in Fig. 3.3 and referred to as local segment-based SSFE (LS<sup>3</sup>FE), non-local segment-based SSFE (NS<sup>3</sup>FE), and mixed segments-based SSFE (MS<sup>3</sup>FE); these strategies are explained in detail as follows:

1) *LS<sup>3</sup>FE*: In LS<sup>3</sup>FE, only the segment that contains current target pixel  $\mathbf{x}_i$  is taken into consideration for pixel-wise similarity measurements. In (3.22), such a segment is  $\mathbf{s}_t^{sort_{-1}}$ , which is the first element of  $\mathbf{s}_t^{seg\_sort}$  since only the segment itself is the most similar to it. If such a searching range is determined, those pixels will be reordered in a descending manner using the similarities calculated via either the PM or BM schemes. However, with the restriction of sequential length  $l$ , extracting enough pixels from a single segment is a challenging task, especially if that segment comes from a small-scale segmentation map. Such a challenge also occurs with the NS<sup>3</sup>FE strategy, described below. To address this issue, we adopt a repeat-sampling strategy to satisfy the requirement of a given sequential length  $l$ . If  $l$  exceeds the number of pixels in the current processing segment, the previous part of sequence is constructed as normal, and the remaining part of the sequence that exceeds the number of pixels in the segment is built by randomly selecting pixels from the current segment, one-by-one, until the constructed sequence length  $l'$  reaches  $l$  [87].

2) *NS<sup>3</sup>FE*: For NS<sup>3</sup>FE, local segment  $\mathbf{s}_t^{sort_{-1}}$  will not be selected for pixel-wise similarity calculation. Apart from the most similar segment, which is the one containing the target pixel itself, we utilize the second-most similar segment compared with the current segment, which is

the second element of  $s_t^{seg\_sort}$ . Local spatial contextual information has been well-investigated for feature extraction and HSI classification. However, regarding the examination of non-local spatial information, limited research has been conducted. In this article, NS<sup>3</sup>FE provides insights on the effects of non-local spatial information on SSFE. Still, the repeat-sampling strategy will be utilized as well when the pre-defined  $l$  is larger than the number of pixels in the current processing segment.

3) *MS<sup>3</sup>FE*: Different from LS<sup>3</sup>FE and NS<sup>3</sup>FE, MS<sup>3</sup>FE does not choose a single segment but multiple segments, and pixels within those selected multiples segments will be selected only once in pixel-wise similarity calculation and sequential feature construction. Still supposing that  $l$  exceeds the number of pixels in the first segment in  $s_t^{seg\_sort}$  which contains target pixel  $x_i$ , those pixels in the current processing segment will be selected and sorted based on calculated similarities as normal. Then, the method will move to the next segment in  $s_t^{seg\_sort}$  to execute the same processing as that in its previous step in order to select pixels to build the sequence until the length of the created sequence  $l'$  reaches  $l$ .

In a nutshell, the procedures of proposed LS<sup>3</sup>FE, NS<sup>3</sup>FE, and MS<sup>3</sup>FE are summarized in Algorithm 1 and 2. Note that the only difference between LS<sup>3</sup>FE and NS<sup>3</sup>FE is with the selection of the segment; we combine those two methods into a single pseudo-code.

### 3.4 Experiments

In this section, we apply the proposed three different SSFE methods on three benchmark HSIs and evaluate their classification performance by comparing them with other state-of-the-art classification models. Moreover, we also conduct a sensitivity analysis of the hyperparameters in this section.

#### 3.4.1 Datasets

1) *Pavia University*: The first HSI utilized is the Pavia University HSI (PU), which was collected by the Reflective Optics System Imaging Spectrometer (ROSIS) sensor over Pavia University, Italy. PU consists of  $610 \times 340$  pixels with 115 spectral bands. After removing noise and water vapor-absorption bands, 103 bands remain for the experiments. The spatial resolution is 1.3

---

**Algorithm 1** Proposed LS<sup>3</sup>FE and NS<sup>3</sup>FE Framework

---

**Input:** HSI  $\mathcal{X}$ , unsorted segment set  $\mathcal{S}$ , sequence length  $l$ ,  
extraction type  $e_t$

**Initialization:** Sequential feature representation  $\mathcal{F} \leftarrow []$

**Begin:**

- 1: **for**  $\mathbf{x}_t$  in  $\mathcal{X}$  **do**
- 2:   Find segment  $\mathbf{s}_t$  that contains  $\mathbf{x}_t$
- 3:   Calculate  $d_{seg}(\mathbf{s}_t, \mathcal{S})$  by (3.18)
- 4:   Calculate  $d_{seg\_sort}(\mathbf{s}_t, \mathcal{S})$  by (3.20)
- 5:   Extract sorted segment set  $\mathbf{s}_t^{seg\_sort}$  by (3.21)
- 6:   **if**  $e_t$  is LS<sup>3</sup>FE **then**
- 7:      $\mathbf{s}_{cur} \leftarrow \mathbf{s}_t^{seg\_sort}[0]$
- 8:   **else**
- 9:      $\mathbf{s}_{cur} \leftarrow \mathbf{s}_t^{seg\_sort}[1]$
- 10:   **end if**
- 11:   Compute number of pixels  $n_{cur}$  of  $\mathbf{s}_{cur}$
- 12:   Calculate  $d_{px}(\mathbf{x}_t, \mathbf{s}_{cur})$  by (3.22)
- 13:   Calculate  $d_{px\_sort}(\mathbf{x}_t, \mathbf{s}_{cur})$  by (3.24)
- 14:   Extract sorted pixel set  $\mathbf{x}_t^{px\_sort}$  by (3.25)
- 15:   **if**  $n_{cur} \geq l$  **then**
- 16:      $\mathcal{F}.append(\mathbf{x}_t^{px\_sort}[0 : l])$
- 17:   **else**
- 18:      $\mathcal{F}.append(\mathbf{x}_t^{px\_sort}[0 : n_{cur}])$
- 19:     **while**  $n_{cur} < l$  **do**
- 20:        $n_{cur} \leftarrow n_{cur} + 1$
- 21:        $x_{rand} \leftarrow RandSelect(\mathbf{x}_t^{px\_sort})$
- 22:        $\mathcal{F}.append(x_{rand})$
- 23:     **end while**
- 24:   **end if**
- 25: **end for**

**Output:** Sequential feature representation  $\mathcal{F}$

---



---

**Algorithm 2** Proposed MS<sup>3</sup>FE Framework

---

**Input:** HSI  $\mathcal{X}$ , unsorted segment set  $\mathcal{S}$ , sequence length  $l$

**Initialization:** Sequential feature representation  $\mathcal{F} \leftarrow []$

**Begin:**

- 1: **for**  $\mathbf{x}_t$  in  $\mathcal{X}$  **do**
- 2:   Find segment  $\mathbf{s}_t$  that contains  $\mathbf{x}_t$
- 3:   Calculate  $d_{seg}(\mathbf{s}_t, \mathcal{S})$  by (3.18)
- 4:   Calculate  $d_{seg\_sort}(\mathbf{s}_t, \mathcal{S})$  by (3.20)
- 5:   Extract sorted segment set  $\mathbf{s}_t^{seg\_sort}$  by (3.21)
- 6:   Initialize accumulated sequence length  $l' \leftarrow 0$
- 7:   Initialize current segment index  $i \leftarrow 0$
- 8:   **while**  $l' < l$  **do**
- 9:      $\mathbf{s}_{cur} \leftarrow \mathbf{s}_t^{seg\_sort}[i]$
- 10:     Calculate  $d_{px}(\mathbf{x}_t, \mathbf{s}_{cur})$  by (3.22)
- 11:     Calculate  $d_{px\_sort}(\mathbf{x}_t, \mathbf{s}_{cur})$  by (3.24)
- 12:     Extract sorted pixels set  $\mathbf{x}_t^{px\_sort}$  by (3.25)
- 13:     Compute number of pixels  $n_{cur}$  of  $\mathbf{s}_{cur}$
- 14:      $l' \leftarrow l' + n_{cur}$
- 15:     **if**  $l' \leq l$  **then**
- 16:        $\mathcal{F}.append(\mathbf{x}_t^{px\_sort})[0 : n_{cur}]$
- 17:     **else**
- 18:        $\mathcal{F}.append(\mathbf{x}_t^{px\_sort})[0 : l - l']$
- 19:     **end if**
- 20:      $i \leftarrow i + 1$
- 21:   **end while**
- 22: **end for**

**Output:** Sequential feature representation  $\mathcal{F}$

---

Class No.	Class Name	Number of Samples
C1	Asphalt	6631
C2	Meadows	18649
C3	Gravel	2099
C4	Trees	3064
C5	Painted Metal Sheets	1345
C6	Bare Soil	5029
C7	Bitumen	1330
C8	Self-Blocking Bricks	3682
C9	Shadows	947
-	Total	42776

Table 3.1: Class Codes and Sample Sizes for Pavia University Image

m and the spectral range is from 430 nm to 860 nm. Nine classes are labeled in the corresponding ground-reference data, and the specific ground categories along with an example false-color composite image are given in Fig. 3.4, and their corresponding class codes and samples sizes are given in Table ??.

2) *Salinas*: The Salinas HSI was obtained by employing the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) over Salinas Valley, California, USA. It comprises 512 rows by 217 columns with a spatial resolution of 3.7 m. Twenty water vapor-absorption bands were discarded, and the remaining 200 bands were kept for subsequent analysis. The Salinas HSI entails 16 classes, as shown in Fig. 3.5. Its class codes and the number of samples per class are listed in Table 3.2.

3) *Indian Pines*: The Indian Pines (INP) HSI was also acquired via the AVIRIS sensor over Northwest Indiana in June 1992. The spatial dimension is  $145 \times 145$  pixels, its spatial resolution is 20 m, with 224 spectral bands. The spectral wavelength range is from 400 nm to 2400 nm. The number of bands was reduced by removing 24 water vapor-absorption bands, with 200 bands remaining. Fig. 3.6 illustrates a false-color composite image, the ground-reference map, and the corresponding list of labeled classes for the INP dataset. More detailed information, including number of samples, is provided in Table 3.3.

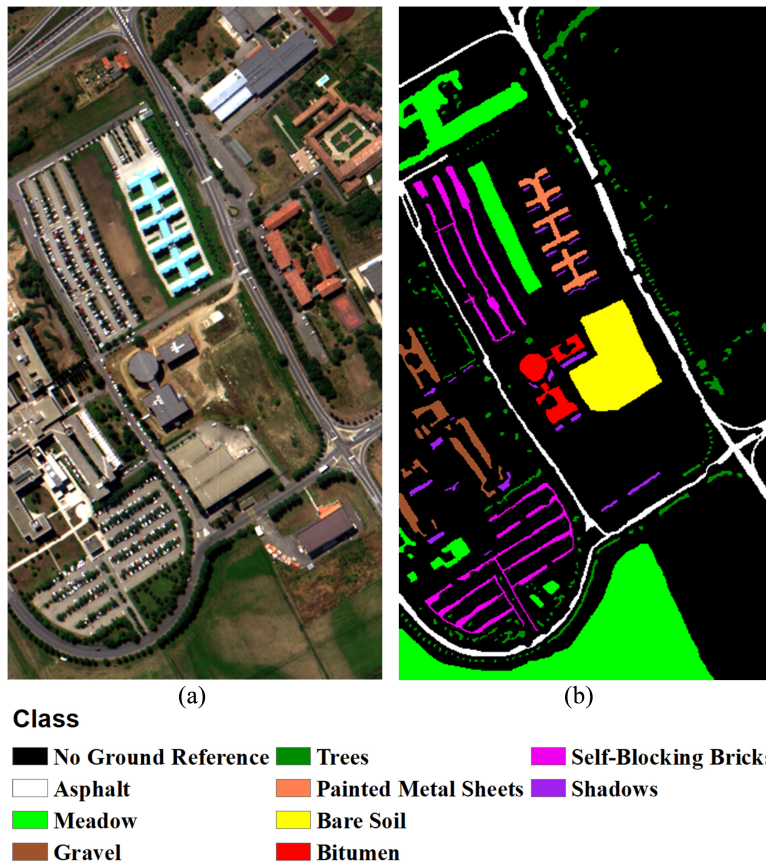


Figure 3.4: Pavia University hyperspectral image: (a) False-color composite (R: band 55, G: band 33, and B: band 13); (b) Ground-reference data

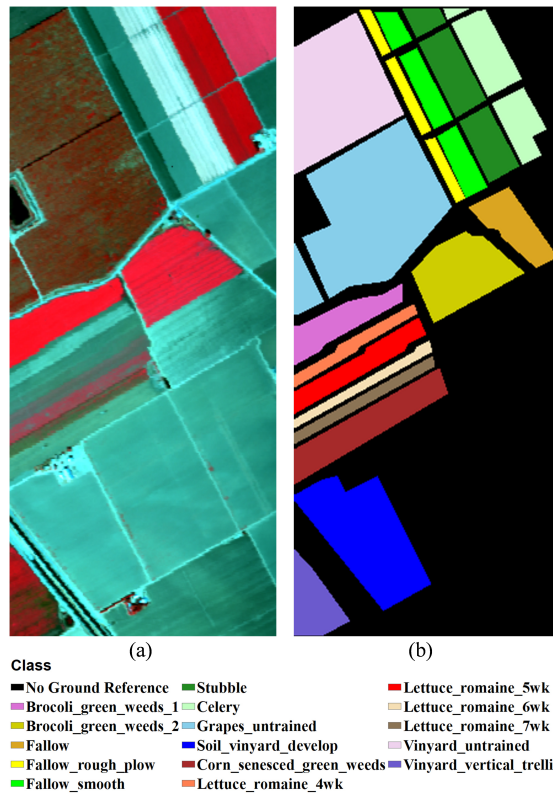


Figure 3.5: Salinas hyperspectral image: (a) False-color composite (R: band 57, G: band 27, and B: band 17); (b) Ground-reference data

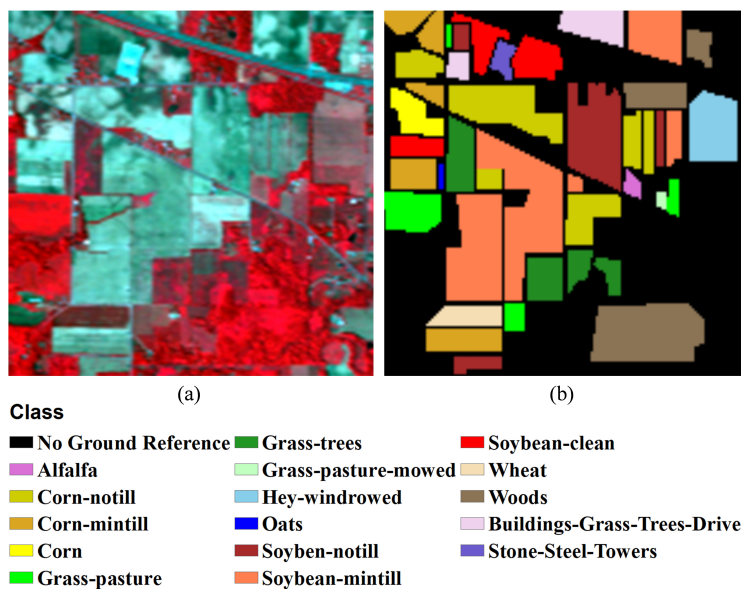


Figure 3.6: Indian Pines hyperspectral image: (a) False-color composite (R: band 57, G: band 27, and B: band 18); (b) Ground-reference data

Table 3.2: Class Codes and Sample Sizes for Salinas Image

Class No.	Class Name	Number of Samples
C1	Brocoli_green_weeds_1	2009
C2	Brocoli_green_weeds_2	3726
C3	Fallow	1976
C4	Fallow_rough_plow	1394
C5	Fallow_smooth	2678
C6	Stubble	3959
C7	Celery	3579
C8	Grapes_untrained	11271
C9	Soil_vinyard_develop	6203
C10	Corn_senesced_green_weeds	3278
C11	Lettuce_romaine_4wk	1068
C12	Lettuce_romaine_5wk	1927
C13	Lettuce_romaine_6wk	916
C14	Lettuce_romaine_7wk	1070
C15	Vinyard_untrained	7268
C16	Vinyard_vertical_trellis	1807
-	Total	54129

### 3.4.2 Experimental Setup

To characterize the performance of our proposed methods quantitatively, similar to [83], three benchmark, or baseline, algorithms are still utilized for comparison, including SVM, 1D\_CNN, and 1D\_LSTM. Additionally, in order to examine the efficiency improvements of our proposed methods, the model developed in [83] with the best classification performance is also utilized here for comparison in this article. Information regarding the different models used is summarized as follows:

1) *SVM*: SVM with a radial basis function (RBF) kernel as the kernel function. The best parameter combination is obtained from cross-validation.

2) *1D\_CNN*: The architecture of a 1D\_CNN is set based on [36], which contains two convolution layers, two maximum pooling layers, and one fully-connected layer for classification for all three HSIs. 10 convolution kernels are utilized, with a size of 8. Regarding the maximum pooling layer, the kernel size is 2. Furthermore, the number of hidden units for the final fully-connected

Table 3.3: Class Codes and Sample Sizes for Indian Pines Image

Class No.	Class Name	Number of Samples
C1	Alfalfa	46
C2	Corn-notill	1428
C3	Corn-mintill	830
C4	Corn	237
C5	Grass-pasture	483
C6	Grass-trees	730
C7	Grass-pasture-mowed	28
C8	Hay-windrowed	478
C9	Oats	20
C10	Soybean-notill	972
C11	Soybean-mintill	2455
C12	Soybean-clean	593
C13	Wheat	205
C14	Woods	1265
C15	Buildings-Grass-Trees-Drives	386
C16	Stone-Steel-Towers	93
-	Total	10249

layer is set to be the number of classes that the specific HSI entails.

3) *1D\_LSTM*: Similar to the 1D\_CNN, 1D\_LSTM is applied to the original 1D spectrum feature. Three LSTM layers are adopted, where the numbers of LSTM units are 32, 64, and 128, respectively. The last fully-connected layer, whose dimension is equal to the number of classes encompassed by the HSI currently being processed, is also included for classification purposes.

4) *BM\_LSTM*: LSTM with the block-matching (BM) strategy to extract similar pixels and construct sequential features, as proposed in [83]. The method of calculating pixel-wise similarity is SAM. For the LSTM network structure, we follow the implementation of LSTM utilized in [83], where four LSTM layers and two fully-connected layers are exploited. The dimensions of the LSTM layers are 32, 64, 128, and 256, respectively. Before the last fully-connected layer (which is same as that of the 1D\_CNN and 1D\_LSTM), there is one additional fully-connected layer with 50 hidden units. Note that such a LSTM model-structure design is applied to all of our LSTM-based schemes in order to maintain consistency for comparisons of classification performance and

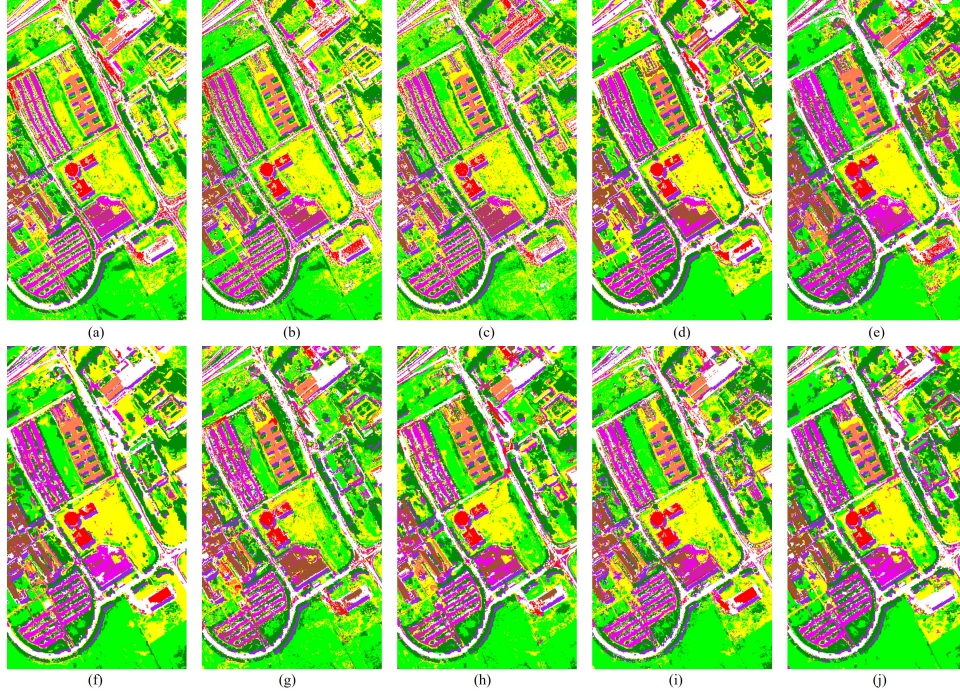


Figure 3.7: Classification results from the PU image, from the eighth trial: (a) SVM (82.65%); (b) 1D\_CNN (84.98%); (c) 1D\_LSTM (83.52%); (d) BM\_LSTM (95.94%); (e) LS<sup>3</sup>FE\_LSTM (PM) (93.39%); (f) LS<sup>3</sup>FE\_LSTM (BM) (95.24%); (g) NS<sup>3</sup>FE\_LSTM (PM) (86.31%); (h) NS<sup>3</sup>FE\_LSTM (BM) (90.36%); (i) MS<sup>3</sup>FE\_LSTM (PM) (93.83%); and (j) MS<sup>3</sup>FE\_LSTM (BM) (97.28%)

computational cost.

5) *LS<sup>3</sup>FE\_LSTM*: LSTM with LS<sup>3</sup>FE. Due to the different pixel-wise similarity calculations, for simplicity, LS<sup>3</sup>FE\_LSTM using PM is abbreviated as LS<sup>3</sup>FE\_LSTM (PM), and LS<sup>3</sup>FE\_LSTM using BM is abbreviated as LS<sup>3</sup>FE\_LSTM (BM).

6) *NS<sup>3</sup>FE\_LSTM*: LSTM with NS<sup>3</sup>FE. Similar to the case of LS<sup>3</sup>FE\_LSTM, the two NS<sup>3</sup>FE\_LSTM approaches using PM and BM are abbreviated as NS<sup>3</sup>FE\_LSTM (PM) and NS<sup>3</sup>FE\_LSTM (BM), respectively.

7) *MS<sup>3</sup>FE\_LSTM*: LSTM with MS<sup>3</sup>FE. Same as with the two aforementioned S<sup>3</sup>FE-based LSTMs, MS<sup>3</sup>FE\_LSTM (PM) and MS<sup>3</sup>FE\_LSTM (BM) represent two MS<sup>3</sup>FE\_LSTM-based methods using PM and BM, respectively.

For the segmentation maps that are utilized in all aforementioned proposed S<sup>3</sup>FE-based meth-



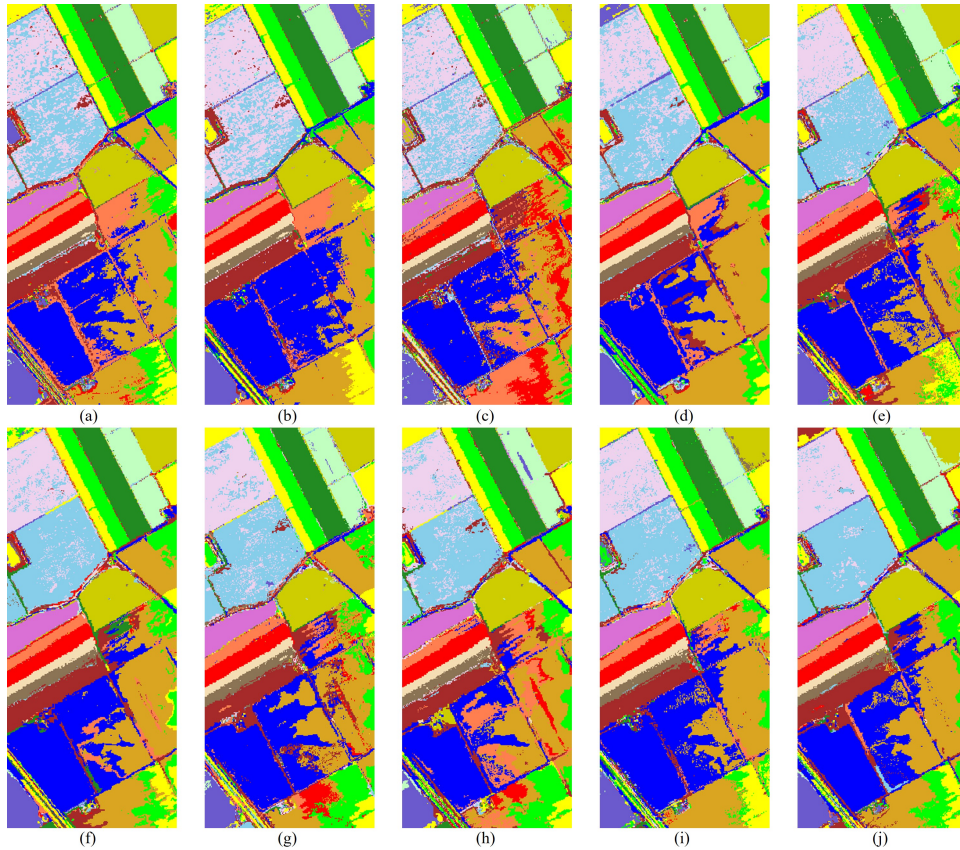


Figure 3.8: Classification results from the Salinas image, from the eighth trial: (a) SVM (85.23%); (b) 1D\_CNN (85.98%); (c) 1D\_LSTM (85.17%); (d) BM\_LSTM (91.94%); (e) LS<sup>3</sup>FE\_LSTM (PM) (94.17%); (f) LS<sup>3</sup>FE\_LSTM (BM) (95.29%); (g) NS<sup>3</sup>FE\_LSTM (PM) (92.92%); (h) NS<sup>3</sup>FE\_LSTM (BM) (92.95%); (i) MS<sup>3</sup>FE\_LSTM (PM) (94.91%); and (j) MS<sup>3</sup>FE\_LSTM (BM) (96.85%)



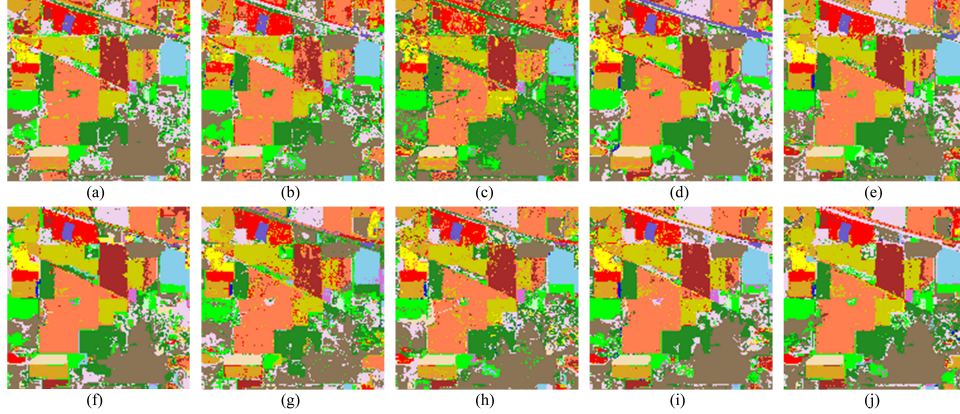


Figure 3.9: Classification results from the INP image, from the eighth trial: (a) SVM (86.00%); (b) 1D\_CNN (79.71%); (c) 1D\_LSTM (76.46%); (d) BM\_LSTM (89.91%); (e) LS<sup>3</sup>FE\_LSTM (PM) (93.43%); (f) LS<sup>3</sup>FE\_LSTM (BM) (94.58%); (g) NS<sup>3</sup>FE\_LSTM (PM) (86.57%); (h) NS<sup>3</sup>FE\_LSTM (BM) (87.16%); (i) MS<sup>3</sup>FE\_LSTM (PM) (92.60%); and (j) MS<sup>3</sup>FE\_LSTM (BM) (93.79%)

ods, we select the eCognition embedded FNEA-segmentation algorithm [90] to obtain the segmentation maps. Parameter values for the scale, compactness, and shape parameters are set to 40, 0.1, and 0.5, respectively. For the extraction of new spectral features with reduced dimensionality, which will be used for segments-based similarity calculations, the first 5 PCs are selected for all three HSIs. To ensure that classification results derived from different classification models are quantitatively comparable, we split the ground-reference data for each HSI into training and testing datasets via a controlled scheme. More specifically, for the PU and Salinas images, 200 samples per class are selected randomly as training samples, and all remaining samples from the ground-reference data are used as testing samples. Regarding the INP image, due to the relatively limited number of labeled pixels for some classes only 30% of samples per class from the ground-reference data are used as training samples, and the remaining samples are exploited as testing samples. Such training-testing sample selection is utilized for all classification models in order to reduce potential errors introduced by sample differences. Moreover, to avoid sampling bias, 10 replications of experiments using such a random sample-selection method for each model are performed, and all classification accuracies are obtained by averaging those accuracies across all

10 replications.

Regarding LSTM-based model training, all codes are implemented based on the Keras framework [91], with TensorFlow [92] as the backend. For model optimization, the optimizer that was utilized is Adadelta, with a learning rate of 0.5. The number of epochs and the batch size are set to 500 and 20, respectively. All experiments regarding DL model training were conducted with the advanced computing resources provided by Texas A&M High Performance Research Computing.

The quantitative metrics for evaluating classification performance used in this research are overall accuracy (OA), average accuracy (AA), and the Kappa coefficient (Kappa). OA is defined by calculating the ratio between the number of pixels classified correctly and the number of all pixels in the set of testing samples. AA is the average of all accuracies obtained across all classes. Kappa is a statistical index for a consistency test, which can be calculated from a confusion matrix.

### 3.4.3 Classification Performance Comparisons

The quantitative classification results for the three HSIs of interest are shown in Tables ??, ??, and ??. For the PU image, the best classification performance was obtained by MS<sup>3</sup>FE\_LSTM (BM) with OA=96.89%, AA=95.17%, and Kappa=95.01%. Compared with BM\_LSTM, which was proposed in our previous work, the OA and AA improved by 0.69% and 0.52%, respectively. Regarding our other proposed methods, such as the LS<sup>3</sup>FE\_LSTM- and MS<sup>3</sup>FE\_LSTM-based methods, the performance of each method is still competitive compared with the benchmark algorithms. The OAs of the two NS<sup>3</sup>FE\_LSTM-based approaches even decreased relative to those of BM\_LSTM, but they still perform better than SVM, 1D\_CNN, and 1D\_LSTM. The two LS<sup>3</sup>FE\_LSTM methods performed better (with OA=94.16% and 96.03%, respectively) than the two MS<sup>3</sup>FE\_LSTM models. However, their OAs are still less than that of the best MS<sup>3</sup>FE\_LSTM model. Moreover, within each S<sup>3</sup>FE scheme, the BM-based method always performed better than that of the PM-based method, which demonstrates the effectiveness of the utilization of spatial-contextual information in the PU image.

For the classification maps shown in Fig. 3.7, the apparent improvements in classification performance can be visually observed. In Fig. 3.7 (a) to (c), the “salt-and-pepper” phenomenon

is obvious, as no spatial-contextual information is utilized for those classifications. For Fig. 3.7 (d), which results from our proposed BM-based method in [83], a smoother classification map was produced by exploiting spatial-contextual information when measuring pixel-wise similarities. For Fig. 3.7 (e) to (j), those classified images generated from the six newly-proposed methods visually exhibit strong correspondences with their respective quantitative accuracy results (e.g., overall accuracies). Among those six classification maps, the two NS<sup>3</sup>FE\_LSTM methods produced more misclassified pixels than other four approaches. The best classification was obtained by MS<sup>3</sup>FE\_LSTM (BM), as it has the highest OA. When comparing the classification maps between each of the sequential feature-extraction schemes, it is also quite evident that, with BM-based methods, there are reductions in discrete misclassified pixels within relatively homogeneous areas in the image, and they thus entail more smoothed classified images. Such a phenomenon illustrates the capability of the utilization of spatial-contextual information for similarity measurements during the extraction of the sequential features.

For the Salinas image, whose classification results are displayed in Table ??, results/phenomena similar to those observed with the PU image are attained. The best classification performance was still obtained from MS<sup>3</sup>FE\_LSTM (BM), with OA=96.64%, AA=95.95%, and Kappa=96.24%. The two NS<sup>3</sup>FE\_LSTM-based methods still performed worse than the other two sequential feature-extraction approaches. The OAs obtained from the two LS<sup>3</sup>FE\_LSTM-based methods are higher than those from the NS<sup>3</sup>FE\_LSTM-based methods, but lower than that from MS<sup>3</sup>FE\_LSTM (BM). The performance of each of the six proposed methods is better (more accurate) than that of the baseline algorithms, especially for BM\_LSTM. Furthermore, the BM-based approaches still obtained higher classification accuracies than their corresponding PM-based methods.

The classification maps derived from the Salinas image are given in Fig. 3.8. The "salt-and-pepper" phenomenon is still very apparent for those approaches that do not exploit spatial-contextual information. After combining both spectral and spatial features in the BM-based matching schemes, such misclassification are markedly alleviated, and the OAs improved as well. Through visual interpretation, we observe that for the MS<sup>3</sup>FE\_LSTM (BM) result, which has the highest

Class No.	Classification Algorithms											
	SVM	1D_CNN	ID_LSTM	BM_LSTM	LS <sup>3</sup> FE_LSTM (PM)	LS <sup>3</sup> FE_LSTM (BM)	NS <sup>3</sup> FE_LSTM (PM)	NS <sup>3</sup> FE_LSTM (BM)	MS <sup>3</sup> FE_LSTM (PM)	MS <sup>3</sup> FE_LSTM (BM)		
C1	97.33	96.62	95.47	98.27	98.59	97.62	96.38	97.42	<b>98.30</b>	98.24		
C2	95.55	94.16	93.69	98.26	98.62	98.22	95.17	96.48	98.55	<b>99.17</b>		
C3	64.38	66.14	68.39	91.22	90.68	92.56	73.00	75.87	90.68	<b>92.62</b>		
C4	59.55	85.71	75.45	94.02	85.52	93.67	80.12	82.33	87.38	<b>95.50</b>		
C5	95.80	97.02	98.78	<b>99.47</b>	96.40	95.02	95.82	96.83	95.69	94.89		
C6	64.14	58.86	59.58	91.20	83.11	90.84	62.30	69.65	84.03	<b>91.28</b>		
C7	52.28	74.87	66.83	90.36	83.28	88.85	70.97	72.46	83.84	<b>90.42</b>		
C8	84.54	83.90	80.81	93.27	96.50	92.72	86.34	89.05	93.71	<b>93.90</b>		
C9	99.86	99.60	<b>99.62</b>	98.86	98.15	98.64	98.11	97.57	95.21	96.64		
OA	81.79	85.29	83.70	96.17	94.16	96.03	85.96	88.54	94.18	<b>96.89</b>		
AA	79.50	83.92	81.92	94.32	92.07	94.17	84.31	86.41	91.93	<b>95.17</b>		
Kappa	76.57	80.69	74.68	94.87	92.25	94.51	81.62	84.96	92.26	<b>95.01</b>		

Table 3.4: Classification results (in units of %) in Pavia University Image

OA, its classification map is less noisy relative to other maps. For the class Grapes\_untrained, colored in light blue in the central part of Salinas image, more pixels are classified correctly compared with other approaches, which provides additional evidence of the effectiveness of the BM-based strategy.

For the INP image, the best results are obtained from the LS<sup>3</sup>FE\_LSTM (BM) method, with OA=94.04%, AA=93.60%, and Kappa=93.21%, which is different from what we attained in the PU and Salinas images. The two NS<sup>3</sup>FE\_LSTM-based approaches still have lower accuracies, with OA=85.86% and 87.92%, respectively, which are even lower than that of the BM\_LSTM. The two MS<sup>3</sup>FE\_LSTM-based approaches achieve higher OAs than those of the NS<sup>3</sup>FE\_LSTM-based methods. We can still find that the performance of each of the BM-based methods can obtain higher OAs than their corresponding PM-based methods.

All of the classification maps derived from the INP image are displayed in Fig. 3.9. Similar to the PU and Salinas image-classification results, those approaches where only PM- or only spectral-feature-based features are applied still yield noisy classification maps, and those methods produce lower OAs compared with their corresponding BM-based approaches. Given the OAs of each classification map, we observe that the classification map with the highest OA is generated by LS<sup>3</sup>FE\_LSTM (BM). Those misclassified pixels that exist in the other classification maps are generally classified correctly via LS<sup>3</sup>FE\_LSTM (BM), especially for those pixels belonging to the Soybean-mintill and Soybean-notill class, located in the central part of the image.

#### **3.4.4 Parameter Analysis**

In order to analyze and evaluate the influence of different parameters that are employed with our proposed methods, parameter sensitivity analyses were conducted for three parameters: the number of training samples  $n_{tr}$ , sequence length  $l$ , and the scale parameter in FNEA  $s_f$ . Moreover, an evaluation of the computational time cost for SSFE was also conducted, as explained in this section.

Class No.	Classification Algorithms											
	SVM	ID_CNN	ID_LSTM	BM_LSTM	LS <sup>3</sup> FE_LSTM (PM)	LS <sup>3</sup> FE_LSTM (BM)	NS <sup>3</sup> FE_LSTM (PM)	NS <sup>3</sup> FE_LSTM (BM)	MS <sup>3</sup> FE_LSTM (PM)	MS <sup>3</sup> FE_LSTM (BM)		
C1	97.30	99.41	94.77	<b>99.78</b>	99.39	99.20	99.73	98.63	99.63	98.96		
C2	98.61	98.98	98.84	99.15	96.31	<b>99.70</b>	99.44	99.42	99.56	99.54		
C3	84.95	95.50	88.72	96.68	97.28	97.25	96.24	<b>97.96</b>	98.13	97.51		
C4	97.21	98.03	96.20	<b>98.13</b>	95.11	96.14	95.01	96.87	97.28	96.08		
C5	95.27	96.84	98.29	98.16	98.54	99.27	<b>99.59</b>	98.71	98.71	99.34		
C6	99.65	99.68	96.81	99.20	99.16	99.20	<b>99.68</b>	98.92	99.55	99.31		
C7	98.53	97.69	97.78	99.16	99.19	99.26	96.71	97.23	98.92	<b>99.31</b>		
C8	76.88	84.97	78.81	86.03	94.23	93.35	91.04	91.72	93.90	<b>96.46</b>		
C9	99.12	99.08	97.72	99.14	99.33	98.86	98.10	98.40	98.91	<b>99.41</b>		
C10	82.48	83.71	84.75	91.28	92.94	93.74	85.91	92.10	<b>94.19</b>	94.17		
C11	68.72	81.95	80.64	85.87	<b>90.04</b>	91.95	76.21	81.90	86.94	89.69		
C12	93.04	97.37	85.49	94.11	93.17	96.34	97.79	<b>97.96</b>	95.76	95.00		
C13	92.37	93.87	91.67	<b>97.24</b>	94.09	95.66	91.30	92.41	96.31	95.96		
C14	<b>90.62</b>	90.20	85.32	84.15	82.87	83.41	86.55	83.78	81.52	84.02		
C15	57.38	58.26	54.65	70.76	84.73	86.08	80.47	80.62	84.68	<b>93.09</b>		
C16	95.03	97.72	93.37	96.94	97.02	96.91	88.50	89.49	96.42	<b>97.37</b>		
OA	84.81	84.85	83.99	90.94	94.08	95.44	92.24	92.92	94.62	<b>96.64</b>		
AA	89.25	91.70	88.67	94.21	94.53	95.82	92.64	93.51	95.03	<b>95.95</b>		
Kappa	81.06	82.39	82.19	89.89	93.39	94.90	91.33	92.09	93.99	<b>96.24</b>		

Table 3.5: Classification results (in units of %) in Salinas Image

Class No.	Classification Algorithms											
	SVM	ID_CNN	ID_LSTM	BM_LSTM	LS <sup>3</sup> FE_LSTM (PM)	LS <sup>3</sup> FE_LSTM (BM)	NS <sup>3</sup> FE_LSTM (PM)	NS <sup>3</sup> FE_LSTM (BM)	MS <sup>3</sup> FE_LSTM (PM)	MS <sup>3</sup> FE_LSTM (BM)	MS <sup>3</sup> FE_LSTM (PM)	MS <sup>3</sup> FE_LSTM (BM)
C1	93.21	86.02	81.34	77.98	<b>95.17</b>	86.08	93.01	80.52	93.05	80.52	93.05	85.79
C2	80.64	78.56	74.74	87.93	89.46	<b>92.47</b>	78.87	81.54	86.98	81.54	86.98	90.69
C3	86.72	75.08	68.95	82.34	89.86	<b>91.33</b>	77.12	86.98	88.72	86.98	88.72	89.04
C4	74.76	66.50	70.12	84.72	85.63	<b>87.25</b>	75.88	82.69	79.94	82.69	79.94	85.87
C5	93.08	89.15	78.13	91.07	96.81	95.85	87.33	87.20	<b>97.30</b>	87.20	<b>97.30</b>	95.48
C6	92.08	93.19	86.97	95.11	97.35	96.34	90.90	96.96	<b>98.09</b>	96.96	<b>98.09</b>	96.66
C7	86.17	80.43	72.69	91.08	90.99	91.44	92.32	91.18	91.16	91.18	91.16	<b>94.13</b>
C8	96.43	95.14	89.84	98.13	98.56	98.78	98.53	95.84	<b>98.92</b>	95.84	<b>98.92</b>	97.91
C9	<b>97.32</b>	78.06	77.06	80.93	92.73	82.59	79.01	83.04	91.57	83.04	91.57	83.33
C10	75.79	74.25	72.81	86.36	<b>92.73</b>	92.08	81.54	83.16	88.45	83.16	88.45	90.58
C11	79.74	76.25	72.24	88.75	92.62	<b>94.84</b>	84.78	89.03	92.24	89.03	92.24	94.32
C12	82.60	74.59	73.96	81.03	87.10	90.46	84.88	73.69	85.12	73.69	85.12	<b>91.19</b>
C13	<b>97.82</b>	97.07	86.12	97.30	96.45	96.84	96.71	96.69	96.70	96.69	96.70	95.04
C14	92.70	91.68	88.05	96.12	98.59	98.23	96.14	94.28	99.05	94.28	99.05	<b>99.52</b>
C15	79.24	75.32	71.76	81.44	93.65	92.71	84.84	92.18	<b>95.37</b>	92.18	<b>95.37</b>	94.61
C16	98.27	97.24	88.84	96.38	<b>99.84</b>	99.71	99.05	98.52	99.54	98.52	99.54	99.41
OA	84.62	79.73	74.33	88.94	93.22	<b>94.04</b>	85.86	87.92	92.26	87.92	92.26	93.69
AA	87.91	83.03	75.67	88.54	92.94	<b>93.60</b>	87.56	88.34	92.64	88.34	92.64	92.72
Kappa (%)	82.39	76.85	71.44	87.39	92.26	<b>93.21</b>	83.89	86.22	91.18	86.22	91.18	92.81

Table 3.6: Classification results (in units of %) in Indian Pines Image

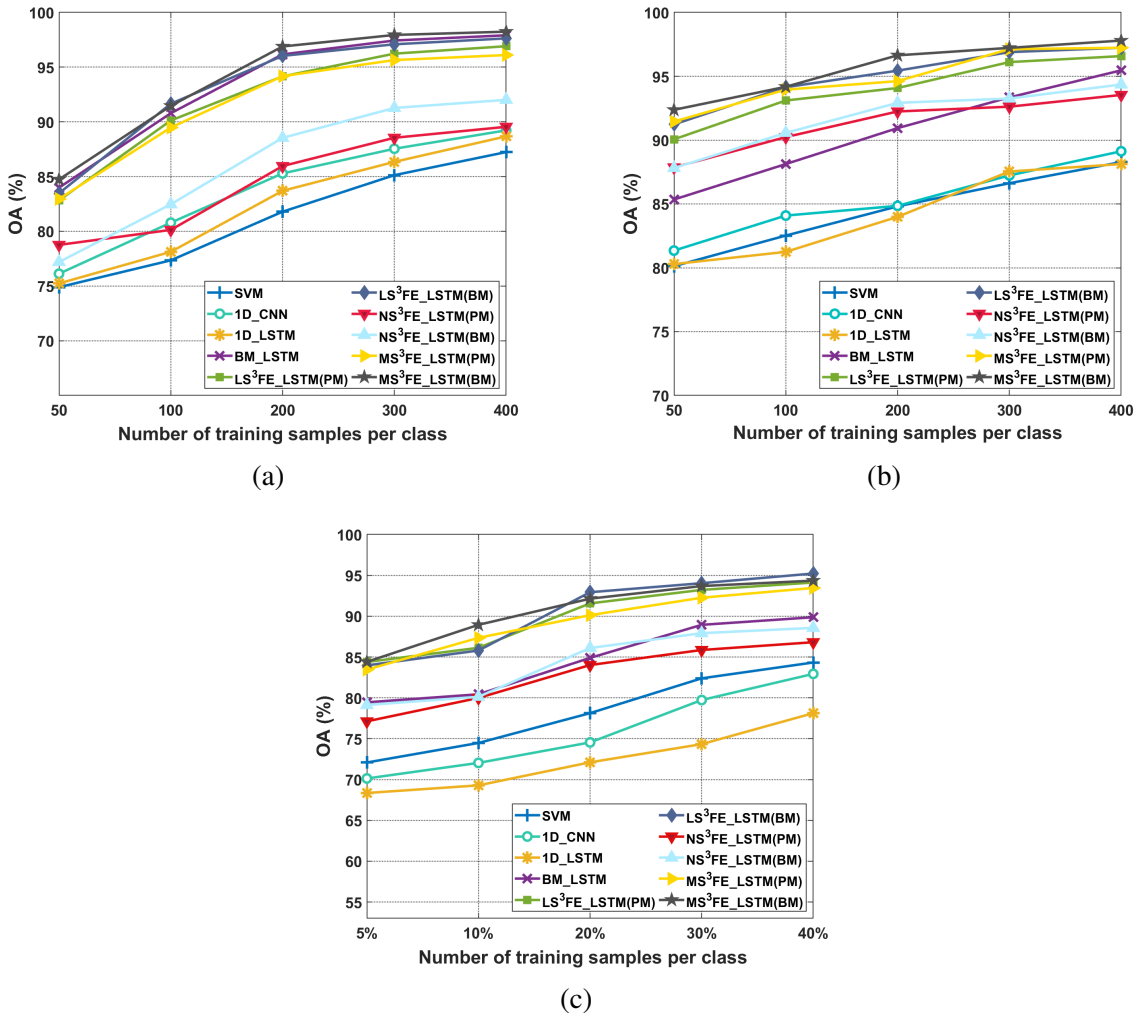
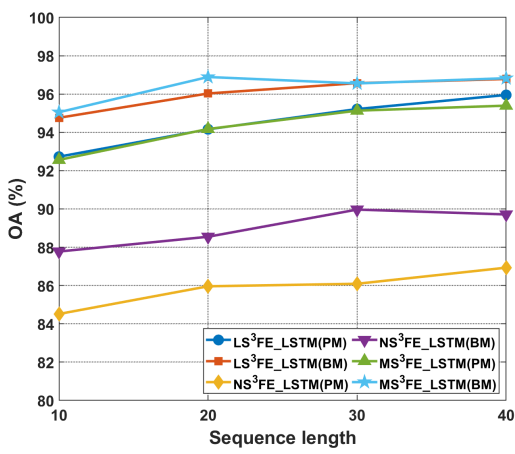
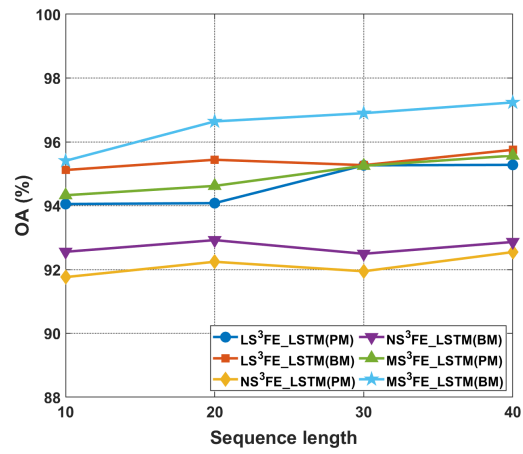


Figure 3.10: Classification accuracies of different models using different numbers of training samples per class on three experimental datasets: (a) PU image; (b) Salinas image; and (c) INP image

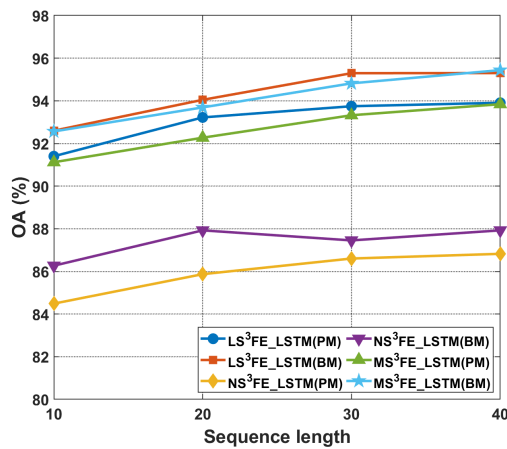




(a)



(b)



(c)

Figure 3.11: Classification accuracies of proposed SSFE-based models using different sequence lengths on three experimental datasets: (a) PU image; (b) Salinas image; and (c) INP image

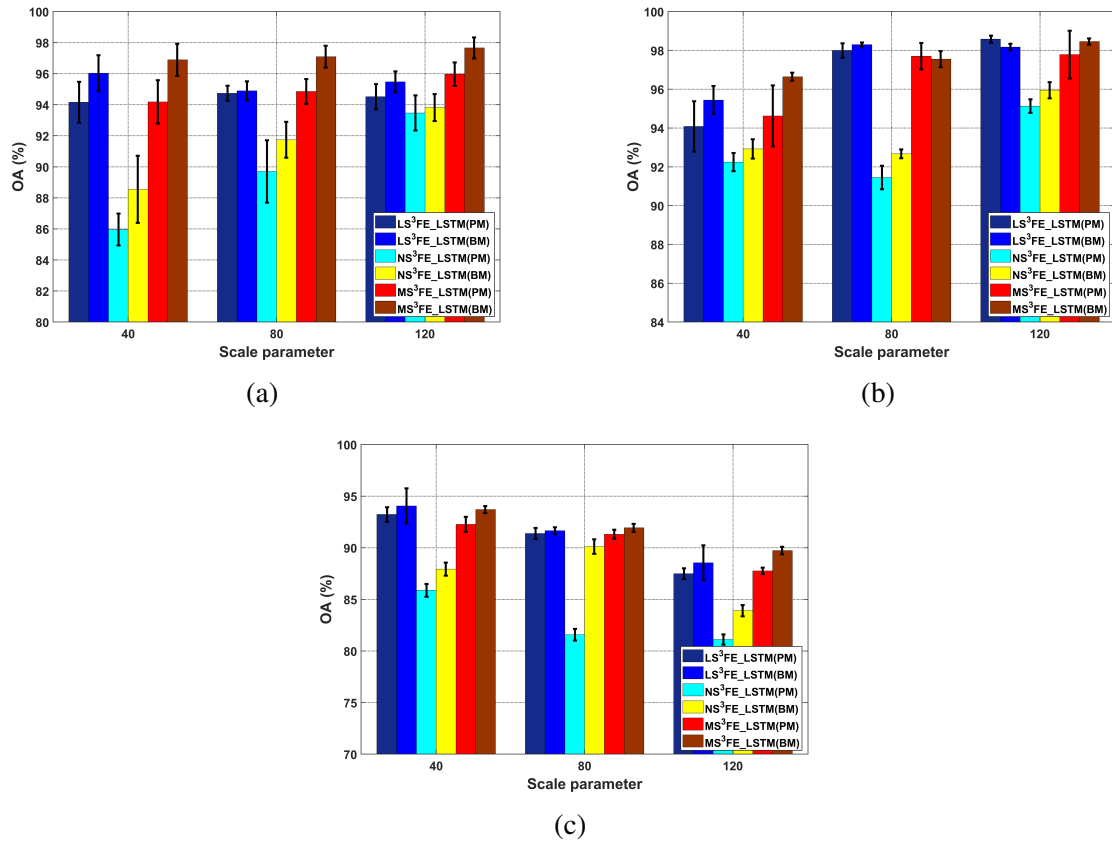


Figure 3.12: Classification accuracies of proposed SSFE-based models using segmentation map with different scale parameters on three experimental datasets: (a) PU image; (b) Salinas image; and (c) INP image

Table 3.7: Numbers of segments obtained using different scale parameter values, for each image

Dataset	$s_f=40$	$s_f=80$	$s_f=120$
PU	5924	1617	797
Salinas	694	218	131
INP	283	89	51

#### 3.4.4.1 Number of training samples

Regarding the number of training samples per class  $n_{tr}$ , two differently-sized sets of training samples were utilized for processing the three experimental HSIs. For the PU and Salinas images, five different values of  $n_{tr}$  ( $\{50, 100, 200, 300, \text{ and } 400\}$ ) are investigated. For the INP image, the five different values are set to be 5%, 10%, 20%, 30%, and 40%. The scale parameter  $s_f$  is fixed to be 40, and the sequence length  $l$  is 20. Fig. 3.10 shows the classification accuracies of all models utilized in this article using different numbers of training samples per class. It is clear that the OAs of all methods increase when the number of training samples increase. Meanwhile, for the PU image, the OAs for the proposed SSFE-based models become stable when  $n_{tr}$  is sufficiently large. However, for the Salinas and INP images, the increases in OAs for those SSFE-based approaches remain essentially the same across all different numbers of training samples.

#### 3.4.4.2 Sequence length

For the sequence length parameter  $l$ , four different sequence lengths ( $\{10, 20, 30, \text{ and } 40\}$ ) are utilized for performance evaluation. The scale parameter  $s_f$  is 40, and the number of training samples per class is set to 200 for the PU and Salinas images, and 30% for the INP image. Fig. 3.11 shows the classification accuracies accrued with all six proposed SSFE-based methods. For most methods, we observe that the classification accuracies tend to rise with increasing sequence length. However, such OA increases may not be applicable for all proposed models. For example, the OAs derived from MS<sup>3</sup>FE\_LSTM (BM) for the PU image decrease when decrementing the sequence length from 20 to 10. However, such a kind of decrease in OA is not significant, and is not exhibited with all proposed models applied to all experimental images.

Table 3.8: Average computational time cost (seconds) for sequential feature extraction from single line (row)

Data	BM_LSTM	Proposed SSFEs					
		PCA	FNEA	LS <sup>3</sup> FE_LSTM (BM)	NS <sup>3</sup> FE_LSTM (BM)	MS <sup>3</sup> FE_LSTM (BM)	MS <sup>3</sup> FE_LSTM (BM)
PU	4155.98	2.02	7.08 ( $s_f = 40$ )	31.53 ( $s_f = 40$ )	45.34 ( $s_f = 40$ )	34.64 ( $s_f = 40$ )	
			6.00 ( $s_f = 80$ )	42.10 ( $s_f = 80$ )	53.19 ( $s_f = 80$ )	44.01 ( $s_f = 80$ )	
			5.66 ( $s_f = 120$ )	50.47 ( $s_f = 120$ )	70.94 ( $s_f = 120$ )	68.78 ( $s_f = 120$ )	
Salinas	3522.94	2.30	8.38 ( $s_f = 40$ )	10.4 ( $s_f = 40$ )	10.82 ( $s_f = 40$ )	8.74 ( $s_f = 40$ )	
			7.09 ( $s_f = 80$ )	22.73 ( $s_f = 80$ )	26.43 ( $s_f = 80$ )	23.68 ( $s_f = 80$ )	
			6.91 ( $s_f = 120$ )	35.39 ( $s_f = 120$ )	39.13 ( $s_f = 120$ )	43.34 ( $s_f = 120$ )	
INP	2054.76	0.40	1.96 ( $s_f = 40$ )	1.98 ( $s_f = 40$ )	2.20 ( $s_f = 40$ )	1.42 ( $s_f = 40$ )	
			1.42 ( $s_f = 80$ )	6.35 ( $s_f = 80$ )	8.04 ( $s_f = 80$ )	6.36 ( $s_f = 80$ )	
			1.38 ( $s_f = 120$ )	10.27 ( $s_f = 120$ )	12.75 ( $s_f = 120$ )	11.45 ( $s_f = 120$ )	

### 3.4.4.3 Scale parameter in FNEA

When employing FNEA in our proposed methods, the scale parameter  $s_f$  plays an important role since it will influence the average size of the extracted segments. In other words, too large of an  $s_f$  will introduce under-segmentation problems, where too many pixels with relatively low similarities will be incorporated into segments. In contrast, too small of an  $s_f$  value will yield an over-segmented segmentation map, where too few pixels will be selected to generate individual segments. In this article, three different  $s_f$  values ( $\{40, 80, \text{and } 120\}$ ) are investigated, and their corresponding classification accuracies acquired by the six proposed models are shown in Fig. ???. The number of segments obtained from FNEA using different  $s_f$  are illustrated in Table 3.7 as well. The sequence length  $l$  is fixed to be 20, and the number of training samples per class is set to 200 for the PU and Salinas images and 30% for the INP image. For the PU and Salinas images, the obtained OAs increase when the scale parameter value increases, and the best classification results for each approach are obtained at the largest  $s_f$ . Therefore, for those two HSIs, larger scale parameter values perform better in characterizing both segment-wise and pixel-wise similarity measurements and yield higher OAs. However, for the INP image, we observe that OAs decrease when the values for the scale parameter  $s_f$  increase, except for the NS<sup>3</sup>FE\_LSTM (BM) method when  $s_f$  is 80. Due to the low spatial resolution (20 m) and limited spatial dimensions ( $145 \times 145$  pixels), a larger  $s_f$  will generate an over-segmented segmentation map, where less similar/relevant pixels will be incorporated, which will reduce the expressiveness of a given segment-based feature and extracted sequential feature.

### 3.4.4.4 Computational time cost for SSFE

To evaluate the computational time cost improvements of our proposed methods compared with our previous work [83], such as the BM\_LSTM method which has been investigated in the previous sections, the average time costs of extracting sequential features from a single line (row) of a given image are calculated since such computation is the most time-consuming step in the proposed methods, and those results are shown in Table 3.8, where three BM-based methods uti-

lizing three SSFE strategies are selected for comparison. The computational time cost of our proposed methods consists of three components, including the computation of PCA, computation of FNEA, and sequential feature extraction. All of those experiments are conducted on a workstation equipped with a 3.2-GHz Intel(R) core i7-8700 CPU, 16GB RAM, and utilizing Python 3.5 as the programming language. It is clear that BM\_LSTM proposed in our previous work [83] entails enormous time costs across the three experimental HSIs. However, such time costs are reduced significantly, by more than one hundred times, with the three newly-proposed models tested here, after combining the time costs of PCA, FNEA, and the corresponding SSFEs. For the INP image, compared with the time cost resulting from the application of BM\_LSTM, the best time-cost reduction is greater than 540 times, a performance obtained by MS<sup>3</sup>FE\_LSTM (BM). Moreover, from Table 3.8, we can see that within each SSFE method, the computational time cost increases when  $s_f$  increases, but at the same time, the time cost of FNEA decreases. Utilizing larger  $s_f$ , FNEA will yield fewer segments with less associated time cost, but the searching time for a given SSFE will increase due to the increasing number of pixels in those segments. Therefore, choosing an appropriate  $s_f$  to achieve a balance between the time cost of the FNEA and SSFE methods is a key point in minimizing the total computational cost.

### 3.5 Conclusion

In this article, a fast single-image sequential feature extraction (SSFE) framework for LSTM-based HSI classification is proposed, where object-based segmentation is utilized to accelerate sequential feature construction. Specifically, similar pixels which are employed for sequential features are not selected from the whole-image scope, but are rather chosen from individual segments within the segmentation map. Three different SSFE-based strategies are developed, where local and non-local segments are considered in a separate and combined manner, respectively. Quantitative and qualitative classification results illustrate that, for the PU and Salinas images, MS<sup>3</sup>FE\_LSTM (BM) achieves the best classification performance, where both local and non-local similar segments are utilized for pixel-wise similarity measurement and sequential feature construction. Such experimental results demonstrate the capability and effectiveness of combining

both the local segment, which contains the target pixel, and the non-local segment with the highest similarity score when searching similar pixels for sequential feature construction. Instead of only using the local segment, the incorporation of non-local-segment information can also aid in selecting similar pixels, which will tend to make the extracted sequential feature more representative, likely leading to improved classification performance. For the INP image, the best classification performance is obtained by LS<sup>3</sup>FE\_LSTM (BM), instead of MS<sup>3</sup>FE\_LSTM (BM), which indicates that for the INP image, the local segment makes a greater contribution to the similar-pixel calculation and selection. This means that, for INP image, similar pixels for/associated with a given target pixel will be more likely to be located in the local segment. Introducing non-local segment will make the algorithm select less similar pixels than that of only utilizing local segment. By investigating the computational time cost of our proposed methods, we find that even though the proposed methods may not, in some cases, achieve a better classification performance than that of the best model proposed in [83] (e.g., the two LS<sup>3</sup>FE\_LSTM methods perform worse than BM\_LSTM on the PU image), those approaches are still very competitive, accuracy-wise, and they are capable of achieving acceptable classification performance with markedly lower computation cost.

In the future, more experiments will be conducted using other segmentation algorithms to further investigate the influence of different segments on sequential feature extraction. Additionally, future work will involve more applications of our proposed methods on large-scale remote-sensing images and aerial images with very high-spatial resolution.

## 4. CHARACTERIZATION OF LAND COVER INFORMATION USING SINGLE-IMAGE BASED RECURRENT NEURAL NETWORK AND LANDSAT IMAGERY: A CASE STUDY IN NORTHERN INDIA<sup>1</sup>

### 4.1 Introduction

As a predominant terrestrial ecosystem of the Earth, forest plays an important role in ecosystem services, including serving as a carbon sink, generating oxygen from carbon dioxide, alleviate nature hazards, and serving as a genetic reserve. Therefore, forest managements are essential for various domains such as biological diversity protection, soil and water conservation, and Carbon fixation. An accuracy and detailed characterization of forest is always required for forest prevention and management. However, those detailed information (e.g., species distribution and composition) is difficult and time consuming to obtain from large scale forest areas by using ground inventory [93]. Remote sensing is a useful technique to analyze forest in a large-scale manner, particularly for those areas with limited accessibility. Abundant studies have been carried out in this field, investigating the capabilities of different remote sensing produces collected from various sensors and platforms. Datasets collected from different platforms with different sensors can provide different information/features with their own peculiarities to analyze forest. Among them, Satellite imagery has been becoming an important data resource for large-scale observation and monitoring due to its relatively low cost, large observation area, and revisitable ability, and has been introduced successfully in forest analysis. The spectral features captured from satellite-based sensors are the critical information that can help us distinguish different objects within forest. And classification, which is a common machine learning based technique to determine the category of a pixel within RS image, has been utilized to process satellite RS image in order to obtain tree species information and biophysical parameter of forest.

---

<sup>1</sup>Part of the data reported in this chapter is reprinted with permission from “Limited Effects of Tree Planting on Forest Canopy Cover and Rural Livelihoods in Northern India” by Eric A Coleman, Bill Schultz, Vijay Ramprasad, Harry Fischer, Pushpendra Rana, Anthony M. Filippi, Burak Güneralp, Andong Ma, Claudia Rodriguez Solorzano, Vijay Guleria, Rajesh Rana, and Forrest Fleischman, 2021. *Nature Sustainability*, 59(7), <https://doi.org/10.1038/s41893-021-00761-z>, ©2021 Nature Portfolio.



Conventional multispectral remote sensing images have been widely exploited in the past years for forest analysis. [94] utilized the iterative self-organizing data analysis technique (ISO-DATA), an unsupervised classification method for deciduous forest classification. Satellite images are collected from Landsat 5 Thematic Mapper (TM) sensor. Within such proposed scheme, other features including normalized difference vegetation index (NDVI) and Tasseled Cap brightness, greenness, and wetness are incorporated as well to generate input data. [95] proposed a stratified forest estimation framework using k-nearest neighbor (k-NN) as classifier. The satellite images utilized in this study are obtained from Landsat 7 Enhanced Thematic Mapper Plus (ETM+) sensor. [96] developed a workflow for Amazonian rain forest classification by employing both Landsat 7 ETM+ data and Shuttle Radar Topography Mission (SRTM) elevation model, where elevation information is considered as ancillary data source. During its classification procedure, linear discriminant analysis (LDA) is utilized for both feature extraction and classification. For those aforementioned studies, they are all constructed for pixel-level classification where each pixel is assigned a predefined label. Consequently, “salt-and-pepper” noise, which is taken as discrete misclassification points, is obvious. In order to alleviate such a phenomenon, benefited from the improvement of spatial resolution of satellite remote sensing produces, segmentation-based approaches are introduced in remote sensing community for forest classification. Segmentation-based algorithms focus on grouping homogeneous pixels in a local area to create pixel cluster, which is named “segment”. For those pixels within the same segment, they share similar spectral features and always be regarded as a whole during classification. Such a strategy can alleviate “salt-and-pepper” phenomenon significantly. [97] utilized eCognition [90] software to create segmentation map from high resolution image generated from the fusion of multispectral and panchromatic images. Then nearest neighbor classifier is applied on segmentation map for forest type classification. [98] proposed an object-oriented classification scheme for forest ecosystem classification. High spatial resolution QuickBird images are used in this study. Regarding feature representation for segmentation map, it combines spectral feature, vegetation indices, and text feature created by grey-level co-occurrence matrix (GLCM). [99] adopted random forest as classifier for tree species classifica-

tion problem. It utilized WorldView-2 satellite images as main data source and characterized tree species information from both pixel and object levels. Similar research can be found in [100] as well. The only difference is that [100] utilized digital elevation model (DEM) and its derived slope and aspect layers as additional datasets.

In recent decade, deep learning, a novel machine learning scheme developed from traditional neural network, have been attracting more attentions due to its promising performance on remote sensing data analysis. Different from those conventional approaches which heavily rely on hand-craft features, deep learning-based models extract more complex and higher-level features in a hierarchical manner. The applications of deep learning in remote sensing community involve land-use and land-cover characterization [101][33], object detection [102][103], scene classification [104][105], image super resolution [106][107], and multi-source data fusion [108][109]. For forest classification, some investigations have been applied using deep learning models. [110] proposed a 1-dimensional convolutional neural network (1DCNN) based model to classify forest. The Sentinel-2 satellite images are used as data source. And various spectral and spatial features are extracted, including NDVI, brightness, GLCM homogeneity, and rectangular fit. [111] developed a data fusion and classification framework in order to fuse light detection and ranging (LiDAR) data and high resolution satellite images. Within this model, two separate deep neural networks are adopted to process two different datasets, and they are merged at specific layer to output classification results. [112] proposed a deep learning-based model, named ForestNet to classify drivers of deforestation on Landsat 8 imagery. It uses conventional CNN as its backbone neural network.

In this research, we focus on a specific deep learning model, recurrent neural network (RNN), for remote sensing image classification. Different from traditional feedforward neural network, RNNs are able to process the sequential inputs by having a recurrent hidden state whose activation at each step depends on that of the previous step. In other words, RNNs is capable to encode dependencies between data at different steps, which provide capacity to handle sequential data properly compared with other machine learning models. And such a property leads its successful application in natural language processing (NLP). For remote sensing sub-field, researchers have proposed

various algorithms for its applications on multi-temporal remote sensing datasets. [113] exploited recurrent convolutional neural network (RCNN) to encode spectral-spatial-temporal features and to detect changes in multispectral images. [44] adopted patch-based RNN for multi-temporal image classification. However, as the availability of multi-temporal images are limited due to the intrinsic restriction of remote sensing platforms (e.g., cloud-cover condition, limitation of light, and revisit cycle), some single-image-based RNN classification models are developed as well in order the aforementioned issues [45][46][81].

From the aforementioned investigations, we can see that the popular strategy to apply RNN on single-image-based classification problem is to take spectral feature of an individual pixel as its sequential feature. However, the dependencies between different bands are characterized in a “sequential” manner starting from the first band to the last band, where the representative reflectance information collected from specific bands may be ignored. In addition, spatial contextual information, which have been demonstrated its significant performance to promote classification accuracy, is not utilized in those proposed models. More importantly, those approaches are mainly developed for hyperspectral remote-sensing image classification, where hundreds of spectral bands can be obtained easily. For multi-spectral remote-sensing image, only limited spectral channels are captured which brings difficulties for RNNs application. And if those images are collected from different sensors, their spectral bands or the number of spectral bands may not be the same, where a single RNN-based model can not be applied to all those images due to the inconsistency of input channels. Inspired on our previous research [114], a modified single-image-based sequential feature extraction (SSFE) and classification using RNN is employed. Similar with [114], segmentation map will be generated first by utilizing fractal net evolution approach (FNEA). Then spectral and shape features of those segments are obtained for segment-based similarity measurement. After determining similar segments, pixel-based similarities between those pixels within selected similar segments and target pixel will be calculated. Lastly, sequential feature of the target pixel will be presented by those similar pixels selected based on similarity indices. This is the first application of

The rest of this section is organized as follows: Section 5.2 describes study area and datasets used in our analysis. Section 5.3 presents the classification methods. Experimental results and discussions are provided in Section 5.4. Finally, Section 5.5 draws the conclusion of this study.

## **4.2 Material**

### **4.2.1 Study site and field data**

For this study, the study area is located in the Kangra district of Himachal Pradesh in the northern India. The total area of Kangra district is 5737.41 square kilometers. As tree plantations have a long history in Kangra district, we will focus on the land-cover classification within plantation areas. In this research, 179 plantations are collected and created. And ground reference data are collected simultaneously within those plantations, which will be utilized for classification model training and evaluation. Here four different land-cover classes are defined, including “Pure Needle Leaf”, “Pure Broad Leaf”, “Mixed Forest”, and “Pasture”. Note that “Mixed Forest” consists of two sub-classes, “Mixed Predominantly Needle Leaf” and “Mixed Predominantly Broad Leaf”. Detailed explanation about how to create training and testing samples are provided in Section 4.2.3 and 4.4.1. The spatial extent is shown in Figure 4.1.

### **4.2.2 Landsat image acquisition and preprocessing**

Satellite images collected from Landsat platform. Since the objective of this study is to analyze the land cover information in spring (March to May) from 1990 to 2018, all images obtained from such a time period have been checked by accessing the EarthExplorer (<https://earthexplorer.usgs.gov/>). The primary factor that needs to be considered is the cloud cover (CC). In this study, images with CC over 5% will be ignored. In the meanwhile, we need to make sure that all Kangra district area will be covered. Therefore, two Landsat scenes (path/row: 148/38, and path/row: 147/38) are selected, and five years, including 1991, 1993, 1996, 1998, 2009, and 2018, are chosen. Table 4.1 illustrates detailed information about those selected images. Note that one image is collected from 1991 and another image is obtained from 1992. The reason is that for the right scene (path/row: 148/37) at 1991, there is no image that meet the requirement of cloud cover. Therefore, we choose

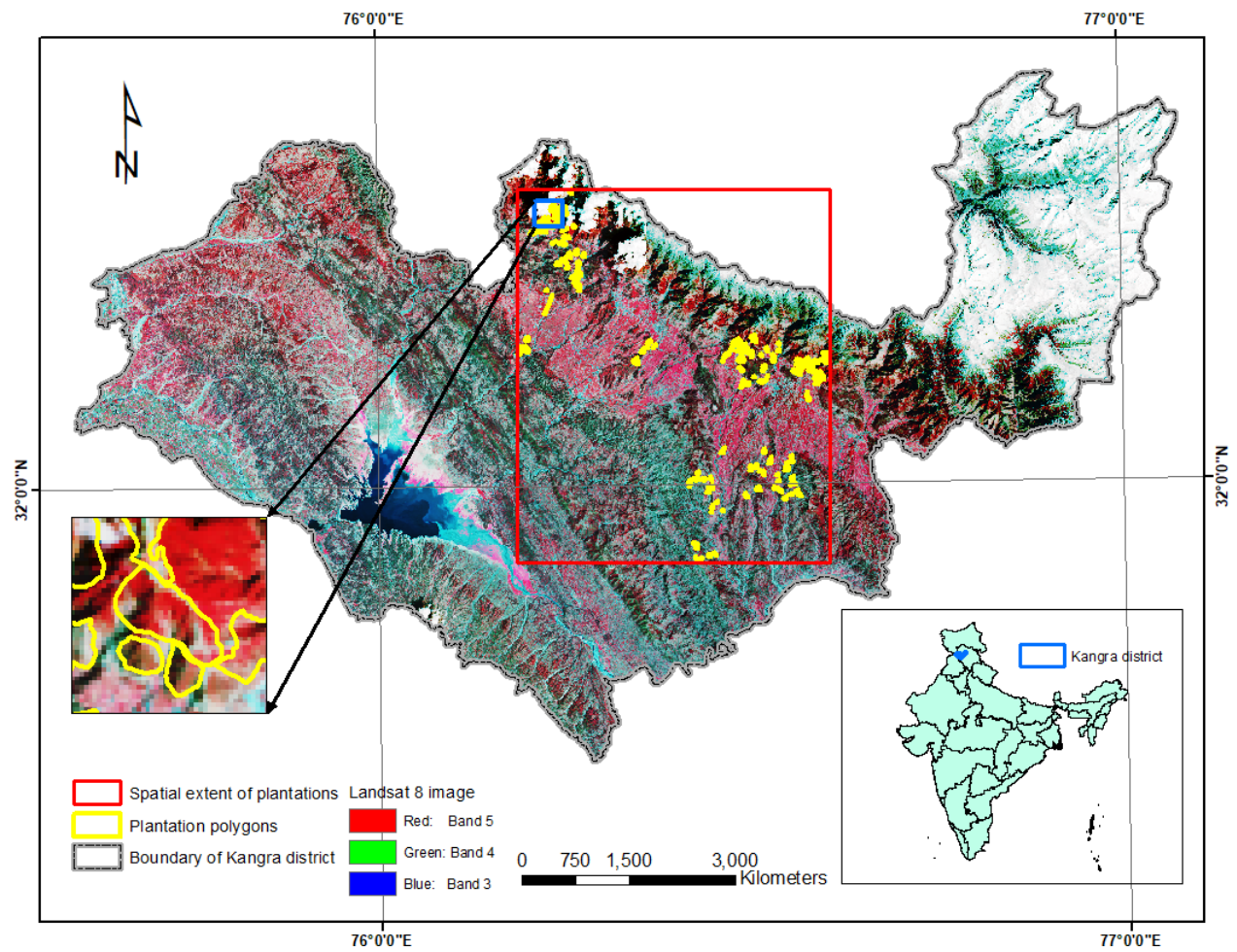


Figure 4.1: The study area. Base image is collected from Landsat 8 where false color composite is applied (R: Band 5, G: Band 4, and B: Band 3)

the image of 1992 collected from similar season.

Sensor	Acquisition date	Path	Row	Cloud cover	Processing level
Landsat 5 TM	05/15/1991	148	38	2.00%	L1TP
Landsat 5 TM	05/10/1992	148	37	2.00%	L1TP
Landsat 5 TM	05/04/1993	148	38	1.00%	L1TP
Landsat 5 TM	04/27/1993	148	37	2.00%	L1TP
Landsat 5 TM	04/26/1996	148	38	0.00%	L1TP
Landsat 5 TM	04/03/1996	148	37	3.00%	L1TP
Landsat 5 TM	04/16/1998	148	38	1.00%	L1TP
Landsat 5 TM	05/27/1998	148	37	3.00%	L1TP
Landsat 5 TM	04/30/2009	148	38	1.00%	L1TP
Landsat 5 TM	05/09/2009	148	37	3.00%	L1TP
Landsat 8 OLI	04/07/2018	148	38	2.56%	L1TP
Landsat 8 OLI	03/31/2018	148	37	1.41%	L1TP

Table 4.1: Information of acquired images. “TM” is Thematic Mapper, and “OLI” represents Operational Land Imager

Besides of those satellite images, DEM is utilized as another data source in this study. DEM data is collected from SRTM whose spatial resolution is 28.83 meters. In order to cover the whole area of Kangra district, six DEM tiles are utilized and mosaiced into a single DEM file for the following steps. Since the spatial resolution (28.83 meters) of SRTM DEM data is different from that of Landsat 5 and 8 images (30 meters). DEM needs to be resampled into 30 meters raster to make sure that both datasets share the same cell size. In this study, nearest neighbor is employed to resample DEM data where minimized changes will be applied to pixel values as no new values are created.

As those images are in L1TP level, atmospheric correction is needed to remove the effects of the atmosphere. FLAASH module embedded in ENVI software is utilized for atmospheric correction. After obtaining atmospheric correction results, seamless mosaic tool in ENVI is applied on two images obtained from two different scenes but within the same year. For the “Input Images” parameter of seamless mosaic tool, the image collected from left scene (path/row: 148/38) is

defined as “Adjust”, and the one obtained from right scene (path/row: 148/37) was defined as “Reference”.

### 4.2.3 Ground reference data generation

As mentioned in Section 4.2.1, four different classes are defined for land-cover classification purpose in this study area, which are “Pure Needle Leaf” (class 1), “Pure Broad Leaf” (class 2), “Mixed Forest” (class 3), and “Pasture” (class 4). Those training samples are selected only from those plantations, and generated accordingly based on the mosaic image of each year. Table 4.2 shows detailed information regarding those labeled samples for different dates.

Acquisition date	Class 1	Class 2	Class 3	Class 4	Total
1991	1880	1759	3973	288	7900
1993	1880	1759	3973	288	7900
1996	1880	1759	3973	288	7900
1998	1933	1759	4147	288	8127
2009	2038	2153	4496	301	8988
2018	2058	2367	4778	314	9517

Table 4.2: Number of pixels for ground reference data

### 4.3 Single-image-based sequential feature extraction and classification

In this study, the classification method proposed in [114] is employed. Regarding the utilization of RNN on remote sensing image classification, the key issue is to find/extract sequential feature representation of a given individual pixel. Figure 4.2 exhibits the workflow of that single-image-based sequential feature extraction (SSFE) and classification scheme. In the first step, we combine Landsat image and DEM into a single raster input where DEM is stacked after the last band of Landsat image. Then the FNEA algorithm is applied on that resulting raster to create segmentation map. There are three main parameters used in FNEA, which are scale, compactness, and shape parameters, to control if two segments will be merged or not by characterizing the heterogeneity of two segments before and after merging them. Once the segmentation map is obtained, segments-

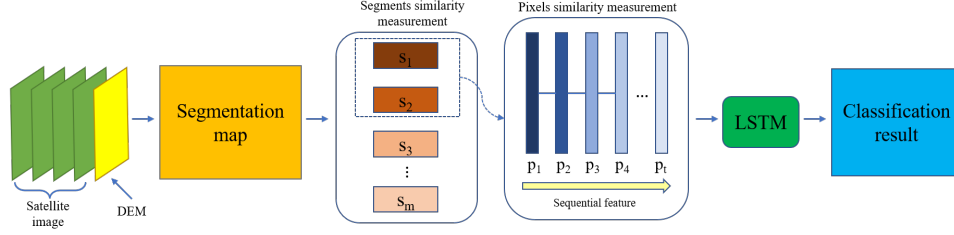


Figure 4.2: Single-image-based sequential feature extraction and classification framework

based similarity calculation will be implemented. Specifically, given a target pixel  $p_t$  and its corresponding segment  $s_t$  that contains  $p_t$ , the similarities between  $s_t$  and all other segments will be calculated by utilizing Euclidean distance (EU). The EU between  $s_t$  and  $s_i$  is defined as follows:

$$d_{EU}(s_t, s_i) = \sqrt{\sum_{k=1}^{b_s} (s_t^k - s_i^k)^2} \quad (4.1)$$

where  $b_s$  represents the number of segment features, and  $s_t^k$  and  $s_i^k$  denote  $k$ th feature for  $s_t$  and  $s_i$ , respectively. For this study, segment features are constructed based on spectral features and geometry features. Spectral feature consists of mean value, standard deviation, and brightness of each band. Geometry feature includes area, asymmetry, border length, compactness, density, main direction, number of pixels, ratio between length and width, and rectangular fit [90].

Once we obtain the segment-based similarities, similar segments will be selected. And pixel-based similarity measurement are calculated between  $p_t$  and all pixels within those selected segments. In this study, block matching (BM) and spectral angle mapping (SAM) [83] are utilized to measure distance/similarity between two pixels. BM characterizes similarity between two individual pixels by exploiting their neighboring pixels located within a fixed-size window, and its effectiveness of sequential feature construction and classification has been demonstrated in [83]. Lastly, the first several similar pixels are chosen and re-ordered based on calculated similarity values in a descending manner to construct the sequential feature of  $p_t$ . After obtaining those sequential features, they will be fed into the RNN-based classification model.



## 4.4 Experiments

In this section, we apply the single-image based sequential feature extraction approach and RNN-based classification model on those six “satellite+DEM” images. To evaluate its performance, we also compare it within 2DCNN [37] classification model. Moreover, we conduct a sensitivity analysis of those hyperparameters utilized in that sequential feature extraction method.

### 4.4.1 Experimental setup

As discussed in Section 5.3, the satellite image and DEM raster is combined into a single raster. To better characterize the classification results within plantation and reduce computational time cost, the minimum bounding box of all plantation polygons is generated and is employed to clip input raster. The spatial dimension of final raster is  $1577 \times 1324$ .

For the 2DCNN model, its input needs to be a 2D image patch with one or multiple channels. To classify individual pixels from those images, a sliding window is applied to all pixels to extract image patches. The size of input image patch is  $5 \times 5$ . Regarding the 2DCNN architecture, it follows the one proposed in [37]. There are two convolutional layers, two batch normalization layers, two dropout layers, one fully connected layers, and one classification layer which utilized softmax as the activation function. For the first convolutional layer, it has 21 trainable convolutional filters of dimension  $3 \times 3$ . In the second convolutional layer, 42 convolutional filters with the dimension of  $3 \times 3$  are adopted. Batch normalization layer and dropout layer are added right after each convolutional layer, and the dropout rate is 0.3. The dimensions of those two fully connected layers are 42 and 4 (number of classes), respectively.

Within SSFE step, parameter values for the scale, compactness, and shape are set to 50, 0.5, and 0.1, respectively when generating segmentation map using FNEA. And only one similar segment is selected. Regarding the RNN classification, we use long short-term memory (LSTM) [42] which is an improved version of RNN as classifier. Its architecture consists of four LSTM layers and two fully connected layers. The dimensions of those four LSTM layers are 32, 64, 128, and 256, respectively. And the first fully connected layer has 50 hidden units.

For the LSTM-based classification model training, it is implemented using Keras [91] and TensorFlow [92]. The optimizer used in training step is Adadelata [115]. Batch size is 64, and the number of epochs is 500. All experiments regarding 2DCNN and LSTM model training are conducted on the Texas A&M High Performance Research Computing.

For the aforementioned two classification approaches, the ground reference data is split into training and testing sample randomly. And the ratio between them is 1:1, which means that 50% of samples per class are selected randomly as training samples and the remaining samples are utilized as testing samples. Moreover, 10 replications of experiments using such a random-selection strategy are applied in order to avoid any bias introduced from sampling procedure. To evaluate classification performance quantitatively, overall accuracy (OA), average accuracy (AA) and Kappa coefficient (Kappa) are exploited in this study. OA is the ratio between the number pixels that are classified correctly and the total number of pixels. AA represents the average accuracy calculated across all classes. Kappa is obtained from confusion matrix and is utilized to indicate the extent of agreement between ground truth and predicted results. All those three accuracies are obtained by averaging corresponding accuracies across 10 replications.

#### **4.4.2 Classification results**

The classification results are display from Table 4.3 to Table 4.8. We can find that OAs obtained from SSFE-based LSTM models are higher that those from 2DCNN methods across all six datasets. The one that increases the most is obtained from 1998-pair image from 82.50% to 85.48%. And the one that increases the least is collected from 1993-pair image from 84.73% to 85.06%. Moreover, not all accuracies of those four classes are improved simultaneously. For example, for the 1991-pair image, the accuracies of classifying “Pure Needle Leaf” and “Pure Broad Leaf” using SSFE-based LSTM decrease compared with those from 2DCNN. However, the accuracy of “Mixed Forest” increases significantly frmm 84.62% to 87.03%. Similar phenomenon can be observed from other images. As mentioned in 4.2, even though different image shares different training samples, “Mixed Forest” still has the most labeled samples. Therefore, SSFE-based LSTM may not achieve a good performance on “Pure Needle Leaf”, “Pure Broad Leaf”, and “Pas-

ture”, but presents better result on “Mixed Forest”. Therefore the final OA is always better than that of 2DCNN across all images. Figure 4.3 illustrates classification results of 2019-pair image.

Class name	2DCNN	SSFE-based LSTM
Pure Needle Leaf	<b>86.01±2.50</b>	84.37±1.79
Pure Broad Leaf	<b>84.65±2.99</b>	82.49±1.86
Mixed Forest	84.62±1.84	<b>87.03±1.13</b>
Pasture	85.93±3.45	<b>88.97±3.83</b>
OA	84.91±1.38	<b>85.43±1.03</b>
AA	85.30±0.66	<b>85.71±2.48</b>
Kappa	76.10±2.17	<b>77.19±1.64</b>

Table 4.3: Classification results (in units of %) in the 1991-pair image

Class name	2DCNN	SSFE-based LSTM
Pure Needle Leaf	<b>86.61±3.73</b>	85.22±2.16
Pure Broad Leaf	<b>83.79±3.01</b>	81.76±1.80
Mixed Forest	84.41±1.29	<b>86.56±1.82</b>
Pasture	<b>86.54±5.26</b>	83.19±5.22
OA	84.73±0.88	<b>85.06±1.77</b>
AA	85.34±1.25	<b>85.58±1.84</b>
Kappa	75.80±1.33	<b>76.54±2.88</b>

Table 4.4: Classification results (in units of %) in the 1993-pair image

In order to evaluate the influence of different hyperparameters utilized in SSFE-based LSTM, parameter sensitivity analysis is conducted on two hyperparameters: scale parameter used in FNEA segmentation algorithm *scale*, and the number of similar segments selected  $n_{seg}$ . Utilizing smaller *scale* will aggregate less pixels into a single segment where the similarities between those pixels are relatively high. A larger *scale* will generate larger segments which means that the total number of segments is decreased compared with that of using smaller *scale*. However, that means more pixels with lower similarities will be incorporated into segments. In this study, two different

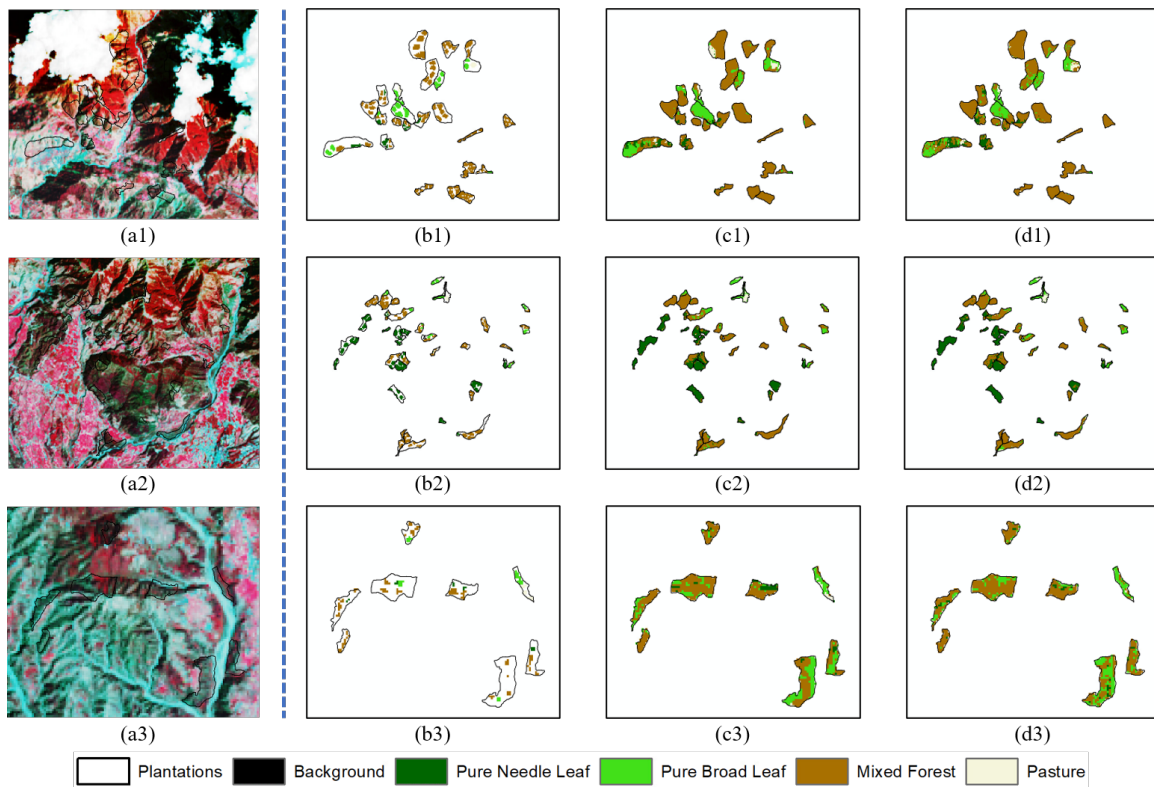


Figure 4.3: Classification results obtained from 2019-pair image. (a1), (a2), and (a3) are false color composite of 2018-pair image collected from three different areas. (b1), (b2) and (b3) are training samples. (c1), (c2) and (c3) are classification results collected from 2DCNN model. (d1), (d2) and (d3) are classification maps generated from SSFE-base LSTM approach

Class name	2DCNN	SSFE-based LSTM
Pure Needle Leaf	82.89±2.66	<b>85.23±1.48</b>
Pure Broad Leaf	<b>83.27±3.05</b>	82.87±1.51
Mixed Forest	82.67±2.62	<b>86.91±0.70</b>
Pasture	85.05±3.68	<b>87.07±2.44</b>
OA	82.73±1.02	<b>85.62±0.59</b>
AA	83.47±0.94	<b>85.52±1.69</b>
Kappa	72.49±1.95	<b>77.44±0.94</b>

Table 4.5: Classification results (in units of %) in the 1996-pair image

Class name	2DCNN	SSFE-based LSTM
Pure Needle Leaf	<b>85.51±4.01</b>	83.81±1.07
Pure Broad Leaf	75.13±5.07	<b>82.70±1.60</b>
Mixed Forest	84.69±1.69	<b>87.33±0.87</b>
Pasture	86.95±6.69	<b>87.86±4.24</b>
OA	82.50±1.29	<b>85.48±0.54</b>
AA	83.07±4.65	<b>85.42±2.21</b>
Kappa	72.47±2.06	<b>77.13±0.86</b>

Table 4.6: Classification results (in units of %) in the 1998-pair image

Class name	2DCNN	SSFE-based LSTM
Pure Needle Leaf	<b>85.38±3.35</b>	83.95±1.20
Pure Broad Leaf	<b>82.83±2.41</b>	81.05±1.38
Mixed Forest	82.81±1.29	<b>85.02±0.94</b>
Pasture	<b>82.40±3.87</b>	81.98±3.64
OA	83.25±0.87	<b>83.72±0.80</b>
AA	83.36±1.18	<b>83.68±1.57</b>
Kappa	73.50±1.33	<b>74.46±1.28</b>

Table 4.7: Classification results (in units of %) in the 2009-pair image

*scale* (50, and 100) are tested. Regarding the parameter  $n_{seg}$ , it will influence how many pixels will be extracted for the further pixel-based similarity measurement. Two different  $n_{seg}$  values, 1 and 10, are investigated in the experiments. Table 4.9 illustrates the experimental results using different combination of those two parameters. For different scale parameter *scale*, we observe

Class name	2DCNN	SSFE-based LSTM
Pure Needle Leaf	80.75±2.62	<b>81.11±1.89</b>
Pure Broad Leaf	<b>84.25±2.91</b>	79.92±1.34
Mixed Forest	81.19±2.13	<b>83.92±1.00</b>
Pasture	<b>87.71±3.29</b>	83.07±3.96
OA	81.78±0.79	<b>82.28±0.93</b>
AA	83.48±2.79	<b>83.79±1.57</b>
Kappa	70.95±1.48	<b>72.13±1.43</b>

Table 4.8: Classification results (in units of %) in the 2018-pair image

that the accuracies decrease when  $s$  increase. And regarding the other parameter  $n_{seg}$ , similar phenomenon can be found where the higher accuracies are always obtained using smaller  $n_{seg}$ . Those results demonstrate that using smaller  $scale$  and  $n_{seg}$  can achieve better classification performance. For an individual segment, large  $scale$  introduces more pixels with low similarities which will reduce the expressiveness of the extracted sequential features. Using larger  $n_{seg}$ , where more similar segments are incorporated for the similar pixel measurements and extraction, will yield similar issue, resulting in lower classification accuracies.

#### 4.5 Conclusion

In this study, we apply the single-image-based sequential feature extraction (SSFE) framework proposed in [114] for LSTM-based land-cover classification. This is an further extension of our proposed approach from the utilization on benchmark dataset to the large-scale satellite image classification problem. For this research, the study site is located in the rural area in the northern India, and the objective is to classify/identify land-cover information within plantations. Landsat images collected from six years are utilized as base data source. Experimental results illustrate that SSFE-based LSTM classification approach achieves better performances compared with the results obtained from 2DCNN. The parameter sensitivity analysis is adopted as well to evaluate the influence of using different segmentation maps and similar segments. We find that smaller scale parameter and less number of similar segments will produce higher accuracy, which illustrates that incorporating more pixels by either increasing scale parameter when generating segmentation map

Acquisition date	2DCNN	SSFEE-based LSTM					
		scale: 50, n_seg: 1		scale: 100, n_seg: 1		scale: 100, n_seg: 10	
1991pair	84.91%	85.43%	82.37%	81.10%	79.90%		
1993pair	84.73%	85.06%	82.37%	81.05%	80.40%		
1996pair	82.73%	85.62%	82.11%	78.35%	79.54%		
1998pair	82.50%	85.48%	81.54%	81.01%	77.16%		
2009pair	83.25%	83.72%	79.63%	80.56%	77.71%		
2018pair	81.78%	82.28%	79.32%	79.93%	77.38%		

Table 4.9: Classification accuracies of different models using different parameters

or adding more similar segments will reduce the expressiveness of the extracted sequential features and thus lower the classification accuracies.

Future work will focus on more detailed classification system by involving canopy-cover classes. And utilizing high-resolution satellite images will be investigated as well to further improve accuracy. Additionally, other spectral features and texture features derived from the original image can be employed as ancillary data source.



## 5. FALLEN TREE DETECTION USING HIGH-RESOLUTION AERIAL PHOTOS AND DEEP LEARNING IN A COASTAL RIVER AND ITS FLOODPLAIN

### 5.1 Introduction

Coastal communities in the southern Texas, United States have experienced hurricane impacts regularly. During each hurricane hazard, flood inundation and strong wind will always introduce forest damage within the coastal area. A large amount of fallen trees will be observed after the hurricane. Those fallen trees play an crucial role in various domains, including ecosystem management, hydrological analysis, and recovery policy-making. For ecosystem management, efficient and accuracy statistical method to characterize the amount of fallen trees is needed [116]. As the landscape and geomorphological features will be changed by fallen trees, the detailed information of those fallen trees can be regarded as additional inputs for hydrological analysis when measuring the further influence of such a disaster [117]. In the mean while, federal and state agencies need to response quickly with information regarding the severity of hurricane by accessing the amount and location of fallen trees in order to better facilitate recovery efforts in those areas [118]. Therefore, characterizing fallen tree, including fallen tree extraction and localization, is an important research focus for those and other potential sub-fields.

As the damage of fallen trees always occurs in rural areas, surveying in the field will be time-consuming and high-cost. Remote sensing is a better tool for the researchers to obtain the data in a timely manner right after the hurricane happens. Basically, remote sensing is an technology of obtaining and monitoring the properties of the Earth surface in a non-contact manner [119]. According to the source of signal the remote-sensing sensors use to explore the object, there are two main types of remote sensing, including the passive and active remote sensing. Passive remote sensing refers to the technique where the electromagnetic radiation (EMR) recorded by sensor is reflected or emitted from the surface of the Earth. On the other side, active remote sensing will receive and operate signals emitted from its own instruments. Both remote sensing systems have a

wide application for fallen tree detection.

For the application of fallen tree detection utilizing passive remote sensing, the most widely used datasets include satellite imagery and unmanned aerial vehicle (UAV) based imagery. Satellite images can provide large-scale raster with relative higher spectral resolution where the slight differences between fallen tree and background can be extracted. Regarding the UAV-based images, they always have very high spatial resolution where the detailed spatial information can be characterized for the extraction of fallen tree. [120] developed a framework to extract damage forest from high-resolution RS images and DEM. Multinomial logit model was adopted as classification model, and accuracies were assessed on pixel level. [121] proposed a method to extract fallen tree pixels by comparing two satellite images with before and after the typhoon. A couple of normalized spectral indices were calculated and exploited under machine learning-based model. [118] utilized high-resolution aerial photos to detect fallen tree trunks. Instead of employing machine learning model, the authors mainly focus on threshold-based selection where edge and line detection maps were computed. [122] investigated the application of deep learning algorithm, to extract forest damage areas from multispectral aerial remote sensing data. The U-Net semantic segmentation [123] was employed to segment images and extract forest damage areas. [124] also proposed U-Net-based framework, but the authors applied this scheme on very-high spatial resolution satellite imagery.

Apart from those aforementioned passive remote sensing based models, active remote sensing produces are still an critical datasets for fallen tree detection. The most popular data source of active remote sensing is light detection and ranging (LiDAR). LiDAR uses a laser to target an object and measures the time/distance for the reflected light to return to the sensor. LiDAR product is 3-dimensional (3D) point cloud where the coordinates of those points can be delineated. Such a property can be used to represent the surface structure of target object in a detailed manner. [125] generated multiple rasters based on LiDAR point cloud datasets and utilized multi-resolution object-based image analysis (OBIA) and classification algorithms to identify downed logs. [126] proposed to generate digital terrain model (DTM) from point clouds obtained from airborne laser

scanning (ALS) system. And then line matching strategy was employed to detect fallen trees. Its accuracy measurements were implemented based on line-to-line comparison.[127] developed a model to detect fallen trees directly from ALS point clouds. Segmentation was applied on point clouds, and the Normalized Cut algorithm was adopted in order to merge short segments into whole stems. Similar with the previous work, [128] exploited specialized constrained conditional random field (CRF) to better construct point cloud segmentation.

Regarding the aforementioned proposed approaches, it can be found that using satellite- or UAV-based images are more straightforward as fallen tree detection problem can be transformed into classification problem. Benefit from that, various machine learning-, even advanced deep learning-based models have been proposed and achieved satisfactory performance. However, their drawbacks are obvious. First, some investigations [118][129] focus on fall tree trunk instead of the whole fallen tree. It means that the properties of those fallen trees with crowns and/or branches will not be delineated correctly. Second, those models utilizing deep learning models are applied on damaged forest identification in a relatively large scale, and detailed information of individual fallen trees will be ignored accordingly. For the LiDAR-based models, the precise structures of fallen trees are characterized well due to the intrinsic properties of 3D point cloud. However, those methods need many user-defined parameters which have strong relationship with LiDAR data itself and terrain condition. Therefore, the issue of how to develop a more general fallen tree detection model, especially for individual fallen tree, still needs more investigation.

In this research, we propose a novel object-detection-based fallen tree detection framework using very-high-resolution aerial photos. Specifically, it consists of three main steps, including fallen tree digitization, automatic training and testing sample generation, and object detection model training and evaluation. As an crucial prerequisite of deep learning-based object detection model, localization of fallen trees is needed. For this research, large-scale high-spatial-resolution aerial photos are employed as base images for both digitization and object detect purposes. Fallen trees will be identified based on visual interpretation and digitized on ArcGIS Pro, which is a popular desktop GIS software developed by ESRI(R). And those digitization results are saved as an indi-

vidual shapefile, a geospatial vector data format for data interoperability. As such a digitization processing is built on geographical coordinates, where the real location information of those fallen trees including boundaries of fallen trees are recorded, and on large-scale aerial photos (i.e. 500 meters by 500 meters for each individual aerial photo), the digitized fallen trees and their corresponding aerial photos will not be regarded as inputs of object detection model. For those deep learning-based object detection models, a single input consists of two parts: 1) an image which contains one or more objects; and 2) an annotation file which encloses the detailed location information of bounding boxes of those objects in image-based coordinates. Consequently, the transformation from large-scale shapefile containing objects of interest and aerial photos to small-scale digital images with bounding boxes information indicating location information of those objects is needed. In this article, we designed an automatic workflow to handle such an issue with the purpose of generating inputs/datasets which can be utilized directly for object detection model. Specifically, given a fallen tree polygon, its minimum bounding box is extracted in the first step. Then based on the size of this bounding box, the size of sub-image (the image used as input of object detection model) is determined. And the sub-image will be extracted from its corresponding aerial photo by utilizing its geospatial location and size. Lastly, for those fallen tree polygons incorporated with the current sub-image, their image-based coordinates will be calculated and added into its annotation file. Such a processing will be iterated until all fallen tree polygons has been employed to create their sub-images and annotation files. Once that dataset is prepared, they will be split into training and testing datasets where training dataset will be utilized to train the object detection model and testing dataset will be exploited to assess the detection accuracy.

The main contributions of this research are summarized as follows:

- 1) We propose a deep learning-based object detection model to detect fallen tree in a large scale manner. To the best of our knowledge, this is the first time to apply for object detection algorithm on individual fallen tree detection problem, especially in such a large scale dataset. There are some research on the utilization of deep learning models to handle this problem but they mainly focus on forest damage measurement, instead of individual fallen tree identification.

2) An automatic scheme for generating/re-organizing training and testing datasets for the object-detection model is specifically designed. Currently, previous studies mainly rely on manual work where the sub-images are determined and extracted manually. However, this strategy introduces loss of pixel information due to the resampling issue, and it is very time-consuming and includes high labor costs. Few studies provide insights on or investigate the possibility of transforming large-scale vector geographic information system (GIS) data (e.g., polygon shapefiles) into deep learning-amenable datasets automatically.

3) The influence of different degrees of fallen tree overlapping is also investigated in this research. Within our study area, some areas have more individual fallen trees but some may have more overlapped fallen trees. The detection performance will be evaluated on those different areas on both individual tree level and tree cluster level.

The rest of this section is organized as follows: Section 5.2 describes some background, including deep learning-based object detection algorithm. Study area and dataset utilized in this study is illustrated in Section 5.3. Section 5.4 proposes the proposed framework for automatic training and testing dataset generation for object detection model. Section 5.5 shows the experimental results. Finally, in Section 5.6, conclusions and discussions are presented.

## **5.2 Background: CNN and Faster R-CNN**

### **5.2.1 CNN**

Convolutional neural networks (CNNs) [130] is a popular deep learning model which was developed from traditional feed-forward neural network. Similar with those simple neural networks, CNNs also have input layers, hidden layers, and output layers, and utilize non-linear activation function to make layer-wise connection. Inspired by biological processes [131], convolutional filters are introduced into CNNs which are utilized to simulate visual fields in a restricted region. Those visual fields overlap partially so that they can cover the whole visual field. Convolutional filters are implemented with convolutional layer, which convolves its input layer and pass its results to the next layer. Such a convolutional filters will be applied on all those pixels within input image,

and the corresponding output is equal to the dot product between convolutional filter and those pixels within the sliding window of current convolutional filter. Another difference between CNN and conventional neural network is pooling layer. Within pooling layer, a pooling filter will be applied on the input feature image as well. Role of pooling layer is to reduce dimension of input through translational and rotational invariants. There are two popular pooling filters, including maximum pooling where the maximum value within filter window is extracted and average pooling which takes the average values. Due to the concept of visual field/receptive field introduced by convolutional filter, CNNs achieve significant performance on 2D image/video based tasks, including image recognition [132][133], object detection [35][134][135], semantic segmentation [123][136], and video analysis [137][138]. CNNs themselves can be regarded as a series of neural network structure and can be easily plugged into other deep learning models as backbone networks for different tasks.

### **5.2.2 Faster R-CNN**

Faster region-based CNN (Faster R-CNN) [135] is utilized as object detection model for this research. It belongs to the R-CNN family, whose basic model was developed and described in [35]. R-CNN itself can be regarded as the first and successfully application using CNN for object detection. R-CNN consists of three steps: 1) region proposal extraction; 2) CNN feature computation; and 3) region classification. Even though the utilization of CNN achieve satisfied performance in terms of feature extraction, the computational time cost is critical due to "selective search" when proposing candidate regions. And additional classifier (e.g. SVM) is employed to further increase memory and time cost. As an improvement of R-CNN, Fast R-CNN [134] was developed by employing a single model instead of a pipeline to learn regions and classification. Within Fast R-CNN, bounding boxes and classification can be yielded simultaneously where computation can be accelerated to some extent. However, it still need a set of candidate regions to be proposed for each image. As a further development and improvement of Fast R-CNN, Faster R-CNN is a single unified model, which consists of two modules: 1) region proposal network (RPN); and 2) Fast R-CNN. Both modules are connected with the same output of a deep CNN, which means

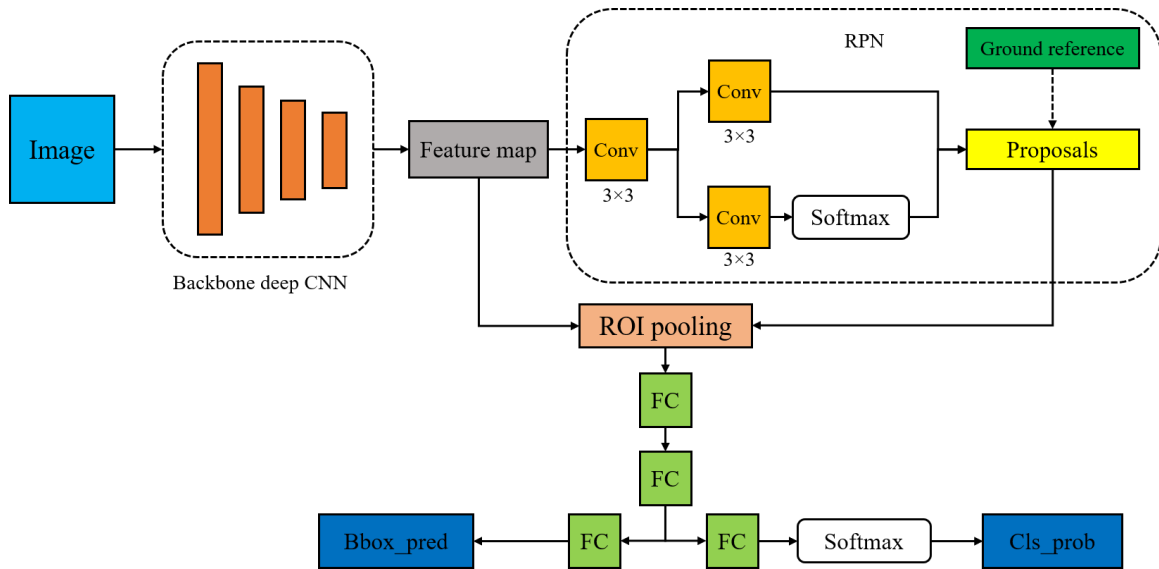


Figure 5.1: The architecture of Faster R-CNN. “Conv” is convolutional layer, “FC” represents fully connected layer. “Bbox\_pred” is position offset of predicted bounding box, and “Cls\_prob” denotes category probability of predicted bounding box

that all those aforementioned step, including feature extract, region proposal extraction, bounding box regression, and classification, are integrated into a single network. Such a strategy speeds up object detection procedure compared with the other two R-CNN-based models and achieve more accurate results. Figure 5.1 illustrates basic framework of Faster R-CNN. The main contribution of Faster R-CNN is RPN, which is utilized to extract region proposals. After obtaining feature map generated from deep CNN backbone network, RPN outputs a set of region proposals with objectness scores that are utilized to measure if an anchor belongs to foreground or background. Once those proposals created from RRN are obtained, they are combined with feature map within ROI pooling layer to get fixed-size proposal feature maps. And those feature maps are fed into multiple fully connected layers to obtain final object detection results, including determining bounding box category and its precise location.

### 5.3 Study area and Data

For this research, we choose the Fennessey Ranch on the Texas Coastal Bend in the U.S. The Fennessey Ranch lies on the Mission-Aransas National Estuarine Research Reserve (MANERR),

which is a large complex of wetland, terrestrial, and marine environments. The total area of our study site is 13752539.544 square meters. Fennessey Ranch is a suitable area for this research regarding hurricane-induced tree-blowdown impacts as it is on the path of Hurricane Harvey. Extreme winds induced substantial amount of tree blowdowns and large wood debris on the river and its floodplain. Obtaining the location of those fallen trees and analyzing their distribution will benefit further investigation on other research sub-fields, including hydrodynamic modeling, geomorphology study, and nature hazard evaluation.

The fallen trees are identified manually from very-high-resolution georeferenced aerial photos obtained from DIMAC RGB digital camera. In total 64 images are utilized in this study, and they were obtained after Hurricane Harvey at 2018. For each aerial photo, its spatial resolution is 5 centimeter and its spatial extend is 500 meter by 500 meter. In order to better extract corresponding aerial photos based on geographical coordinates, those aerial photos follow a name convention using the lower-left coordinate as the seed for the file name. Assume we have such an aerial photo named "2018\_Refugio\_667000\_3122000.tif", that means "667000" is the easting value of its lower left corner, and "3122000" is the northing value of the lower left corner. During the visual interpretation of fallen trees by utilizing those high-resolution aerial photos, the detailed boundary will be digitized by using ArcGIS Pro and saved in shapefile which will not only be utilized in this study but also benefit other related geospatial research. Figure 5.2 illustrates basic framework of Faster R-CNN.

#### **5.4 Methodology**

For the fallen tree detection problem, it is critical to identify the location and boundary of those trees from large-scale remote sensing product (e.g., high-resolution aerial photos) and create datasets accepted by deep learning-based object detection model accordingly. Basically, in order to create such kind of datasets in geospatial data science, especially for object detection problem, a popular workflow is to only focus on dataset creation. During this step, training and testing images can be extracted directly by either clipping the original images or taking screenshots. Then a labeling tool is employed to annotate objects by drawing their bounding boxes and export annota-



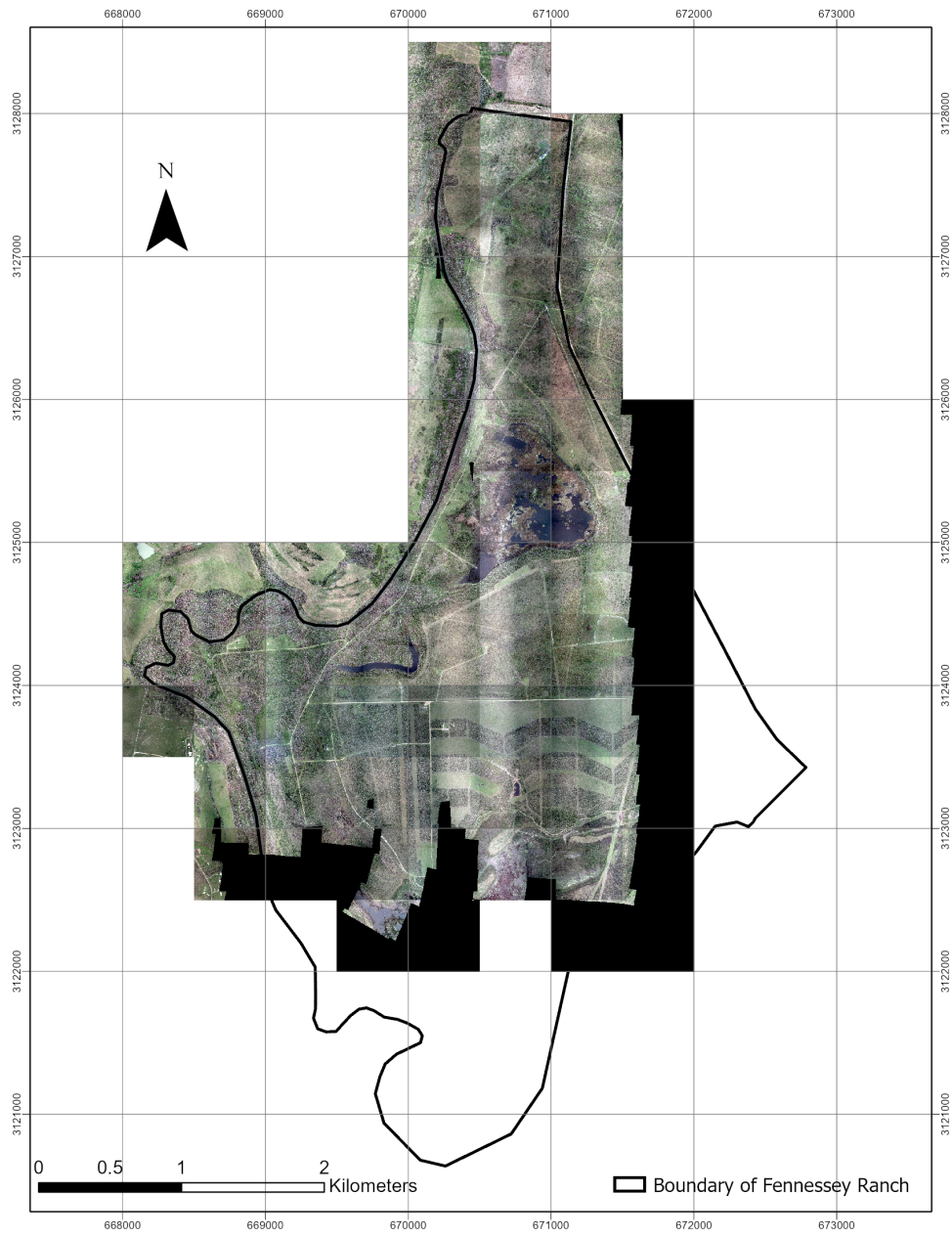


Figure 5.2: Study area for this research

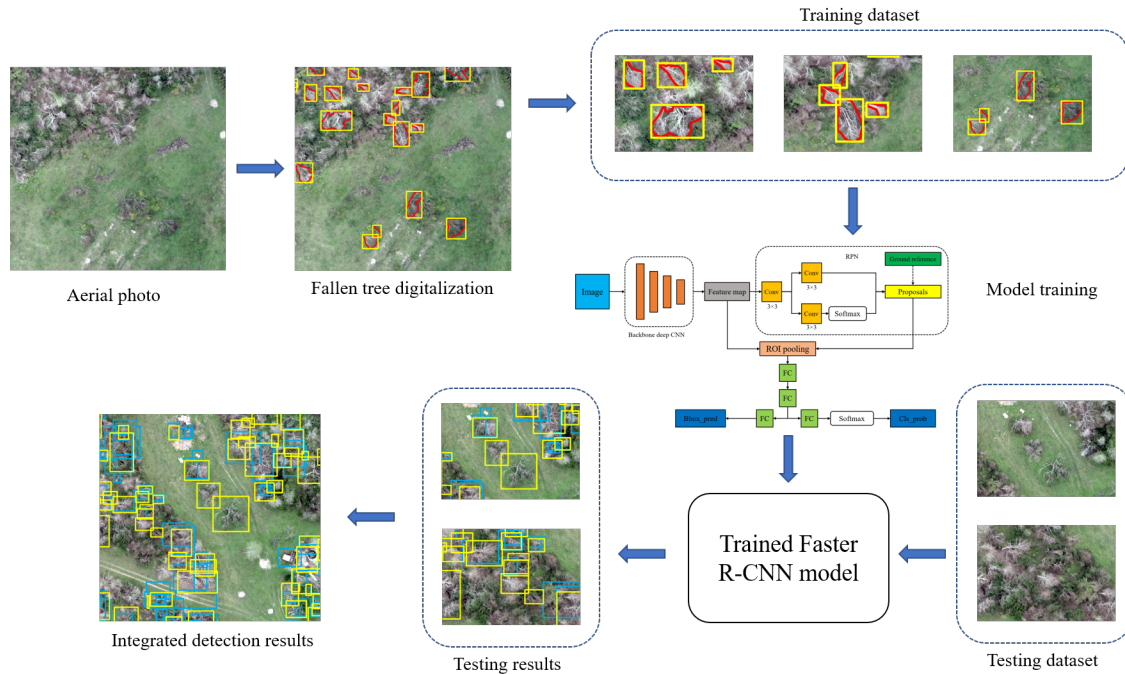


Figure 5.3: Architecture of proposed framework. Red polygons are digitized fallen trees. Yellow rectangles are the minimum bounding boxes of fallen trees. Blue rectangles are object detection results

tion files. From this strategy, scale issue can be well controlled as the size of training and testing images are always determined manually. However, if the annotation files are generated in this manner, the geospatial information of those bounding boxes of objects is always ignored which means that we know where are those objects on the images but we don't know where are they on the Earth. Thus, to address the aforementioned issue, we propose automatic scheme for dataset creation by utilizing digitized fallen trees with geographical coordinates and aerial photos. Within this framework, geospatial information of fallen trees will be kept which is able to be utilized in any potential further geospatial analysis. Figure 5.3 provides a basic workflow of our proposed scheme.

#### 5.4.1 Fallen tree digitization

In the research, digitization of fallen trees is implemented in ArcGIS Pro, a widely used geospatial software, where both vector data (i.e. shapefile) and raster data can be displayed simultane-

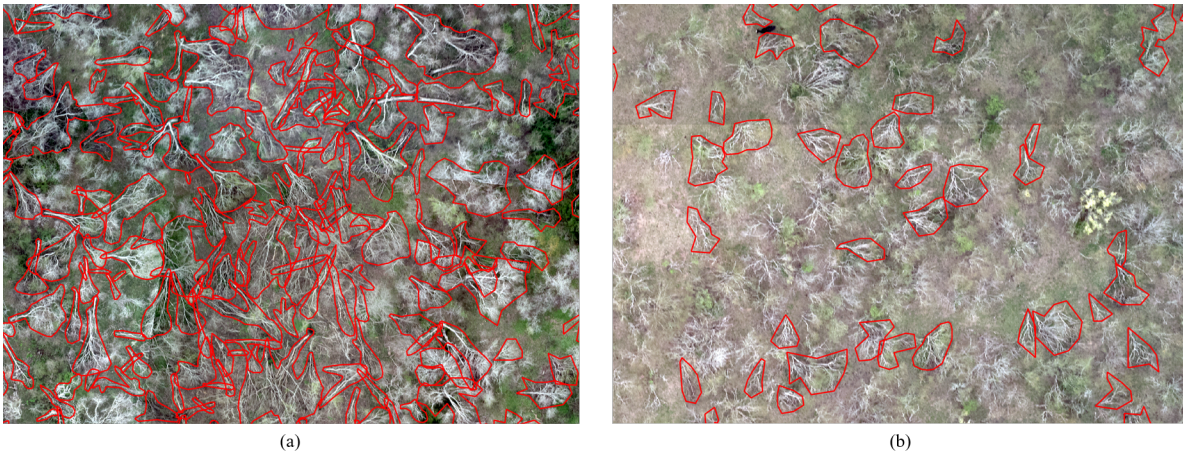


Figure 5.4: Examples of fallen tree digitization. Red polygons are digitized fallen trees

ously. Given an aerial photo, those fallen trees, especially for their boundaries, appeared in this image will be identified and digitized based on visual interpretation. And they will be marked on individual fallen tree level. It means that even if two fallen trees are overlapped with each other, those two will still be digitized as two individual fallen trees. Moreover, when digitizing fallen trees, not only tree trunks, but canopies will be delineated as well in order to keep the integrity of fallen trees. Figure 5.4 shows two examples of digitized fallen trees. Once all fallen trees have been digitized, they will be saved into a single shapefile, a common geospatial vector data, and their corresponding bounding boxes of each fallen tree will be computed separately as an input dataset for the following steps.

#### **5.4.2 Automatic image and annotation generation for deep learning-based object detection model**

Once the shapefile containing all bounding boxes of fallen trees is generated, the next step is to combine it with aerial photos to generate image and annotation datasets for object detection model. In general, the entire scheme for image and annotation generation consists of four parts: 1) locating fallen tree and aerial photo based on geographical coordinates; 2) determining image size based on current processing fallen tree bounding box; 3) clipping and resampling aerial photo; and 4) creating annotation files.

The first issue of generating image datasets with annotation information automatically is to find the correct aerial photo that can cover the target fallen tree polygon. Benefited from the naming convention of those aerial photos, correct image will be determined by constructing the relationship between the geographical coordinates of fallen tree and those of lower left corner of aerial photo. Suppose we have a digitized fallen tree set  $\mathcal{T} = \{t^{(1)}, t^{(2)}, \dots, t^{(n-1)}, t^{(n)}\}$  where  $t^{(i)}$  is polygon for  $i$ th fallen tree and  $n$  denotes the number of fallen trees, its minimum bounding box set  $\mathcal{B} = \{b^{(1)}, b^{(2)}, \dots, b^{(n-1)}, b^{(n)}\}$  where  $b^{(i)}$  represents the minimum bounding box of  $t^{(i)}$ , and aerial photo dataset  $\mathcal{P} = \{p^{(1)}, p^{(2)}, \dots, p^{(63)}, p^{(64)}\}$  as we have 64 aerial photos in total. Since the fallen tree bounding box and aerial photo can be represented by the geographical coordinates of their four vertices, respectively,  $\mathcal{B}$  and  $\mathcal{P}$  can be rewrite as sets of coordinates where  $\mathcal{B} = \{\mathbf{bbox}^{(1)}, \mathbf{bbox}^{(2)}, \dots, \mathbf{bbox}^{(n-1)}, \mathbf{bbox}^{(n)}\} \in \mathbb{R}^{n \times 4}$  and  $\mathcal{P} = \{\mathbf{pbox}^{(1)}, \mathbf{pbox}^{(2)}, \dots, \mathbf{pbox}^{(n-1)}, \mathbf{pbox}^{(n)}\} \in \mathbb{R}^{64 \times 4}$ , respectively. For the  $\mathbf{bbox}^{(i)}$  and  $\mathbf{pbox}^{(i)}$ , their representations are defined as follows:

$$bbox^{(i)} = [b\_geog_{east}^{min(i)}, b\_geog_{east}^{max(i)}, b\_geog_{north}^{min(i)}, b\_geog_{north}^{max(i)}] \quad (5.1)$$

$$pbox^{(i)} = [p\_geog_{east}^{min(i)}, p\_geog_{east}^{max(i)}, p\_geog_{north}^{min(i)}, p\_geog_{north}^{max(i)}] \quad (5.2)$$

Figure 5.5 shows the detailed information of those geographical coordinates for fallen tree bounding boxes and aerial photo. Given the  $\mathbf{bbox}^{(i)}$ , its corresponding aerial photo located at the same area for further image extraction will be determined by exploiting the following equations:

$$\begin{aligned} easting\_name^{(i)} &= b\_geog_{east}^{min(i)} // 1000 * 1000 \\ &+ (b\_geog_{east}^{min(i)} // 100 - b\_geog_{east}^{min(i)} // 1000 * 10) // 5 * 5 * 100 \end{aligned} \quad (5.3)$$

$$\begin{aligned}
n\text{orthing\_name}^{(i)} &= b\_geog_{north}^{min(i)} // 1000 * 1000 \\
&+ (b\_geog_{north}^{min(i)} // 100 - b\_geog_{north}^{min(i)} // 1000 * 10) // 5 * 5 * 100
\end{aligned}
\tag{5.4}$$

where  $easting\_name^{(i)}$  and  $northing\_name^{(i)}$  are two geographical coordinates which are taken as components of file name of target aerial photo, and “//” represents floor division where the result is the quotient in which the digits after the decimal point are removed.

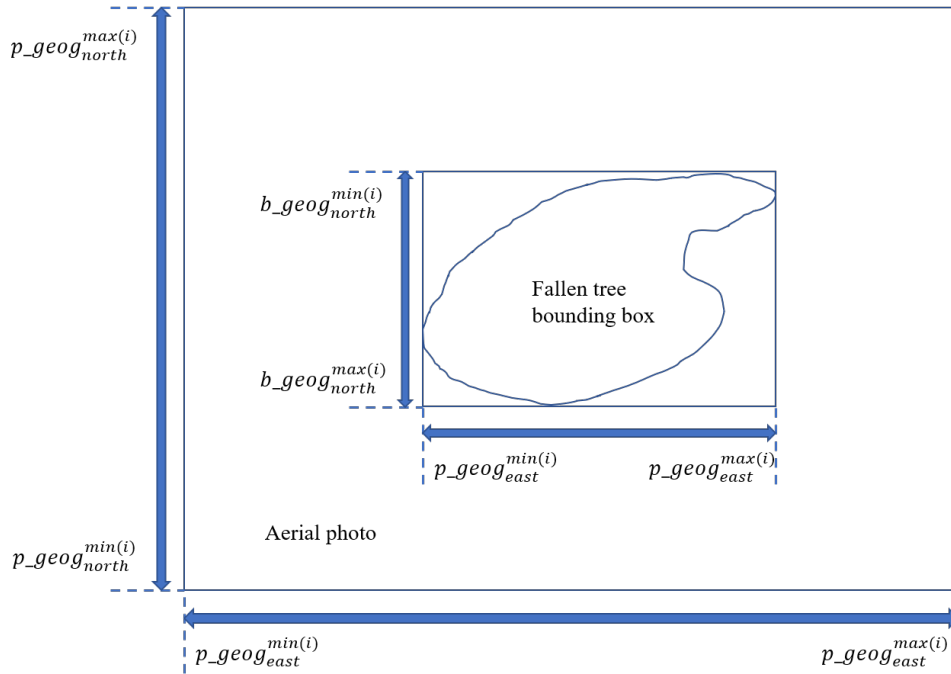


Figure 5.5: Example of geographical coordinates for fallen tree bounding box and aerial photo

After obtaining the correct aerial photo, next step is to choose an appropriate rectangle around the target fallen tree to clip aerial photo. Since different fallen trees have different spatial dimensions, the size of this rectangle cannot be specified in the same size and applied to all fallen trees as very “small” or “large” fallen tree measured by image size will be extracted which will bring more difficulties and challenges for object detection task. Therefore, we propose a dynamic strategy to

determine the spatial extend for clipping aerial photos. The clipped image is named as “sub-image” and will be used in the following text, and its coordinates of 4 vertices need to be calculated. Given the  $i$ th fallen tree bounding boxes  $bbox^{(i)}$ , we define a scale parameter  $c$  which will be applied to modify the length and width of  $bbox^{(i)}$  in the following way:

$$s\_geog_x^i = (b\_geog_{east}^{max(i)} - b\_geog_{east}^{min(i)})/c \quad (5.5)$$

$$s\_geog_y^i = (b\_geog_{north}^{max(i)} - b\_geog_{north}^{min(i)})/c \quad (5.6)$$

where  $s\_geog_x^i$  and  $s\_geog_y^i$  represent width and length of new created sub-image clipped from its aerial photo and extracted based on the geographical coordinates of  $i$ th fallen tree bounding boxes, respectively. This sub-image which is extracted based on  $bbox^{(i)}$  is named as  $s^{(i)}$ .

Here another issue that needs to be consider is the aspect ratio. Since different fallen trees may share different aspect ratio and the sizes of bounding boxes influence the size of extracted sub-image, we choose to define a fixed aspect ratio  $r$  that works for all sub-images in order to make sure that all sub-images share the same aspect ratio. Thus, before moving forward to the next step, aspect ratio for sub-image  $s^{(1)}$  will be checked and updated as:

$$s\_geog_{x\_updated}^i = \begin{cases} s\_geog_{x\_updated}^i, & \text{if } s\_geog_x^i/s\_geog_y^i \geq r \\ s\_geog_y^i \times r, & \text{otherwise} \end{cases} \quad (5.7)$$

$$s\_geog_{y\_updated}^i = \begin{cases} \frac{s\_geog_x^i}{r}, & \text{if } s\_geog_x^i/s\_geog_y^i \geq r \\ s\_geog_{y\_updated}^i, & \text{otherwise} \end{cases} \quad (5.8)$$

Once the updated width  $s\_geog_{x\_updated}^i$  and length  $rect\_img_i^{y\_updated}$  for sub-image  $s^{(i)}$  are obtained, then the spatial extent of  $s^{(i)}$  will be used to clip aerial photo. Note that  $s^{(i)}$  shares the same centroid with its fallen tree bounding box  $bbox^{(i)}$ . Therefore, the geographical coordinates of four vertices of this rectangle can be calculated as

$$s\_geog_{east}^{min(i)} = (b\_geog_{east}^{max(i)} - b\_geog_{east}^{min(i)})/2 - s\_geog_{x\_updated}^i/2 \quad (5.9)$$

$$s\_geog_{east}^{max(i)} = (b\_geog_{east}^{max(i)} - b\_geog_{east}^{min(i)})/2 + s\_geog_{x\_updated}^i/2 \quad (5.10)$$

$$s\_geog_{north}^{min(i)} = (b\_geog_{north}^{max(i)} - b\_geog_{north}^{min(i)})/2 - s\_geog_{y\_updated}^i/2 \quad (5.11)$$

$$s\_geog_{north}^{max(i)} = (b\_geog_{north}^{max(i)} - b\_geog_{north}^{min(i)})/2 + s\_geog_{y\_updated}^i/2 \quad (5.12)$$

Currently, the representation of coordinates of sub-image  $s^{(i)}$  is in geographical coordinates. In order to extract clipping image from aerial photo in a more efficient way, using image coordinates (indices in rows and columns) is well accepted due to the intrinsic vector structure of aerial photo. In other words, given the geographical coordinates of a bounding boxes of fallen tree and its corresponding aerial photo, we will need to know the position of this bounding box on the aerial photo describing in rows and columns. Consequently, regarding the coordinates of vertices of sub-image  $s^{(i)}$ , they need to be converted as follows:

$$s\_img_{row}^{min(i)} = (p\_geog_{north}^{max(i)} - s\_geog_{north}^{max(i)})/pixel\_size\_northing \quad (5.13)$$

$$s\_img_{row}^{max(i)} = (p\_geog_{north}^{max(i)} - s\_geog_{north}^{min(i)})/pixel\_size\_northing \quad (5.14)$$

$$s\_img_{col}^{min(i)} = (s\_geog_{east}^{min(i)} - p\_geog_{east}^{min(i)})/pixel\_size\_easting \quad (5.15)$$

$$s\_img_{col}^{max(i)} = (s\_geog_{east}^{max(i)} - p\_geog_{east}^{min(i)})/pixel\_size\_easting \quad (5.16)$$

where  $pixel\_size\_northing$  and  $pixel\_size\_easting$  represent spatial resolution for an individual

pixel along northing (vertical) and easting (horizontal) directions, respectively. By utilizing results calculated from Equations 5.13 - 5.15 as row and column indices, sub-image  $s^{(i)}$  can be extracted from the original aerial photo as follows:

$$s^{(i)} = p^{(i)}[s\_img_{row}^{min(i)} : s\_img_{row}^{max(i)}, s\_img_{col}^{min(i)} : s\_img_{col}^{max(i)}] \quad (5.17)$$

After obtaining sub-image  $ss^{(i)}$  from aerial photo, next step will be on extraction of image coordinates for those fallen tree bounding boxes located within this sub-image. Here we calculate intersect areas (denotes as  $iter\_areas^{(i)}$ ) between the spatial extent of  $ss^{(i)}$  and all fallen tree bounding boxes, and only extract those bounding boxes with intersect areas that are equal to their own areas. Those selected fallen tree bounding boxes are represented as  $\mathbf{included\_bbox}^{(i)} = \{\mathbf{included\_bbox}_1^{(i)}, \mathbf{included\_bbox}_2^{(i)}, \dots, \mathbf{included\_bbox}_m^{(i)}\}$  where  $m$  is the number of included bounding boxes. Similar with Equations 5.13 - 5.15, transformation from geographical coordinates to image coordinates is essential for all those included fallen tree bounding boxes. Given a bounding box  $\mathbf{bbox}^{(j)}$  which is included in sub-image  $s^{(i)}$ , its image coordinates on  $s^{(i)}$  can be calculated as follows:

$$b\_img_{row}^{min(j)} = (s\_geog_{north}^{max(i)} - b\_geog_{north}^{max(j)})/pixel\_size\_northing \quad (5.18)$$

$$b\_img_{row}^{max(j)} = (s\_geog_{north}^{min(i)} - b\_geog_{north}^{min(j)})/pixel\_size\_northing \quad (5.19)$$

$$b\_img_{col}^{min(j)} = (b\_geog_{east}^{min(j)} - s\_geog_{east}^{min(i)})/pixel\_size\_easting \quad (5.20)$$

$$b\_img_{col}^{max(j)} = (b\_geog_{east}^{max(j)} - s\_geog_{east}^{min(i)})/pixel\_size\_easting \quad (5.21)$$

As the original spatial dimension of  $s^{(i)}$  may not fit our predefined dimension (numbers of rows and columns), resampling needs to be implemented on the extracted sub-image. Suppose we have



a user-defined two parameters, including the number of rows ( $nR$ ) and the number of columns ( $nC$ ), the  $subimage_i$  will be resampled as follows:

$$s_{resample}^{(i)} = f_r(s^{(i)}, nR, nC) \quad (5.22)$$

where  $s_{resample}^{(i)}$  is the resampled image of  $s^{(i)}$ , and  $f_r(\cdot)$  denotes resampling function. In the research, we utilize nearest neighbour as resampling algorithm in order to avoid reconstructing pixel information. Accordingly, the image coordinates of those bounding boxes will be modified as well. Therefore, for Equations 5.18 - 5.20, they can be re-written as follows:

$$b_{img_{row}^{min(j)}} = (s_{geog_{north}^{max(i)}} - b_{geog_{north}^{max(j)}}) / pixel\_size\_northing * par_{row}^{(i)} \quad (5.23)$$

$$b_{img_{row}^{max(j)}} = (s_{geog_{north}^{max(i)}} - b_{geog_{north}^{min(j)}}) / pixel\_size\_northing * par_{row}^{(i)} \quad (5.24)$$

$$b_{img_{col}^{min(j)}} = (b_{geog_{east}^{min(j)}} - s_{geog_{east}^{min(i)}}) / pixel\_size\_easting * par_{col}^{(i)} \quad (5.25)$$

$$b_{img_{col}^{max(j)}} = (b_{geog_{east}^{max(j)}} - s_{geog_{east}^{min(i)}}) / pixel\_size\_easting * par_{col}^{(i)} \quad (5.26)$$

where  $par_{row}^{(i)}$  and  $par_{col}^{(i)}$  represent resampling parameters for the vertical and horizontal dimensions for  $i$ th fallen tree bounding box, respectively. Their calculations can be seen as follows

$$par_{row}^{(i)} = nR / (s_{img_{row}^{max(i)}} - s_{img_{row}^{min(i)}}) \quad (5.27)$$

$$par_{col}^{(i)} = nC / (s\_img_{col}^{max(i)} - s\_img_{col}^{min(i)}) \quad (5.28)$$

For those extracted image coordinates of all fallen tree bounding boxes included within a given clipping rectangle, they will be reorganized and saved into an annotation file associated with the corresponding extracted sub-image. There are two main data formats which have been widely used for annotating objects, including COCO data format [139] and Pascal VOC data format [140]. In this research, Pascal VOC format is adopted for annotation file generation. The detailed algorithm description can be found in Algorithm 3.

## 5.5 Experiments

### 5.5.1 Experimental setup

For the digitized fallen tree dataset, it consists of 41796 fallen tree polygons and their corresponding bounding box rectangles. Their resulting sub-image and annotation dataset contains 27295 extracted JPG image and XML annotation files. This dataset will be split into training and testing datasets where training samples are utilized to train object detection model, and testing samples are employed to evaluate the performance of object detection model. Regarding the implementation of Faster R-CNN model, its backbone deep neural network is ResNet-50 [141] which was proposed in 2016. It introduces residual learning framework to overcome difficulties when training deeper neural networks. The size of its input image is  $600 \times 800 \times 3$ . The optimizer is Adam [142] with the initial learning rate is  $1 \times 10^{-5}$ . The number of epoch is 60 and the length for each epoch is 400. All codes are implemented based on Keras [91], a popular deep learning package, within Tensorflow [92] as its backend.

The quantitative evaluation for evaluating object detection performance utilized in this study are intersection over union (IOU), true positive (TP), false positive (FP), false negative (FN), precision, and recall. The first index IOU is utilized to measure overlap between two bounding boxes. The IOU of two bounding boxes  $bbox_1$  and  $bbox_2$  can be calculated as follows:

---

**Algorithm 3** Automatic image and annotation generation for object detection model

---

**Input:** Dataset of digitized fallen tree  $\mathcal{T}$ , dataset of aerial photo  $\mathcal{P}$ , scale parameter  $c$  aspect ratio  $r$ , target number of rows ( $nR$ ), and target number of columns ( $nC$ )

**Initialization:** Sub-image dataset  $\mathcal{IMG} \leftarrow []$ , annotation dataset  $\mathcal{AN}\mathcal{O} \leftarrow []$ , list  $b\_processed.all() \leftarrow False$

**Begin:**

```
1: for  $t^{(i)}$  in  $\mathcal{T}$  do
2:   if  $b\_processed[t^{(i)}$  is False then
3:     Calculate its minimum bounding box  $\mathbf{bbox}^{(i)}$ 
4:     Determine the correct aerial photo  $p_i$  by (5.3) to (5.4)
5:     Calculate original size of sub-image  $s^{(i)}$  by (5.5) to (5.6)
6:     Reshape  $s^{(i)}$  by (5.7) to (5.8)
7:     Calculate updated geographical coordinates of  $s^{(i)}$  by (5.9) to (5.12)
8:     Calculate image coordinates of  $s^{(i)}$  by (5.13) to (5.16)
9:     Extract sub-image  $s^{(i)}$  by (5.17)
10:    Calculate intersect areas  $iter\_areas^{(i)}$  and extract  $\mathbf{included\_bbox}^{(i)}$  using  $s^{(i)}$ 
11:    Calculate  $s_{resample}^{(i)}$  by (5.22)
12:    for  $\mathbf{included\_bbox}_1^{(i)}$  in  $\mathbf{included\_bbox}^{(i)}$  do
13:      Calculate image coordinates  $b\_img_{row}^{min(j)}$ ,  $b\_img_{row}^{max(j)}$ ,  $b\_img_{col}^{min(j)}$ , and  $b\_img_{col}^{max(j)}$ 
        based on  $s^{(i)}$  by (5.18) to (5.21)
14:       $b\_processed[\mathbf{included\_bbox}_1^{(i)}] \leftarrow True$ 
15:      Update image coordinates  $b\_img_{row}^{min(j)}$ ,  $b\_img_{row}^{max(j)}$ ,  $b\_img_{col}^{min(j)}$ , and  $b\_img_{col}^{max(j)}$ 
        based on  $s^{(i)}$  by (5.23) to (5.26) using  $s_{resample}^{(i)}$ 
16:       $\mathcal{AN}\mathcal{O}[i].append([b\_img_{row}^{min(j)}$ ,  $b\_img_{row}^{max(j)}$ ,  $b\_img_{col}^{min(j)}$ , and  $b\_img_{col}^{max(j)}])$ 
17:    end for
18:     $\mathcal{IMG}.append(s_{resample}^{(i)})$ 
19:  end if
20: end for
```

**Output:** Sub-image dataset  $\mathcal{IMG}$ , and annotation dataset  $\mathcal{AN}\mathcal{O}$

---

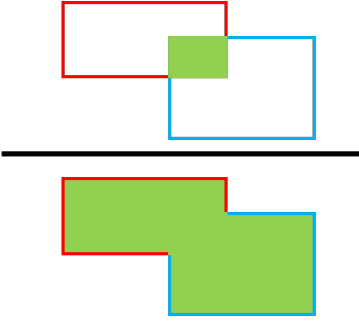
$$IOU = \frac{\text{area of intersection}}{\text{area of union}} = \frac{\text{green area}}{\text{red and blue area}} = \frac{\text{green area}}{\text{red and blue area}}$$


Figure 5.6: Calculation of IOU between two bounding boxes. Red rectangle is ground reference bounding box, and blue rectangle is detected bounding box

$$IOU = \frac{bbox_1 \cap bbox_2}{bbox_1 \cup bbox_2} \quad (5.29)$$

where  $bbox_1 \cap bbox_2$  represents the calculation of intersection area between  $bbox_1$  and  $bbox_2$ , and  $bbox_1 \cup bbox_2$  denotes union area between  $bbox_1$  and  $bbox_2$ . Figure 5.6 illustrates the IOU between a ground reference bounding box and a detected bounding box.

IOU will be utilized to measure if a ground reference bounding box is extracted or not by a given threshold  $IOU$ . In this research,  $IOU$  is set to 0.5. It means that if IOU is greater or equal to  $IOU$ , the ground reference bounding box is regarded as detected. Based on the aforementioned definition, additional three metrics, including TP, FP, and FN. TP is correct detection with IOU is greater or equal to  $IOU$ . FN is a wrong detection if IOU is smaller than  $IOU$ . If a ground reference bounding box is given in the image and model fails to detect this object, it will be classified as FN. Consequently, the calculations of precision and recall can be calculated as follows:

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{\text{number of all detections}} \quad (5.30)$$

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{\text{number of all ground truths}} \quad (5.31)$$

## 5.5.2 Experimental results on testing dataset

The quantitative results of Faster R-CNN on our generated dataset are shown in Table 5.1. Note that "No. of GT" represents the number of ground reference bounding boxes, and "No. of PRE" denotes the number of detected bounding boxes. And those terms will be reused in the following tables. All those results are calculated on sub-image level, which means that those metrics computation is implemented sub-image by sub-image. From it we can see that the recall and precision are 51.71% and 42.46%, respectively. In the meanwhile, the number of detected bounding boxes are larger than that of the ground truth bounding boxes, which illustrates that this object detection model predicts more bounding boxes compared with the ground reference data. Qualitative results obtained from Figure 5.7 also exhibits similar phenomenon. It is obvious that some fallen trees can be detected very well. However, some fallen trees have more than one detected bounding boxes, where the precision is reduced.

Recall	Precision	TP	FP	FN	No. of GT	No. of PRE
51.72%	42.46%	13733	18609	12818	26551	32342

Table 5.1: Object detection result on sub-image level using the entire testing dataset

From Figure 5.7 we also find that there are some fallen trees (e.g., Figure 5.7 (e) and (f)) located near the boundaries of sub-images but without labeling information. It is determined and generated based on our proposed sub-image extraction strategy. In the our proposed method, those fallen trees located on the boundary of sub-image belong to “intersect” fallen trees, instead of “included” fallen trees and they will be ignored for annotation for current sub-image. However, their remaining parts within sub-image may have notable features which allow object detection model to detect them as “fallen trees”. Therefore, in order to characterize and evaluate those detected “fallen trees”, we calculate the accuracy indices on the global scale. Specifically, instead of computing those indices sub-image by sub-image, we extract the geographical coordinates of

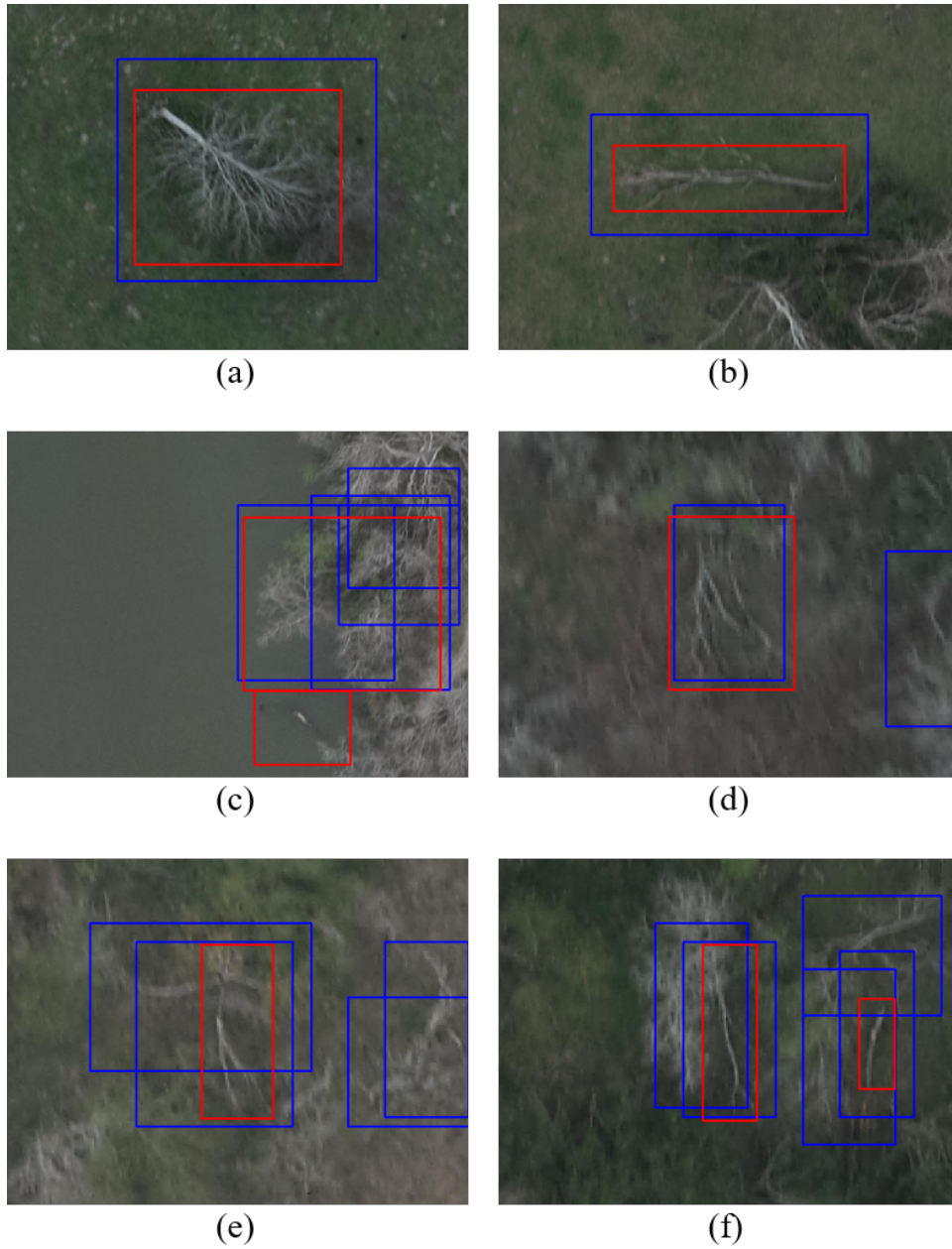


Figure 5.7: Some object detection results. Red rectangle is ground reference bounding box, and blue rectangle is detected bounding box

those detected bounding boxes and all ground reference bounding boxes from all testing images. Then the IOUs and other accuracy metrics are calculated based on geographical coordinates. Table 5.2 shows the results using geographical coordinates. The apparent improvements in object detection performance can be observed. The recall is increased from 51.72% to 61.07%, and the precision is increased from 42.46% to 50.15%. In the meanwhile, TP is increased, and FP and FN are decreased as well. Such an improvement illustrates the importance of considering the detection results coming from boundaries of sub-images.

Recall	Precision	TP	FP	FN	No. of GT	No. of PRE
61.07%	50.14%	16216	16126	10335	26551	32342

Table 5.2: Object detection result on geographical coordinate level using the entire testing dataset

### 5.5.3 Analysis on different areas with different overlapping level

From the digitization results of fallen trees, we can find that there are a lot overlapping fallen trees in our study area. That is common phenomenon for dense woods and forest. As the object detection performance shown in Table 5.1 and 5.2 is not competitive and good enough compared with other object detection tasks, we create two different datasets based on different overlapping level: 1) dataset A with more isolated fallen trees; and 2) dataset B with more overlapping fallen trees. Those two datasets are all generated from the original dataset in Section X. Within dataset A, there are 2165 fallen trees included. And for dataset B, 3383 fallen trees are selected with relatively higher density and overlapping level. Afterwards, the same Faster R-CNN models are implemented on those two datasets starting from initial training to accuracy assessments. Moreover, the object detection performance are still evaluated on both image coordinate level (evaluation image by image) and geographical coordinate level. Table 5.3 illustrated the accuracies on those two datasets. From this table we can find that the best recall and precision are obtained from the dataset with more isolated fallen trees when evaluating accuracies on geographical coordinate. Compared with

the best results obtained from Table 5.2, its precision is decreased slightly from 50.14% to 47.76% but its recall increases markedly from 61.07% to 88.40%. Such an improvement illustrates that this object detection model performs better on individual fallen tree detection. In other words, it is more easier to detect isolated fallen trees for object detection model. For the other dataset with more overlapping fallen trees, it is obvious that the model performs worse with lower recall and precision compared with those results within Table 5.2. Figure 5.8 shows object detection results collected from the aforementioned two datasets, where Figure 5.8 (a) to (f) are detection results from dataset A and 5.8 (g) to (l) are obtained from dataset B.

Data	Evaluation	Recall	Precision	TP	FP	FN	No. of GT	No. of PRE
A	IMG	79.63%	43.02%	1724	2283	441	2165	4007
A	GEO	<b>88.40%</b>	<b>47.76%</b>	1914	2093	251	2165	4007
B	IMG	35.74%	29.92%	1209	2832	2174	3383	4041
B	GEO	46.44%	38.88%	1571	2470	1812	3383	4041

Table 5.3: Object detection results on two different datasets. “A” represents the dataset with more isolated fallen trees. “B” is the dataset with more overlapping fallen trees. “IMG” denotes the accuracy evaluation on sub-image coordinates. “GEO” is the accuracy evaluation on geographical coordinates

In order to analysis the influence of those overlapping fallen trees and to overcome such a difficult, we propose another evaluation framework to alleviate misdetection. Specifically, we choose to merge individual overlapped bounding boxes to create multiple polygon clusters, and evaluate the accuracies between those clusters. Figure 5.9 illustrates an example of creating a cluster based on multiple overlapping bounding boxes. This experiments are implemented only on dataset B, which is a subset of the original dataset with more overlapping fallen trees. And the geographical coordinates are utilized for IOU calculation. The merging criterion is defined based on the IOU between two overlapped polygons. If the IOU is greater than a predefined threshold, then those two bounding boxes will be regarded as “overlapped” bounding boxes and will be merged into a single one. As resulting clustering polygons are irregular polygon which may



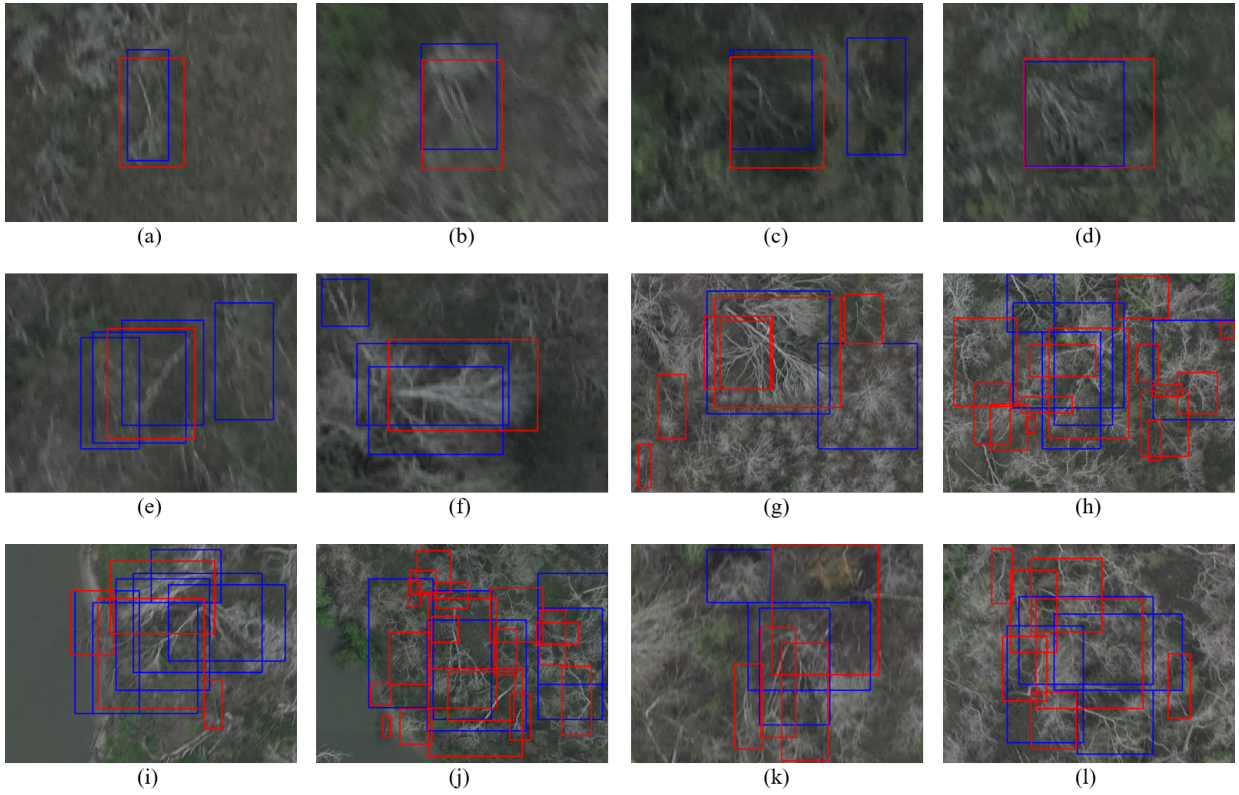


Figure 5.8: Some object detection results collected from two datasets with different overlapping degrees



Figure 5.9: Example of creating clusters based on overlapped fallen trees. Red rectangles represent the original polygons, and yellow rectangles denote merged polygons. Merged bounding boxes are highlighted in blue rectangles

belong to concave polygons or have holds, calculating their intersect and union are pretty complex and difficult. To simplify this problem, we still create minimum bounding boxes for those cluster polygons and utilize those bounding boxes for accuracy evaluation. Algorithm 4 provides detailed explanations for this merging procedure.

In order to further investigate the influence of clusters on accuracy assessment, two experiments are implemented. The first is to only create those clusters and corresponding bounding boxes for overlapping fallen trees on the detected results. And we exploit the individual bounding boxes of ground reference dataset as base dataset for accuracy assessment. In other words, for the first experiment, the accuracy evaluation is calculated between individual fallen tree level and fallen tree cluster level. Additionally, for the IOU parameter ( $iou\_pred$ ), 10 different values (0, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%) are utilized in current experiments. Table 5.4 illustrates all results obtained from those different parameters. The best recall (46.36%) is obtained when ( $iou\_pred$ ) is 90%, which is slightly decreased compared with the base one (46.44%) collected from bounding box to bounding box comparison. Regarding the precision, we can see that for multiple trials when ( $iou\_pred$ ) is set to from 20% to 90%, precision values are always greater

---

**Algorithm 4** Generation of bounding box clusters by merging overlapping bounding boxes

---

**Input:** Dataset of fallen tree bounding boxes  $\mathcal{B}$ , IOU parameter  $iou$

**Initialization:** Cluster dataset  $\mathcal{CLU} \leftarrow []$ , list  $list\_cluster\_index.all() \leftarrow 0$ , and index  $process\_idx \leftarrow 0$

**Begin:**

```
1: for  $bbox^{(i)}$  in  $\mathcal{B}$  do
2:   if  $list\_cluster\_index[bbox^{(i)}]$  is not 0 then
3:     Calculate  $list\_IOU^{(i)}$  between  $bbox^{(i)}$  and  $\mathcal{B}$ 
4:     Select overlapped fallen tree set  $list\_op\_tree^{(i)} \leftarrow \mathcal{B}[where(list\_IOU^{(i)} > iou)]$ 
5:      $list\_cluster\_index.all(list\_op\_tree^{(i)}) \leftarrow process\_idx$ 
6:     while  $list\_op\_tree^{(i)}.size$  is not 0 do
7:       if  $list\_op\_tree_i.size[0]$  is not  $bbox^{(i)}$  then
8:          $list\_op\_tree^{(j)} = list\_op\_tree^{(i)}[0]$ 
9:         Calculate  $list\_IOU^{(j)}$  between  $list\_op\_tree^{(j)}$  and  $\mathcal{B}$ 
10:        Select overlapped fallen tree set  $list\_op\_tree^{(j)} \leftarrow \mathcal{B}[where(list\_IOU^{(j)} > iou)]$ 
11:         $list\_cluster\_index.all(list\_op\_tree^{(j)}) \leftarrow process\_idx$ 
12:       end if
13:        $list\_op\_tree^{(i)}.size.pop(0)$ 
14:        $list\_op\_tree^{(i)}.size.append(list\_op\_tree_j)$ 
15:     end while
16:      $process\_idx \leftarrow process\_idx + 1$ 
17:   end if
18: end for
19:  $\mathcal{CLU} = Dissolve(\mathcal{B}, list\_cluster\_index)$ 
```

**Output:** Cluster dataset  $\mathcal{CLU}$

---

than the base one that is 38.88%. However, it may not be available to achieve a better results in terms of both recall and precision.

The other experiment is to create clusters for both ground reference bounding boxes and detected bounding boxes. Therefore, calculation of those accuracy metrics is implemented on “cluster-to-cluster” level. For this experiments, two different IOU parameters ( $iou\_gt$  and  $iou\_pred$ ) are utilized on both ground reference data and detected results, respectively. And both of them are all 10 different values from 0 to 90% with a step of 10%. Thus, we have 100 different combinations using those two IOU parameters in total. All those results are displayed in Figure 5.11 and 5.12.

Regarding the recall results, we noticed that given a specified  $iou\_gt$ , a higher recall value can be obtained when the  $iou\_pred$  is large enough. And the trend is smaller  $iou\_gt$  and larger

IOU	Recall	Precision	TP	FP	FN	No. of GT	No. of PRE
0	8.63%	32.48%	292	607	3090	3382	899
0.1	11.65%	35.82%	394	706	2988	3382	1100
0.2	26.76%	45.00%	905	1106	2477	3382	2011
0.3	32.82%	<b>47.29%</b>	1110	1237	2272	3382	2347
0.4	35.87%	46.96%	1213	1370	2169	3382	2583
0.5	41.69%	44.88%	1410	1732	1972	3382	3142
0.6	45.74%	40.48%	1547	2275	1835	3382	3822
0.7	45.98%	39.57%	1555	2375	1827	3382	3930
0.8	46.30%	39.24%	1566	2425	1816	3382	3991
0.9	<b>46.36%</b>	38.90%	1568	2463	1814	3382	4031

Table 5.4: Accuracy assessment on the dataset with more overlapping fallen trees using clusters created from detected results

$iou\_pred$  will produce higher recall values. Moreover, the best recall is obtain when  $iou\_gt$  is 0 and  $iou\_pred$  is 60%, where all overlapped bounding boxes are merged into clusters. However, such a trend is opposite when analyzing precision results. The best precision is achieved when  $iou\_gt$  is 60% and  $iou\_pred$  is 30%. And precision tends to rise with decreasing  $iou\_pred$ , which means that if more predicted bounding boxes are merged into clusters, higher precision will be yielded. Furthermore, among those 100 results, we find that there are 48 records with higher recall and 24 records with high precision compared with the base one shown in Table 5.3. And none of them has higher recall and precision simultaneously. Those results illustrates that detecting overlapping fallen trees is still a very challenging problem, even considering clustering those overlapping fallen trees bounding boxes.

## 5.6 Conclusion

In this study, for the purpose of detecting fallen tress from large scale aerial photos, we develop a framework to automatically create image and annotation dataset which can be utilized as input data for deep learning based object detection model. Based on such a dataset generated from digitized fallen trees and very high resolution aerial photos, we investigate the fallen tree detection problem using Faster R-CNN model. Quantitative and qualitative classification results illustrate that, for isolated fallen trees, our model achieves promising results considering recall index. How-

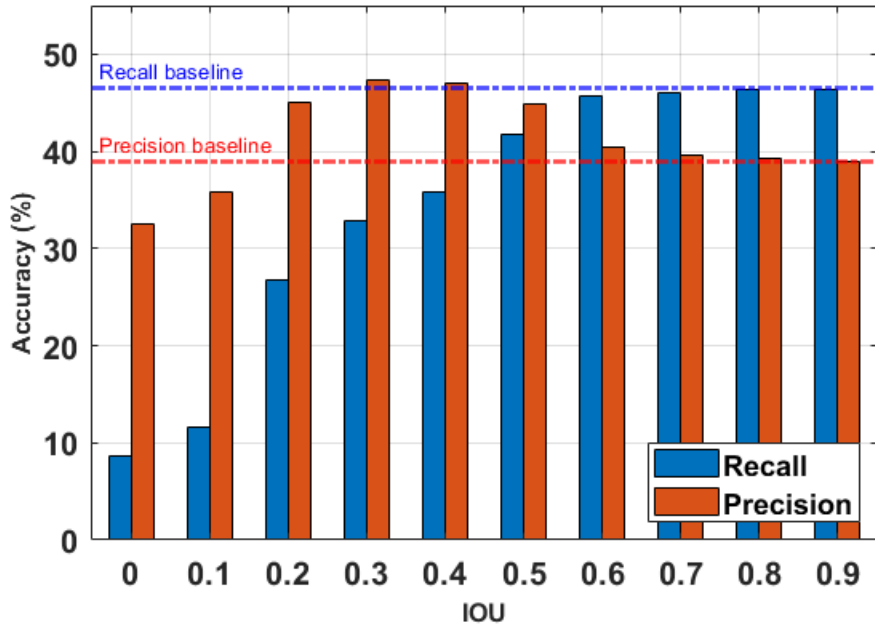


Figure 5.10: Recall and precision using different IOU parameters

ever, many false positive detection results are generated as well. As the proposed automatic data generation framework ignores those fallen trees lying on the boundaries of extracted sub-image, the accuracy calculation method is modified as well from image level to geographical level using the geographical coordinates of all predicted bounding boxes. Experimental results show that utilizing geographical coordinates based accuracy measurement will have higher accuracies. We also investigate the influence of overlapping fallen trees. For the study area with more overlapping fallen trees, clustering based analysis is introduced where overlapped fallen trees will be merged into a single one for further accuracy assessment. From experimental results, we notice that recall and precision can be improved at some specific circumstances when using different IOU parameter for different merging level. However, they will not be improved simultaneously. Therefore, detecting overlapping fallen trees is still a challenging task, especially for those large scale datasets collected from the real world.

In the future, more experiments using different object detection algorithms will be implemented. Furthermore, combining aerial photos and hyperspectral remote sensing images needs to

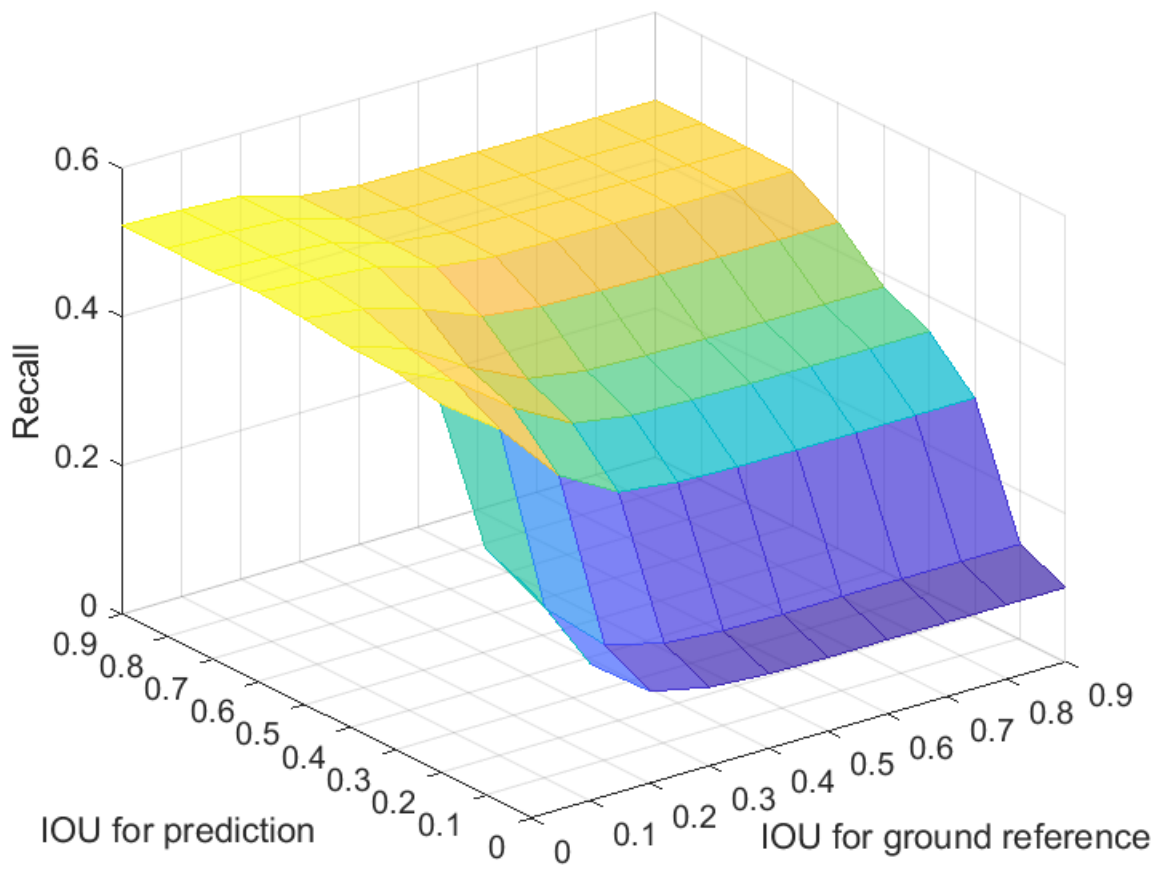


Figure 5.11: Recall results using different IOU parameters

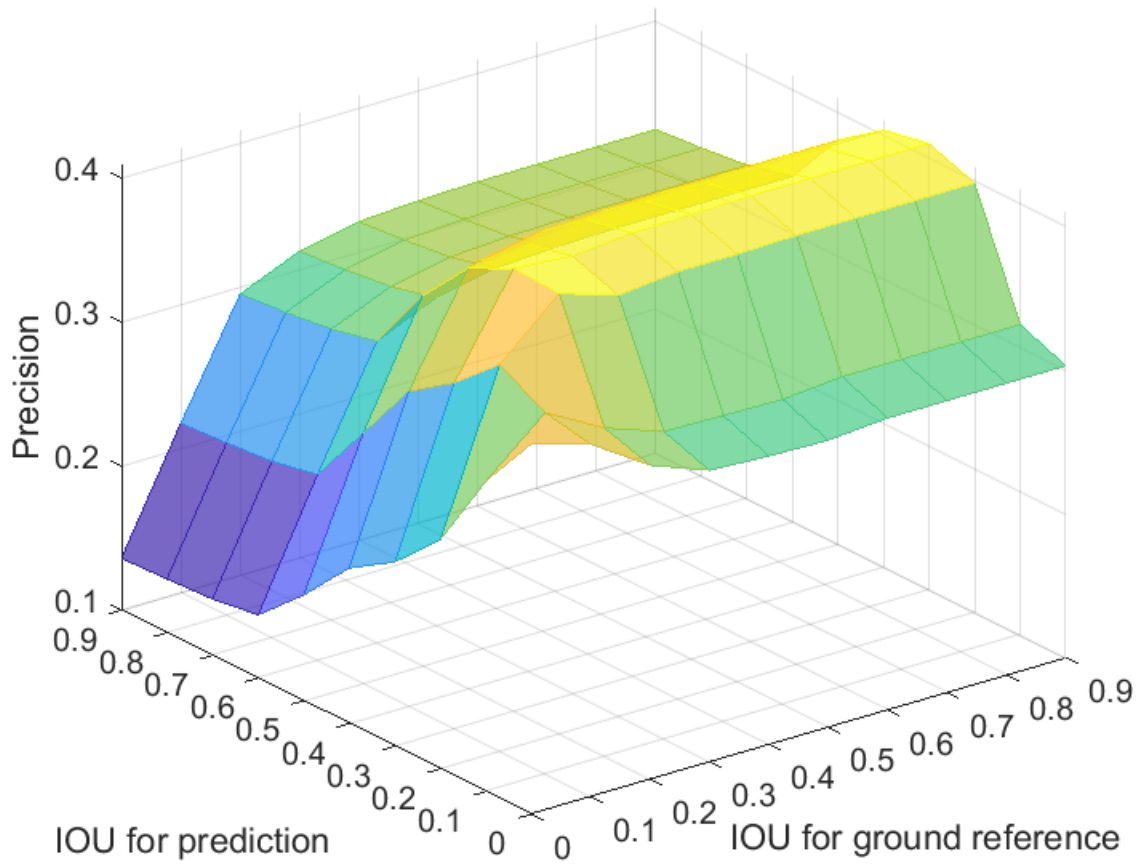


Figure 5.12: Precision results using different IOU parameters

be conducted to investigate the potential detection improvements by adding additional data source.



## 6. CONCLUSION

In this study, a novel sequential feature extraction framework for single image classification problem is proposed, where unlabeled data are well-exploited in order to construct sequential features from a single image. Instead of using spectral features as the sequential data structure of RNN, similar pixels collected from the entire image are used to construct the respective sequential features. Such schemes take full advantage of unlabeled data in the HSI. Moreover, block-matching-based schemes also consider spatial contextual information in the classification process, and it is demonstrated in this research that such schemes are effective in increasing image classification performance. From the experimental results, we also found that choosing a smaller sequence length and smaller window size for block-matching is an appropriate strategy for our proposed methods to achieve higher classification.

Then we improved such a model by introducing object-based segmentation to accelerate sequential feature construction. Similar pixels which are employed for sequential features are not selected from the whole-image scope, but are rather chosen from individual segments within the segmentation map. Three different sequential feature extraction strategies are developed, where local and non-local segments are considered in a separate and combined manner, respectively. Classification results illustrate that, for the Pavia University and Salinas images, the method using mixed strategy and block matching achieves the best classification performance. For the Indian Pine image, the best classification performance is obtained by the method using local segment and block matching, which indicates that for the Indian Pine image with lower spatial resolution, the local segment makes a greater contribution to the similar-pixel calculation and selection. Similar pixels associated with a given target pixel will be more likely to be located in the local segment. By investigating the computational time cost of our proposed methods, we find that those approaches are still very competitive, accuracy-wise, and they are capable of achieving acceptable classification performance with markedly lower computation cost.

We also apply this proposed framework for LSTM-based land-cover classification. This is a

first extension of our proposed approach from the utilization on benchmark dataset to the large-scale satellite image classification problem. For this research, the study site is located in the rural area in the northern India, and the objective is to classify/identify land-cover information within plantations. Experimental results illustrate that our proposed approach achieves better performances compared with the results obtained from 2DCNN. The parameter sensitivity analysis is adopted as well to evaluate the influence of using different segmentation maps and similar segments. We find that smaller scale parameter and less similar segments will produce higher accuracy.

Apart from classification, we also investigated the application of object detection for remote sensing data analysis. The main objective is to detect fallen trees from coastal areas in southern Texas, United States. We developed a framework to automatically create image and annotation dataset which can be utilized as input data for deep learning based object detection model. Based on such a dataset, we investigated the fallen tree detection problem using Faster R-CNN model. Quantitative and qualitative classification results illustrate that, for isolated fallen trees, our model achieves promising results considering recall index. However, many false positive detection results are generated as well. As the proposed automatic data generation framework ignores those fallen trees lying on the boundaries of extracted sub-image, the accuracy calculation method is modified as well from image level to geographical level using the geographical coordinates of all predicted bounding boxes. Experimental results show that utilizing geographical coordinates based accuracy measurement will have higher accuracies. We also explore the influence of overlapping fallen trees. For the study area with more overlapping fallen trees, clustering based analysis is introduced where overlapped fallen trees will be merged into a single one for further accuracy assessment. From experimental results, we notice that recall and precision can be improved at some specific circumstances when using different IOU parameter for different merging level. However, they will not be improved simultaneously.

Our future work includes applying novel deep learning models in other tasks utilizing datasets collected from real scenarios, such as natural hazard assessment and climate changes analysis.

## REFERENCES

- [1] Y. Hu, S. Gao, S. D. Newsam, and D. D. Lunga, “Geoai 2018 workshop report the 2nd acm sigspatial international workshop on geoi: Ai for geographic knowledge discovery seattle, wa, usa-november 6, 2018.,” *ACM SIGSPATIAL Special*, vol. 10, no. 3, p. 16, 2018.
- [2] D. Haboudane, J. R. Miller, E. Pattey, P. J. Zarco-Tejada, and I. B. Strachan, “Hyperspectral vegetation indices and novel algorithms for predicting green lai of crop canopies: Modeling and validation in the context of precision agriculture,” *Remote sensing of environment*, vol. 90, no. 3, pp. 337–352, 2004.
- [3] G. Camps-Valls, D. Tuia, L. Bruzzone, and J. A. Benediktsson, “Advances in hyperspectral image classification: Earth monitoring with statistical learning methods,” *IEEE Signal Processing Magazine*, vol. 31, no. 1, pp. 45–54, 2014.
- [4] J. M. Bioucas-Dias, A. Plaza, G. Camps-Valls, P. Scheunders, N. Nasrabadi, and J. Chanussot, “Hyperspectral remote sensing data analysis and future challenges,” *IEEE Geoscience and remote sensing magazine*, vol. 1, no. 2, pp. 6–36, 2013.
- [5] M. T. Eismann, J. Meola, and R. C. Hardie, “Hyperspectral change detection in the presence of diurnal and seasonal variations,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 1, pp. 237–249, 2008.
- [6] Q. Wang, Z. Yuan, Q. Du, and X. Li, “Getnet: A general end-to-end 2-d cnn framework for hyperspectral image change detection,” *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1–11, 2018.
- [7] A. Plaza, Q. Du, Y.-L. Chang, and R. L. King, “High performance computing for hyperspectral remote sensing,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 4, no. 3, pp. 528–544, 2011.

- [8] L. Ma, M. M. Crawford, and J. Tian, "Local manifold learning-based  $k$ -nearest-neighbor for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 11, pp. 4099–4109, 2010.
- [9] X. Jia and J. A. Richards, "Fast k-nn classification using the cluster-space approach," *IEEE Geoscience and Remote Sensing Letters*, vol. 2, no. 2, pp. 225–228, 2005.
- [10] Q. Wang, F. Zhang, and X. Li, "Optimal clustering framework for hyperspectral band selection," *IEEE Transactions on Geoscience and Remote Sensing*, no. 99, pp. 1–13, 2018.
- [11] F. Melgani and L. Bruzzone, "Classification of hyperspectral remote sensing images with support vector machines," *IEEE Transactions on geoscience and remote sensing*, vol. 42, no. 8, pp. 1778–1790, 2004.
- [12] J. Gualtieri and S. Chettri, "Support vector machines for classification of hyperspectral data," in *Geoscience and Remote Sensing Symposium, 2000. Proceedings. IGARSS 2000. IEEE 2000 International*, vol. 2, pp. 813–815, IEEE, 2000.
- [13] Y. Tarabalka, M. Fauvel, J. Chanussot, and J. A. Benediktsson, "Svm and mrf-based method for accurate classification of hyperspectral images," *IEEE Geoscience and Remote Sensing Letters*, vol. 7, no. 4, pp. 736–740, 2010.
- [14] Y. Chen, N. M. Nasrabadi, and T. D. Tran, "Hyperspectral image classification using dictionary-based sparse representation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 10, pp. 3973–3985, 2011.
- [15] Y. Chen, N. M. Nasrabadi, and T. D. Tran, "Hyperspectral image classification via kernel sparse representation," *IEEE Transactions on Geoscience and Remote sensing*, vol. 51, no. 1, pp. 217–231, 2013.
- [16] H. Zhang, J. Li, Y. Huang, and L. Zhang, "A nonlocal weighted joint sparse representation classification method for hyperspectral imagery," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2056–2065, 2014.

- [17] Q. Wang, X. He, and X. Li, “Locality and structure regularized low rank representation for hyperspectral image classification,” *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1–13, 2018.
- [18] J. A. Benediktsson, P. H. Swain, and O. K. Ersoy, “Conjugate-gradient neural networks in classification of multisource and very-high-dimensional remote sensing data,” *International Journal of Remote Sensing*, vol. 14, no. 15, pp. 2883–2903, 1993.
- [19] H. Yang, “A back-propagation neural network for mineralogical mapping from aviris data,” *International Journal of Remote Sensing*, vol. 20, no. 1, pp. 97–110, 1999.
- [20] L. Gómez-Chova, G. Camps-Valls, J. Muñoz-Mari, and J. Calpe, “Semisupervised image classification with laplacian support vector machines,” *IEEE Geoscience and Remote Sensing Letters*, vol. 5, no. 3, pp. 336–340, 2008.
- [21] L. Bruzzone, M. Chi, and M. Marconcini, “A novel transductive svm for semisupervised classification of remote-sensing images,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 11, pp. 3363–3373, 2006.
- [22] L. Ma, M. M. Crawford, X. Yang, and Y. Guo, “Local-manifold-learning-based graph construction for semisupervised hyperspectral image classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 5, pp. 2832–2844, 2015.
- [23] L. Ma, A. Ma, C. Ju, and X. Li, “Graph-based semi-supervised learning for spectral-spatial hyperspectral image classification,” *Pattern Recognition Letters*, vol. 83, pp. 133–142, 2016.
- [24] Z. Wang, N. M. Nasrabadi, and T. S. Huang, “Semisupervised hyperspectral classification using task-driven dictionary learning with laplacian regularization,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 3, pp. 1161–1173, 2015.
- [25] I. Kotsia, W. Guo, and I. Patras, “Higher rank support tensor machines for visual recognition,” *Pattern Recognition*, vol. 45, no. 12, pp. 4192–4203, 2012.

- [26] H. Zhou, L. Li, and H. Zhu, “Tensor regression with applications in neuroimaging data analysis,” *Journal of the American Statistical Association*, vol. 108, no. 502, pp. 540–552, 2013.
- [27] K. Makantasis, A. D. Doulamis, N. D. Doulamis, and A. Nikitakis, “Tensor-based classification models for hyperspectral data analysis,” *IEEE Transactions on Geoscience and Remote Sensing*, no. 99, pp. 1–15, 2018.
- [28] J. A. Benediktsson, J. A. Palmason, and J. R. Sveinsson, “Classification of hyperspectral data from urban areas based on extended morphological profiles,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 3, pp. 480–491, 2005.
- [29] L. Zhang, L. Zhang, D. Tao, and X. Huang, “On combining multiple features for hyperspectral remote sensing image classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 3, pp. 879–893, 2012.
- [30] M. Shi and G. Healey, “Using multiband correlation models for the invariant recognition of 3-d hyperspectral textures,” *IEEE transactions on geoscience and remote sensing*, vol. 43, no. 5, pp. 1201–1209, 2005.
- [31] L. Zhang, L. Zhang, and B. Du, “Deep learning for remote sensing data: A technical tutorial on the state of the art,” *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, no. 2, pp. 22–40, 2016.
- [32] Q. Wang, S. Liu, J. Chanussot, and X. Li, “Scene classification with recurrent attention of vhr remote sensing images,” *IEEE Transactions on Geoscience and Remote Sensing*, no. 99, pp. 1–13, 2018.
- [33] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, “Deep learning-based classification of hyperspectral data,” *IEEE Journal of Selected topics in applied earth observations and remote sensing*, vol. 7, no. 6, pp. 2094–2107, 2014.

- [34] Y. Chen, X. Zhao, and X. Jia, "Spectral–spatial classification of hyperspectral data based on deep belief network," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 6, pp. 2381–2392, 2015.
- [35] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.
- [36] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *Journal of Sensors*, vol. 2015, 2015.
- [37] K. Makantasis, K. Karantzalos, A. Doulamis, and N. Doulamis, "Deep supervised learning for hyperspectral data classification through convolutional neural networks," in *Geoscience and Remote Sensing Symposium (IGARSS), 2015 IEEE International*, pp. 4959–4962, IEEE, 2015.
- [38] Y. Li, H. Zhang, and Q. Shen, "Spectral–spatial classification of hyperspectral imagery with 3d convolutional neural network," *Remote Sensing*, vol. 9, no. 1, p. 67, 2017.
- [39] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.
- [40] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.
- [41] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 IEEE international conference on*, pp. 6645–6649, IEEE, 2013.
- [42] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

- [43] D. Ienco, R. Gaetano, C. Dupaquier, and P. Maurel, "Land cover classification via multitemporal spatial data by deep recurrent neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 10, pp. 1685–1689, 2017.
- [44] A. Sharma, X. Liu, and X. Yang, "Land cover classification from multi-temporal, multi-spectral remotely sensed imagery using patch-based recurrent neural networks," *Neural Networks*, 2018.
- [45] L. Mou, P. Ghamisi, and X. X. Zhu, "Deep recurrent neural networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3639–3655, 2017.
- [46] H. Wu and S. Prasad, "Convolutional recurrent neural networks for hyperspectral data classification," *Remote Sensing*, vol. 9, no. 3, p. 298, 2017.
- [47] C. Shi and C.-M. Pun, "Multi-scale hierarchical recurrent neural networks for hyperspectral image classification," *Neurocomputing*, vol. 294, pp. 82–93, 2018.
- [48] B. Romera-Paredes and P. H. S. Torr, "Recurrent instance segmentation," in *European Conference on Computer Vision*, pp. 312–329, Springer, 2016.
- [49] Y. Zhong, J. Zhao, and L. Zhang, "A hybrid object-oriented conditional random field classification framework for high spatial resolution remote sensing imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 11, pp. 7023–7037, 2014.
- [50] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of cnn and rnn for natural language processing," *arXiv preprint arXiv:1702.01923*, 2017.
- [51] F. Fan and Y. Deng, "Enhancing endmember selection in multiple endmember spectral mixture analysis (mesma) for urban impervious surface area mapping using spectral angle and spectral distance parameters," *International Journal of Applied Earth Observation and Geoinformation*, vol. 33, pp. 290–301, 2014.
- [52] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.



- [53] M. Belkin and P. Niyogi, “Laplacian eigenmaps for dimensionality reduction and data representation,” *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.
- [54] G. P. Asner and K. B. Heidebrecht, “Spectral unmixing of vegetation, soil and dry carbon cover in arid regions: comparing multispectral and hyperspectral observations,” *International Journal of Remote Sensing*, vol. 23, no. 19, pp. 3939–3958, 2002.
- [55] W. R. Tobler, “A computer movie simulating urban growth in the detroit region,” *Economic geography*, vol. 46, no. sup1, pp. 234–240, 1970.
- [56] H. Pu, Z. Chen, B. Wang, and G.-M. Jiang, “A novel spatial–spectral similarity measure for dimensionality reduction and classification of hyperspectral imagery,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 11, pp. 7008–7022, 2014.
- [57] D. P. Huttenlocher, G. A. Klanderma, and W. J. Rucklidge, “Comparing images using the hausdorff distance,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 15, no. 9, pp. 850–863, 1993.
- [58] R. G. Congalton, “A review of assessing the accuracy of classifications of remotely sensed data,” *Remote sensing of environment*, vol. 37, no. 1, pp. 35–46, 1991.
- [59] B. Datt, T. R. McVicar, T. G. Van Niel, D. L. Jupp, and J. S. Pearlman, “Preprocessing EO-1 hyperion hyperspectral data to support the application of agricultural indexes,” vol. 41, no. 6, pp. 1246–1259, 2003.
- [60] S. Chakravorty, J. Li, and A. Plaza, “A technique for subpixel analysis of dynamic mangrove ecosystems with time-series hyperspectral image data,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 4, pp. 1244–1252, 2017.
- [61] L. Ma, M. M. Crawford, L. Zhu, and Y. Liu, “Centroid and covariance alignment-based domain adaptation for unsupervised classification of remote sensing images,” vol. 57, no. 4, pp. 2305–2323, 2018.
- [62] T. Schmid, M. Rodríguez-Rastrero, P. Escribano, A. Palacios-Orueta, E. Ben-Dor, A. Plaza, R. Milewski, M. Huesca, A. Bracken, V. Cicuéndez, *et al.*, “Characterization of soil erosion

- indicators using hyperspectral data from a mediterranean rainfed cultivated region,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 2, pp. 845–860, 2015.
- [63] J. Ham, Y. Chen, M. M. Crawford, and J. Ghosh, “Investigation of the random forest framework for classification of hyperspectral data,” vol. 43, no. 3, pp. 492–501, 2005.
- [64] Y. Chen, N. M. Nasrabadi, and T. D. Tran, “Hyperspectral image classification via kernel sparse representation,” vol. 51, no. 1, pp. 217–231, 2012.
- [65] Z. Wang, N. M. Nasrabadi, and T. S. Huang, “Semisupervised hyperspectral classification using task-driven dictionary learning with laplacian regularization,” vol. 53, no. 3, pp. 1161–1173, 2014.
- [66] L. Ma, M. M. Crawford, X. Yang, and Y. Guo, “Local-manifold-learning-based graph construction for semisupervised hyperspectral image classification,” vol. 53, no. 5, pp. 2832–2844, 2014.
- [67] Z. Sun, C. Wang, D. Li, and J. Li, “Semisupervised classification for hyperspectral imagery with transductive multiple-kernel learning,” vol. 11, no. 11, pp. 1991–1995, 2014.
- [68] D. Tuia, F. Ratle, F. Pacifici, M. F. Kanevski, and W. J. Emery, “Active learning methods for remote sensing image classification,” vol. 47, no. 7, pp. 2218–2232, 2009.
- [69] J. Li, J. M. Bioucas-Dias, and A. Plaza, “Semisupervised hyperspectral image segmentation using multinomial logistic regression with active learning,” vol. 48, no. 11, pp. 4085–4098, 2010.
- [70] J. Li, J. M. Bioucas-Dias, and A. Plaza, “Spectral–spatial classification of hyperspectral data using loopy belief propagation and active learning,” vol. 51, no. 2, pp. 844–856, 2012.
- [71] G. Camps-Valls, L. Gomez-Chova, J. Muñoz-Marí, J. Vila-Francés, and J. Calpe-Maravilla, “Composite kernels for hyperspectral image classification,” vol. 3, no. 1, pp. 93–97, 2006.

- [72] M. Fauvel, J. Chanussot, and J. A. Benediktsson, “A spatial–spectral kernel-based approach for the classification of remote-sensing images,” *Pattern Recognit.*, vol. 45, no. 1, pp. 381–392, 2012.
- [73] D. Tuia, F. Ratle, A. Pozdnoukhov, and G. Camps-Valls, “Multisource composite kernels for urban-image classification,” vol. 7, no. 1, pp. 88–92, 2009.
- [74] M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. R. Sveinsson, “Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles,” vol. 46, no. 11, pp. 3804–3814, 2008.
- [75] L. Zhang, L. Zhang, D. Tao, and X. Huang, “On combining multiple features for hyperspectral remote sensing image classification,” vol. 50, no. 3, pp. 879–893, 2011.
- [76] X. Huang and L. Zhang, “An SVM ensemble approach combining spectral, structural, and semantic features for the classification of high-resolution remotely sensed imagery,” vol. 51, no. 1, pp. 257–272, 2012.
- [77] Y. Tarabalka, J. A. Benediktsson, J. Chanussot, and J. C. Tilton, “Multiple spectral–spatial classification approach for hyperspectral data,” vol. 48, no. 11, pp. 4122–4132, 2010.
- [78] Z. Zhang, E. Pasolli, M. M. Crawford, and J. C. Tilton, “An active learning framework for hyperspectral image classification using hierarchical segmentation,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 2, pp. 640–654, 2015.
- [79] L. Zhu, Y. Chen, P. Ghamisi, and J. A. Benediktsson, “Generative adversarial networks for hyperspectral image classification,” vol. 56, no. 9, pp. 5046–5063, 2018.
- [80] D. H. T. Minh, D. Ienco, R. Gaetano, N. Lalande, E. Ndikumana, F. Osman, and P. Maurel, “Deep recurrent neural networks for winter vegetation quality mapping via multitemporal SAR sentinel-1,” vol. 15, no. 3, pp. 464–468, 2018.
- [81] R. Hang, Q. Liu, D. Hong, and P. Ghamisi, “Cascaded recurrent neural networks for hyperspectral image classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 8, pp. 5384–5394, 2019.

- [82] X. Zhang, Y. Sun, K. Jiang, C. Li, L. Jiao, and H. Zhou, "Spatial sequential recurrent neural network for hyperspectral image classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 11, pp. 4141–4155, 2018.
- [83] A. Ma, A. M. Filippi, Z. Wang, and Z. Yin, "Hyperspectral image classification using similarity measurements-based deep recurrent neural networks," *Remote Sens.*, vol. 11, no. 2, p. 194, 2019.
- [84] M. Pesaresi and J. A. Benediktsson, "A new approach for the morphological segmentation of high-resolution satellite imagery," vol. 39, no. 2, pp. 309–320, 2001.
- [85] A. K. Shackelford and C. H. Davis, "A combined fuzzy pixel-based and object-based approach for classification of high-resolution multispectral data over urban areas," vol. 41, no. 10, pp. 2354–2363, 2003.
- [86] Y. Zhong, R. Gao, and L. Zhang, "Multiscale and multifeature normalized cut segmentation for high spatial resolution remote sensing imagery," vol. 54, no. 10, pp. 6061–6075, 2016.
- [87] A. Ma and A. M. Filippi, "Hyperspectral image classification via object-oriented segmentation-based sequential feature extraction and recurrent neural network," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, accepted, 2020.
- [88] T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, and M. Ranzato, "Learning longer memory in recurrent neural networks," *arXiv preprint arXiv:1412.7753*, 2014.
- [89] M. Baatz and A. Schäpe, "Multiresolution segmentation - an optimization approach for high quality multi-scale image segmentation," in *Angewandte Geographische Informationsverarbeitung XII* (J. Strobl, T. Blaschke, and G. Griesebner, eds.), pp. 12–23, Wichmann-Verlag, Heidelberg, 2000.
- [90] Trimble, Inc., Munich, Germany, *Trimble eCognition Developer User Guide*, 2019. Release 9.5.
- [91] F. Chollet *et al.*, "Keras." <https://keras.io>, 2015.

- [92] M. Abadi *et al.*, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015. Software available from tensorflow.org.
- [93] M. Dalponte, L. Bruzzone, and D. Gianelle, “Fusion of hyperspectral and lidar remote sensing data for classification of complex forest areas,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 5, pp. 1416–1427, 2008.
- [94] C. C. Dymond, D. J. Mladenoff, and V. C. Radeloff, “Phenological differences in tasseled cap indices improve deciduous forest classification,” *Remote sensing of environment*, vol. 80, no. 3, pp. 460–472, 2002.
- [95] R. E. McRoberts, M. D. Nelson, and D. G. Wendt, “Stratified estimation of forest area using satellite imagery, inventory data, and the k-nearest neighbors technique,” *Remote Sensing of Environment*, vol. 82, no. 2-3, pp. 457–468, 2002.
- [96] K. J. Salovaara, S. Thessler, R. N. Malik, and H. Tuomisto, “Classification of amazonian primary rain forest vegetation using landsat etm+ satellite imagery,” *Remote Sensing of Environment*, vol. 97, no. 1, pp. 39–51, 2005.
- [97] N. Kosaka, T. Akiyama, B. Tsai, and T. Kojima, “Forest type classification using data fusion of multispectral and panchromatic high-resolution satellite imageries,” in *Proceedings. 2005 IEEE International Geoscience and Remote Sensing Symposium, 2005. IGARSS’05.*, vol. 4, pp. 2980–2983, IEEE, 2005.
- [98] K. Johansen, N. C. Coops, S. E. Gergel, and Y. Stange, “Application of high spatial resolution satellite imagery for riparian and forest ecosystem classification,” *Remote sensing of Environment*, vol. 110, no. 1, pp. 29–44, 2007.
- [99] M. Immitzer, C. Atzberger, and T. Koukal, “Tree species classification with random forest using very high spatial resolution 8-band worldview-2 satellite data,” *Remote sensing*, vol. 4, no. 9, pp. 2661–2693, 2012.
- [100] B. Melville, A. Lucieer, and J. Aryal, “Object-based random forest classification of landsat etm+ and worldview-2 satellite imagery for mapping lowland native grassland communities

- in tasmania, australia,” *International journal of applied earth observation and geoinformation*, vol. 66, pp. 46–55, 2018.
- [101] N. Kussul, M. Lavreniuk, S. Skakun, and A. Shelestov, “Deep learning classification of land cover and crop types using remote sensing data,” *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 5, pp. 778–782, 2017.
- [102] K. Li, G. Cheng, S. Bu, and X. You, “Rotation-insensitive and context-augmented object detection in remote sensing images,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 4, pp. 2337–2348, 2017.
- [103] G. Cheng and J. Han, “A survey on object detection in optical remote sensing images,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 117, pp. 11–28, 2016.
- [104] Q. Zou, L. Ni, T. Zhang, and Q. Wang, “Deep learning based feature selection for remote sensing scene classification,” *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 11, pp. 2321–2325, 2015.
- [105] 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 transactions on geoscience and remote sensing*, vol. 56, no. 5, pp. 2811–2821, 2018.
- [106] A. Ducournau and R. Fablet, “Deep learning for ocean remote sensing: an application of convolutional neural networks for super-resolution on satellite-derived sst data,” in *2016 9th IAPR Workshop on Pattern Recogniton in Remote Sensing (PRRS)*, pp. 1–6, IEEE, 2016.
- [107] S. Lei, Z. Shi, and Z. Zou, “Super-resolution for remote sensing images via local–global combined network,” *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 8, pp. 1243–1247, 2017.
- [108] X. Xu, W. Li, Q. Ran, Q. Du, L. Gao, and B. Zhang, “Multisource remote sensing data classification based on convolutional neural network,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 2, pp. 937–949, 2017.

- [109] Y. Du, W. Song, Q. He, D. Huang, A. Liotta, and C. Su, “Deep learning with multi-scale feature fusion in remote sensing for automatic oceanic eddy detection,” *Information Fusion*, vol. 49, pp. 89–99, 2019.
- [110] E. Miranda, A. B. Mutiara, *et al.*, “Forest classification method based on convolutional neural networks and sentinel-2 satellite imagery,” *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 19, no. 4, pp. 272–282, 2019.
- [111] E. Ayrey, D. J. Hayes, J. B. Kilbride, S. Fraver, J. A. Kershaw, B. D. Cook, and A. R. Weiskittel, “Synthesizing disparate lidar and satellite datasets through deep learning to generate wall-to-wall regional forest inventories,” *bioRxiv*, p. 580514, 2019.
- [112] J. Irvin, H. Sheng, N. Ramachandran, S. Johnson-Yu, S. Zhou, K. Story, R. Rustowicz, C. Elsworth, K. Austin, and A. Y. Ng, “Forestnet: Classifying drivers of deforestation in indonesia using deep learning on satellite imagery,” *arXiv preprint arXiv:2011.05479*, 2020.
- [113] L. Mou, L. Bruzzone, and X. X. Zhu, “Learning spectral-spatial-temporal features via a recurrent convolutional neural network for change detection in multispectral imagery,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 2, pp. 924–935, 2018.
- [114] A. Ma, A. M. Filippi, Z. Wang, Z. Yin, D. Huo, X. Li, and B. Güneralp, “Fast sequential feature extraction for recurrent neural network-based hyperspectral image classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 7, pp. 5920–5937, 2020.
- [115] M. D. Zeiler, “Adadelata: an adaptive learning rate method,” *arXiv preprint arXiv:1212.5701*, 2012.
- [116] W. Mücke, B. Deák, A. Schroiff, M. Hollaus, and N. Pfeifer, “Detection of fallen trees in forested areas using small footprint airborne laser scanning data,” *Canadian Journal of Remote Sensing*, vol. 39, no. sup1, pp. S32–S40, 2013.
- [117] I. Güneralp, A. M. Filippi, C. R. Castillo, B. U. Hales, M. Han, C. Guidry, and C. Attaway, “Tree blowdown impacts of hurricane harvey on hydrologic surface connectivity in a coastal

- river channel-floodplain system,” in *AGU Fall Meeting Abstracts*, vol. 2018, pp. EP41D–2713, 2018.
- [118] Z. Szantoi, S. Malone, F. Escobedo, O. Misas, S. Smith, and B. Dewitt, “A tool for rapid post-hurricane urban tree debris estimates using high resolution aerial imagery,” *International journal of applied earth observation and geoinformation*, vol. 18, pp. 548–556, 2012.
- [119] J. B. Campbell and R. H. Wynne, *Introduction to remote sensing*. Guilford Press, 2011.
- [120] H. Taguchi, Y. Usuda, H. Fukui, T. Furutani, and F. Kurutawa, “Forest damage detection using high resolution remotely sensed data,” *Keio University, Endo, Fujisawa, Kanagawa, Japan*, pp. 1–4, 2005.
- [121] B. Aosier, M. Kaneko, and M. Takada, “Evaluation of the forest damage by typhoon using remote sensing technique,” in *2007 IEEE International Geoscience and Remote Sensing Symposium*, pp. 3022–3026, IEEE, 2007.
- [122] Z. M. Hamdi, M. Brandmeier, and C. Straub, “Forest damage assessment using deep learning on high resolution remote sensing data,” *Remote Sensing*, vol. 11, no. 17, p. 1976, 2019.
- [123] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.
- [124] D. E. Kislov and K. A. Korznikov, “Automatic windthrow detection using very-high-resolution satellite imagery and deep learning,” *Remote Sensing*, vol. 12, no. 7, p. 1145, 2020.
- [125] S. D. Blanchard, M. K. Jakubowski, and M. Kelly, “Object-based image analysis of downed logs in disturbed forested landscapes using lidar,” *Remote Sensing*, vol. 3, no. 11, pp. 2420–2439, 2011.
- [126] E. Lindberg, M. Hollaus, W. Mücke, J. E. Fransson, and N. Pfeifer, “Detection of lying tree stems from airborne laser scanning data using a line template matching algorithm,” *Proceedings of ISPRS Annals II-5 W*, vol. 2, pp. 11–13, 2013.



- [127] P. Polewski, W. Yao, M. Heurich, P. Krzystek, and U. Stilla, "Detection of fallen trees in als point clouds using a normalized cut approach trained by simulation," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 105, pp. 252–271, 2015.
- [128] P. Polewski, W. Yao, M. Heurich, P. Krzystek, and U. Stilla, "Learning a constrained conditional random field for enhanced segmentation of fallen trees in als point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 140, pp. 33–44, 2018.
- [129] F. Duan, Y. Wan, and L. Deng, "A novel approach for coarse-to-fine windthrown tree extraction based on unmanned aerial vehicle images," *Remote Sensing*, vol. 9, no. 4, p. 306, 2017.
- [130] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [131] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets*, pp. 267–285, Springer, 1982.
- [132] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *2012 IEEE conference on computer vision and pattern recognition*, pp. 3642–3649, IEEE, 2012.
- [133] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.
- [134] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.
- [135] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, pp. 91–99, 2015.

- [136] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.
- [137] S. Ji, W. Xu, M. Yang, and K. Yu, “3d convolutional neural networks for human action recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2012.
- [138] J. Huang, W. Zhou, Q. Zhang, H. Li, and W. Li, “Video-based sign language recognition without temporal segmentation,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [139] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*, pp. 740–755, Springer, 2014.
- [140] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.
- [141] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [142] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.