МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет аеронавігації, електроніки та телекомунікацій
Кафедра авіаційних комп’ютерно-інтегрованих комплексів

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____ В. М. Синєглазов
«___» _____ 2023р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬО-КВАЛІФІКАЦІЙНОГО РІВНЯ
“БАКАЛАВР”
Спеціальність 151 «Автоматизація та комп'ютерно-інтегровані технології»
Освітньо-професійна програма «Комп’ютерно-інтегровані технологічні процеси і виробництва»

Тема: Класифікатор гіперспетральних зображень

Виконавець: студент групи ФАЕТ-404 Люсік Богдан Адамович
Керівник: кандидат технічних наук, Гордієнко Олександр

Нормоконтролер:_____Філяшкін М. К.
(підпис)

Київ 2023

EDUCATION AND SCIENCE MINISTRY OF UKRAINE

NATIONAL AVIATION UNIVERSITY

Faculty of Aeronavigation, Electronics and Telecommunications

Department of computer integrated complexes

ADMIT TO DEFENSE

Head of the graduate department

_____ Viktor M. Sineglazov

«___» _____ 2023

QUALIFICATION WORK

(EXPLANATORY NOTE)

GRADUATE OF EDUCATION AND QUALIFICATION LEVEL

“BACHELOR”

Specialty 151 "Automation and computer-integrated technologies"

Educational and professional program "Computer-integrated technological processes and production"

Theme: Classifier of hyperspectral images

Performer: student of group FAET-404 Liusik Bohdan Adamovych

Supervisor: Candidate of Technical Sciences, Oleksandr Hordiienko

Normocontroller: _____ Filyashkin M. K.

(signature)

Kyiv 2023

NATIONAL AVIATION UNIVERSITY

Faculty of aeronavigation, electronics and telecommunications

Department of Aviation Computer Integrated Complexes

Educational level: bachelor

Specialty: 151 "Automation and computer-integrated technologies"

APPROVED

Head of Department

Sineglazov V. M.

«____» _____2023

TASK

For the student's thesis

Liusik Bohdan Adamovych

1. Theme of project: “Classifier of hyperspectral images”.
2. The term of the project: from May 10, 2023, until June 7, 2023
3. Output data to the project: classification of objects in hyperspectral images with the help of a neural network using own method.
4. Contents of the explanatory note: 1. Remote Earth sensing: the purpose of remote sensing, main characteristics of satellite images, general characteristics of space remote sensing systems; 2. Description of the subject of research: how hyperspectral images are created, their advantages over multispectral images; 3. Modern systems and methods of object classification: description of the systems currently used for classification; 4. Software implementation of object classification on hyperspectral images.
5. List of required illustrative material: tables, figures, diagrams, graphs.
6. Planned schedule.

№	Task	Execution term	Execution mark
1.	Getting the task	01.04.2023 – 02.04.2023	Done
2.	Formation of the purpose and main objectives of the study	02.04.2023 – 14.04.2023	Done
3.	Analysis of existing methods	15.04.2023 – 30.04.2023	Done
4.	Theoretical consideration of problem solving	01.05.2023 – 05.05.2023	Done
5.	Software implementation of the hyperspectral image classification program	06.05.2023 – 25.05.2023	Done
6.	Preparation of an explanatory note	26.05.2023 – 03.06.2023	Done
7.	Preparation of presentation and handouts	04.06.2023 – 06.06.2023	Done

7. Date of task receiving: «___» _____ 2023.

Diploma thesis supervisor _____ Hordiienko Oleksandr

(signature)

Issued task accepted _____ Liusik Bogdan

(signature)

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет аеронавігації, електроніки та телекомунікацій

Кафедра авіаційних комп’ютерно-інтегрованих комплексів

Освітній ступінь: Бакалавр

Спеціальність: 151 «Автоматизація та комп'ютерно-інтегровані технології»

ЗАТВЕРЖДУЮ

Завідувач кафедри

Синєглазов В. М.

«___» _____2023

ЗАВДАННЯ

На виконання дипломної роботи

Люсіка Богдана Адамовича

1. Тема проекту: «Класифікатор гіперспктральних зображень».
2. Термін виконання роботи: з 10.05.2023р. по 07.06.2023р.
3. Вихідні дані роботи: класифікація об'єктів на гіперспектральних зображеннях за допомогою нейронної мережі використовуючи власний методом.
4. Зміст пояснювальної записки: 1. Дистанційне зондування Землі: мета дистанційного зондування, основні характеристики супутникових знімків, загальна характеристика космічних систем дистанційного зондування; 2. Опис предмету дослідження: як створюються гіперспектральні зображення, їх переваги над багатоспектральними зображеннями; 3. Сучасні системи та методи класифікації об'єктів: опис систем, які використовуються для класифікації; 4. Програмна реалізація класифікації об'єктів на гіперспектральних зображеннях.

5. Перелік обов’язкового графічного матеріалу: таблиці, рисунки, діаграми, графіки.
6. Календарний план-графік.

№	Завдання	Термін виконання	Підпис керівника
1.	Отримання завдання	01.04.2023 – 02.04.2023	Виконано
2.	Формування мети та основних завдань дослідження	02.04.2023 – 14.04.2023	Виконано
3.	Аналіз існуючих методів	15.04.2023 – 30.04.2023	Виконано
4.	Теоретичний розгляд рішення задач	01.05.2023 – 05.05.2023	Виконано
5.	Програмна реалізація програми класифікації гіперспектральних зображень	06.05.2023 – 25.05.2023	Виконано
6.	Оформлення пояснювальної записки	26.05.2023 – 03.06.2023	Виконано
7.	Підготовка презентації та роздаткового матеріалу	04.06.2023 – 06.06.2023	Виконано

7. Дата видачі завдання: «__» _____ 2023р.

Керівник дипломної роботи (проекту): _____ Гордієнко Олександр

(підпис)

Завдання прийняв до виконання: _____ Люсік Богдан

(підпис)

ABSTRACT

Explanatory note of the qualification work "Hyperspectral image classifier" 58 p., 18 figs., 4 tables, 26 sources.

HYPERSPECTRAL IMAGES, REMOTE SENSING OF THE EARTH, NEURAL NETWORKS, OBJECT CLASSIFICATION.

The object of research is hyperspectral image.

Subject of research - a detailed study of object classification in hyperspectral images.

Purpose of the qualification work - software implementation of object classification on hyperspectral images, comparison of the method's effectiveness with existing ones.

Research methods - comparative analysis, processing of literature sources, digital mathematical modeling.

The paper covers the topic of object classification on hyperspectral images obtained by remote sensing, providing a detailed analysis of modern classification systems and methods. The research begins with an overview of remote sensing, detailing its purpose and main characteristics of satellite images, as well as a comprehensive study of space-based remote sensing systems.

The main contribution of the paper is the software implementation of object classification in hyperspectral images. This new approach demonstrates how advanced machine learning algorithms can analyze and classify complex hyperspectral data, presenting exciting potential for expanding existing remote sensing capabilities.

The results of the study promise significant progress in remote sensing, object classification, and hyperspectral image analysis.

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи «Класифікатор гіперспектральних зображень» 58 с., 18 рис., 4 табл, 26 джерел.

ГІПЕРСПЕКТРАЛЬНІ ЗОБРАЖЕННЯ, ДИСТАНЦІЙНЕ ЗОНДУВАННЯ ЗЕМЛІ, НЕЙРОННІ МЕРЕЖІ, КЛАСИФІКАЦІЯ ОБ'ЄКТІВ.

Об'єкт дослідження – гіперспектральне зображення.

Предмет дослідження - детальне вивчення класифікації об'єктів на гіперспектральних знімках.

Мета кваліфікаційної роботи - Програмна реалізація класифікації об'єктів на гіперспектральних знімках, порівняння ефективності методу з існуючими.

Метод дослідження - порівняльний аналіз, обробка літературних джерел, цифрове математичне моделювання.

Робота висвітлює тему класифікації об'єктів на гіперспектральних зображення, отриманих за допомогою дистанційного зондування, надаючи детальний аналіз сучасних систем і методів класифікації. Дослідження починається з огляду дистанційного зондування Землі, деталізації його мети та основних характеристик супутникових знімків, а також всебічного вивчення космічних систем дистанційного зондування.

Основний внесок роботи полягає в програмній реалізації класифікації об'єктів на гіперспектральних зображеннях. Цей новий підхід демонструє, як передові алгоритми машинного навчання можуть аналізувати і класифікувати складні гіперспектральні дані, представляючи захоплюючий потенціал для розширення існуючих можливостей дистанційного зондування.

Результати дослідження обіцяють значний прогрес у сфері дистанційного зондування, класифікації об'єктів та аналізу гіперспектральних зображень.

CONTENT

GLOSSARY

CNN – Convolutional Neural Network

SVM – Support Vector Machine

ANN – Artificial Neural Network

MDC – Minimum Distance Classifier

MLC – Maximum Likelihood Classifier

DBN – Deep Belief Network

SSRN – Spectral–Spatial Residual Network

SA – Salinas Scene

UP – University of Pavia

IP – Indian Pines

OA – Overall Accuracy

AA – Average Accuracy

Kappa – Kappa Coefficient

HSI – Hyperspectral Image

PROBLEM STATEMENT

The field of remote sensing has seen incredible advances over the past few decades, with hyperspectral imaging forming a significant component of this progress. This cutting-edge technology, which captures and processes information from across the electromagnetic spectrum, enables more accurate identification of objects and materials than traditional imaging methods. However, despite its potential, the classification of hyperspectral images (HSIs) poses unique challenges that demand rigorous exploration. The primary aim of this diploma is to address these challenges, focusing on the development and evaluation of advanced classification techniques for hyperspectral imaging data.

Hyperspectral images carry rich information as they consist of hundreds of contiguous spectral bands. This high dimensionality, however, brings about the 'curse of dimensionality,' where the increased complexity in handling, processing, and interpreting these images becomes a challenge. This diploma aims to address this problem by developing effective dimension reduction method and feature extraction techniques that will simplify the classification process, while preserving the maximum amount of spectral information.

Through an in-depth analysis, experimentation, and implementation, this diploma aims to push the frontiers of current hyperspectral image classification methodologies. It will strive to develop a scalable, efficient, and accurate classification system capable of handling the challenges associated with hyperspectral data, thus paving the way for a broader range of applications in areas like agriculture, mineralogy, environmental science, and defense.

1 REMOTE SENSING OF THE EARTH

Remote Sensing is a scientific method used to acquire information about the Earth's surface without making any direct physical contact. It involves the use of sensors on satellites or aircraft to collect data about the environment. The sensors measure the radiation that is reflected or emitted from the Earth’s surface. Different types of remote sensing techniques include aerial photography, satellite imagery, radar, and sonar, each serving specific purposes in various fields [1].

There are two types of remote sensing: passive and active. Passive remote sensing involves recording radiation that is naturally reflected or emitted by the Earth's surface or the atmosphere, for example, sunlight reflected by forests or fields. On the other hand, active remote sensing systems like radar or LIDAR, emit their own energy to scan objects and areas where they then measure the reflection. Remote sensing data is processed and interpreted using sophisticated algorithms, often to produce a 2D or 3D image that allows for analysis and interpretation [2].

1.1 The purpose of remote sensing

Space-based remote sensing systems are designed to provide socio-economic sectors and public authorities with observation data on natural and man-made objects, phenomena, and events. The development of space technology and information technologies has created scientific and technical capabilities for high-resolution space sensing of the Earth [3]. To conduct such sensing, optoelectronic devices (OEDs), synthetic aperture radar (SAR) and space photographic equipment (SPE) are installed on spacecraft (SP). The experience of using space-based observation systems shows great potential for using the results of remote sensing of

the Earth in solving a wide range of problems in almost all sectors of the economy and social sphere.

Remote sensing of the Earth provides unique opportunities for operational data collection on a global scale with high spatial, spectral and temporal resolution, which determines the great information capabilities of space systems, the possibility of their military use and potential economic efficiency. The systematic approach requires the division of the set of tasks of space remote sensing means by indicators of scientific, industrial, economic and social orientation, namely:

- control of weather and climate factors;
- monitoring the state of sources of air, water and soil pollution;
- control of man-made and natural emergencies nature;
- information support of economic activity, rational land use, rational land use;
- information support of national security and defense;
- creation of a dynamic model of the Earth as an ecological system.

Nowadays, various thematic tasks are successfully performed using remote sensing methods to provide information on scientific, economic, national security and defense issues, among others:

1. Inventory of agricultural land, allocation and identification of crop types, crop forecasting, and analysis of agricultural potential.

2. Monitoring global atmospheric changes - measuring surface temperature, determining surface conditions, determining the state of the atmosphere, observing cloud cover, and studying the greenhouse effect.

3. Search for minerals and energy resources (oil, natural gas, coal).

4. Topographic mapping, map creation and updating, monitoring urban growth, and monitoring the condition of soils and pastures.

5. Observation of coastal zones and oceans, control of water sources - studying and determining ocean resources, measuring ice thickness, determining snow cover and its water equivalent, identifying places and sources of water pollution.

6. Monitoring the condition of forests, determining the types of forest plantations and dominant species, assessing timber reserves, and logging.

7. Monitoring of emergency situations - prevention, control and assessment of the effects of floods, fires and earthquakes.

8. Defense surveillance - determining the condition of military, military-industrial and engineering facilities, monitoring border areas, and controlling mass movements of troops.

For military systems, the main task is space reconnaissance.

1.2 Main characteristics of satellite images

The effectiveness of space image analysis and interpretation is determined by the content and volume of information about remote sensing objects, the list of which is determined by the thematic task. As you know, space images are formed by recording electromagnetic radiation reflected or generated by earth formations and artificial (anthropogenic) objects. Different objects of remote sensing have different spectral and energy characteristics of radiation and differ in geometric size, shape and behavior in time and space [4]. All these features of remote sensing objects should be considered when choosing a space system that will be used to generate images.

First, the following characteristics are considered:

- The spectral range in which the objects and processes under observation and study are active;

- The degree of detail of observation and registration of the geometric shape of objects and spatial relationships;

- Radiometric resolution, or the maximum number of bits that quantizes the dynamic range of pixel brightness's of images of earth surface objects;

- Area (geometric dimensions of the survey frame) of the scene - a certain area of the Earth's surface to be observed;

- Guaranteed provision of one-time control or monitoring (periodic observation with a certain time interval) of a certain geographical area.

1.3 General characteristics of space remote sensing systems

A space system (SS) is a set of coordinated, functionally interconnected spacecraft and ground-based technical means designed to solve targeted tasks.

The space remote sensing system includes a space complex and a ground-based information complex (GBIC). Space observation complex is a set of functionally interconnected orbital and ground-based means designed to independently solve special tasks from space or to ensure the fulfillment of such tasks as part of the space observation system. The space complex includes: a spacecraft or a group of spacecraft, a rocket and space complex, a control and reference complex, a ground control complex, a spacecraft landing and maintenance complex.

Thus, space observation data acquisition and dissemination systems are based on the following main components:

- carriers of imaging equipment, in this case, artificial earth satellites (AES);
- the actual remote sensing equipment;
- onboard means of data transmission to Earth;

- a ground-based information system for receiving this information, processing it and providing it to consumers.

The classification of remote sensing systems is their division into classes (subclasses, groups) based on the commonality of homogeneous essential features (properties), which fixes the natural relationships between classes of systems in a particular field of knowledge. The characteristics of the above components of the system of space observation data acquisition and dissemination or their parameters are most often the basis for the classification of space remote sensing systems.

Modern space systems can be divided into scientific, military, and commercial according to the purpose and content of the tasks they solve.

In turn, the scientific ones include research and experimental manned and automatic space stations, research spacecraft that conduct research on planets and stars, outer and interstellar space, geophysical research of the Earth, and experimental ones: scientific and military experimental spacecraft that conduct scientific experiments and test elements of advanced spacecraft. This division is purely arbitrary. In practice, most scientific satellites are multifunctional, i.e., they contain research, scientific and experimental devices.

Commercial spacecraft are designed to solve economic problems, provide all types of communications and telecommunications, and facilitate the safety of land, air and sea traffic. Commercial spacecraft include domestic spacecraft (as a rule, these are dual-purpose spacecraft, i.e. only spacecraft that, if necessary, can be used in full or in part to solve problems in the interests of armed struggle) and domestic spacecraft leased by other states or launched in the interests of other states.

The set of space complexes and systems for military purposes constitutes space weapons. Military space systems are divided into combat and support systems according to the tasks they perform. Combat spacecraft are designed to conduct combat operations in space or from space, or are the space part of combat ground-

space complexes (systems). These are strike spacecraft, space-based missile defense and air defense systems, electronic warfare and missile launch detection spacecraft. Combat support spacecraft are designed to support the daily and combat activities of all branches of the armed forces. They are classified as reconnaissance, navigation, communication, meteorological, topographic, and transportation.

According to the definition of the Scientific and Technical Subcommittee of the UN Committee on Space, remote sensing is "the observation and measurement of energy and polarization characteristics of the intrinsic and reflected radiation of the Earth's land, ocean and atmosphere in different ranges of electromagnetic waves, which help to describe the location, nature and temporal variability of natural parameters and phenomena, the Earth's natural resources, the environment, as well as anthropogenic objects and formations". As can be seen from the above definition, remote sensing methods allow for different types of classification: by the spectral range of the electromagnetic radiation used, by the type of signal recorded (own or reflected, natural or directed from an artificial radiation source); by image parameters (spatial resolution, spectral resolution, viewing frequency, frame size on the ground, speed of application execution, rights to distribute and copy images); by the characteristics of the imaging equipment carriers and its parameters, etc. Image parameters and overview characteristics depend on the parameters of the spacecraft trajectory and the characteristics of its onboard special equipment.

1.4 Overview of modern hyperspectral sensors

Hyperspectral sensors, such as the Reflective Optics System Imaging Spectrometer (ROSIS) and the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS), are vital tools in remote sensing, used for a wide range of scientific, environmental, and industrial applications. They capture high-resolution image data

across the electromagnetic spectrum, allowing for detailed analysis and interpretation of the Earth's surface and atmosphere.

Characteristic	ROSIS-03	AVIRIS
Angular field of view (FOV)	16°	30°
Instantaneous field of view (IFOV)	0.56 mrad	0.95 mrad
Number of pixels per line	512	614
Scan principle	Pushbroom	Whiskbroom
Ground resolution	1 m–6 m	20 m
Radiometric resolution	14 bits	10 bits
Spectral range	430 nm–800 nm	400 nm–2450 nm
Spectral sampling	4 nm	9.6 nm–10.0 nm
Inflight calibration	0.2 nm	0.5 nm

Figure 1.1 Main characteristics for both ROSIS-03 and AVIRIS sensors.

Here are several satellites they are installed on:

1. MODIS (Moderate Resolution Imaging Spectroradiometer) – mounted on the Terra and Aqua satellites, launched by NASA (USA) in 1999 and 2002 respectively. MODIS provides high-frequency imagery in a wide spectrum (36 spectral bands, ranging from 0.4 to 14.4 μm) and is used for monitoring global-level processes, including vegetation dynamics, carbon cycling, and water cycles.
2. Landsat series – this is a series of American Earth-observing satellites. The latest of these, Landsat 9, was launched in 2021. Landsat satellites employ the Multispectral Scanner (MSS), Thematic Mapper (TM), Enhanced Thematic Mapper (ETM), and Operational Land Imager (OLI). They provide imagery of the Earth's surface in the visible, near-infrared, and thermal infrared spectrums.

3. Sentinel series – these are satellites from the European Space Agency, launched as part of the Copernicus Earth observation program. Sentinel-2, for example, has a Multispectral Instrument (MSI) for observations in the visible, near-infrared, and shortwave infrared spectrums.
4. ASTER (Advanced Spaceborne Thermal Emission and Reflection Radiometer) – this instrument is mounted on the Terra satellite. Launched by NASA in 1999, ASTER provides imagery in 14 channels of the visible, near-infrared, and thermal infrared spectrums.
5. WorldView-3 – a commercial satellite, launched by DigitalGlobe (now Maxar Technologies) in 2014. WorldView-3 can provide high-resolution imagery and multispectral images.

2 DESCRIPTION OF THE RESEARCH OBJECT

In the past several decades, the utility of satellites in various domains such as earth monitoring, remote sensing, communication, and navigation has been proven effective. Remote sensing, as per the context of my work above, is the technique of obtaining data from a particular object without making direct physical contact with it. Each object, due to differences in their molecular composition, uniquely absorbs and emits the incident electromagnetic radiation. This interaction of radiation with the object results in a specific pattern called a spectral signature, which can be utilized to identify any material, given its unique nature for each substance found on Earth's surface.

The concept is as follows: by observing the spectral signature or spectral response, we can accurately identify the materials or objects featured in the hyperspectral image we've captured. Hence, hyperspectral sensors have been designed to detect radiation across an expansive wavelength range present in the electromagnetic spectrum. This range encompasses the visible, short, mid, and long-wave infrared region, with each region having a breadth of about 10nm [5].

The emission of radiation from a scene, captured at a specific wavelength as an image, is organized in layers (each representing different wavelengths) to construct a hyper-spectral data-cube, as illustrated in Figure 2.1.

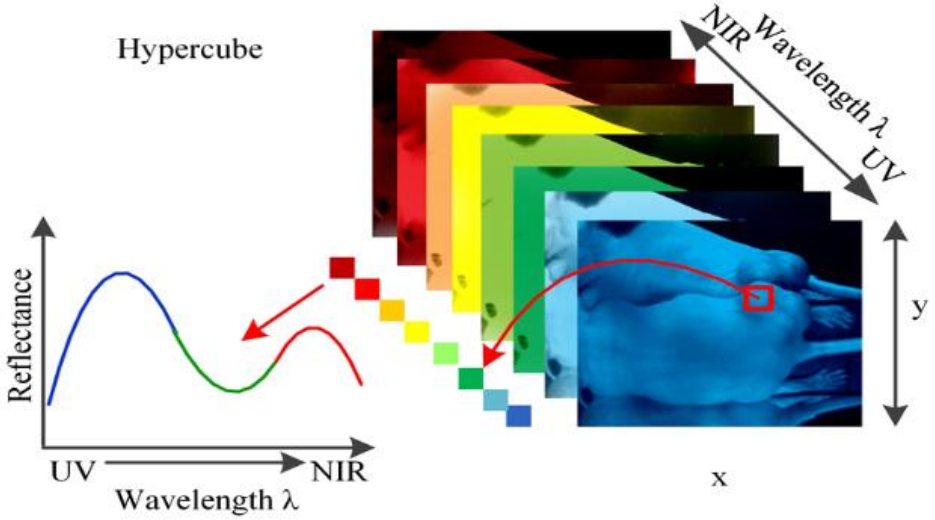

Figure 2.1. Hyperspectral data cube with spectral signature.

The hyperspectral data-cube's spatial information is conveyed through the x-y plane, while its spectral content is depicted in the z-plane. Every hyperspectral image band has a dimension where each pixel signifies a digital number (DN), which corresponds to the radiance value gathered by the sensor (IFOV). Notably, each band corresponds to a specific wavelength. Typically, the HSI data cube (a 3D hypercube) is represented as a $\chi \in^{n_1 \times n_2 \times n_b}$, where $n = n_1 \times n_2$ indicates the total pixel count, and nb denotes the number of bands.

Each pixel in the spectral space, created by the number of bands, is represented as a single-dimensional vector. Materials of a similar kind are categorized using clustering algorithms, which are based on spectral properties that are close to one another. Widely used clustering algorithms in hyperspectral image analysis include k-means clustering, fuzzy c-means clustering, and clustering methods based on spectral unmixing. Given the high correlation in the spectral space, the data is portrayed in a lower dimensional space, smaller than the number of spectral bands. The reduction in data dimensionality is achieved using techniques such as principal component analysis (PCA) or independent component analysis

(ICA). In this scenario, an image is displayed as a matrix in spatial space. Similarly to spectral properties, spatial properties of like materials are closely related. The practice of grouping materials based on spatial properties is called segmentation. Meanwhile, the concurrent processing of a pixel based on adjacent pixels in the spectral space, along with band processing based on neighboring bands in the spatial space, is termed spectral-spatial representation [6].

2.1 Creating hyperspectral images

Hyperspectral imaging, an advanced technique that collects and processes data from across the electromagnetic spectrum, outperforms traditional spectral imaging methods by providing more detailed and comprehensive information [7]. Unlike the human eye, which can only perceive light in three bands (red, green, and blue), hyperspectral imaging partitions the spectrum into numerous bands. It can even capture data beyond the visible range, making it versatile for a wide array of applications.

This technology plays a crucial role in diverse fields such as agriculture, mineralogy, physics, surveillance systems, and forensics. The backbone of hyperspectral imaging is the hyperspectral sensor which investigates an object by using most of the electromagnetic spectrum. Unique 'fingerprints' are produced by certain objects across this spectrum range, identified as spectral features of matter. This information can be used to identify and characterize the materials present in the subject of the study. For instance, mineralogists can locate new oil deposits by analyzing the distinctive spectral lines of oil.

Hyperspectral detectors generate data as a collection of "images", where each image represents a different spectral range within the electromagnetic spectrum. These images are subsequently merged to form a three-dimensional hyperspectral

data volume. This data structure is conducive for comprehensive analysis and processing, granting a thorough insight into the object under investigation.

The efficiency of hyperspectral cameras is gauged primarily by the spectral resolution, or the width of each band of the captured spectrum. If the object's spectrum contains a large number of adequately narrow frequency bands, the identification of objects is possible even if they only span a few pixels in the image. However, spatial resolution plays a complementary role to spectral resolution. A large pixel size can capture multiple objects within the same pixel, complicating differentiation. Conversely, a small pixel size can result in low light energy reception per sensor pixel, leading to a decrease in the signal-to-noise ratio and compromised parameter measurement accuracy.

Three main methods are employed in hyperspectral image processing technology:

1. Spatial image scanning sequentially captures total spectral data.

2. Spectral image scanning sequentially captures complete spatial information.

3. The "snapshot" method captures all spectral and spatial information simultaneously.

Two key benefits of this type of spectrometer, which influence its speed, include:

1. Absence of spectral scanning allows for real-time examination of all spectral components (a concept known as Fellgett's advantage in metrology).

2. FT-IR spectrometers feature larger apertures than those in dispersive spectrometers due to their high bandwidth (also known as the Jacquinot or bandwidth advantage).

Two predominant types of interferometers - the Michelson interferometer and the Fabry-Perot interferometer - are commonly utilized in this technology. These

devices offer superior speed performance compared to other spectral or spatial instruments, contributing to the enhanced efficiency of hyperspectral imaging.

2.2 Advantages of hyperspectral images over multispectral ones

Hyperspectral imaging offers a key benefit by capturing the entire spectrum at each point, eliminating the need for the operator to possess prior knowledge of the sample. Through postprocessing, it becomes possible to extract all the valuable information from the dataset. Moreover, hyperspectral imaging leverages the spatial connections between various spectra in a given area, enabling the use of sophisticated spectral-spatial models that enhance the precision of image segmentation and classification.

Hyperspectral imaging surpasses multispectral imaging [8, 9, 10] in terms of its numerous advantages, which are as follows:

1. Hyperspectral remote sensing data exhibits high spatial resolution, providing detailed and precise information about the observed area. This level of detail allows for more accurate analysis and interpretation of the data.

2. Hyperspectral data is typically collected within a specific and well-defined spectral range. This focused range enables targeted analysis of specific materials, phenomena, or characteristics within the captured scene.

3. The bands of hyperspectral data are contiguous and overlapping, ensuring that no valuable information is missed. This continuous coverage allows for the detection of subtle variations and nuanced features in the scene, enhancing the overall understanding of the data.

4. The contiguous spectrum obtained from hyperspectral imaging facilitates the identification of atmospheric windows. This information is crucial for effectively removing atmospheric interference from the radiance signal, resulting in cleaner and

more accurate data. In contrast, multispectral sensors lack the continuous spectrum necessary for identifying atmospheric windows.

5. The signal-to-noise ratio of hyperspectral data can be improved by comparing pixel spectra. This comparative analysis helps reduce noise and enhance the quality of the data. Conversely, multispectral data, with its non-contiguous bands, does not lend itself well to this type of pixel-based noise reduction.

6. Hyperspectral imaging provides a solution to the challenge of mixed spectra. By directly deriving the relative abundance of materials, it becomes possible to identify and analyze the composition of complex scenes accurately. This capability is particularly valuable in applications such as environmental monitoring, geology, and agriculture.

7. Hyperspectral images offer the flexibility to derive information from various spaces. This includes the spectral space, where the unique spectral signatures of objects or classes can be identified; the image space, which allows for spatial analysis and pattern recognition; and the character space, where additional contextual information about the scene can be extracted. This multi-dimensional approach enhances the overall comprehension and utilization of hyperspectral data.

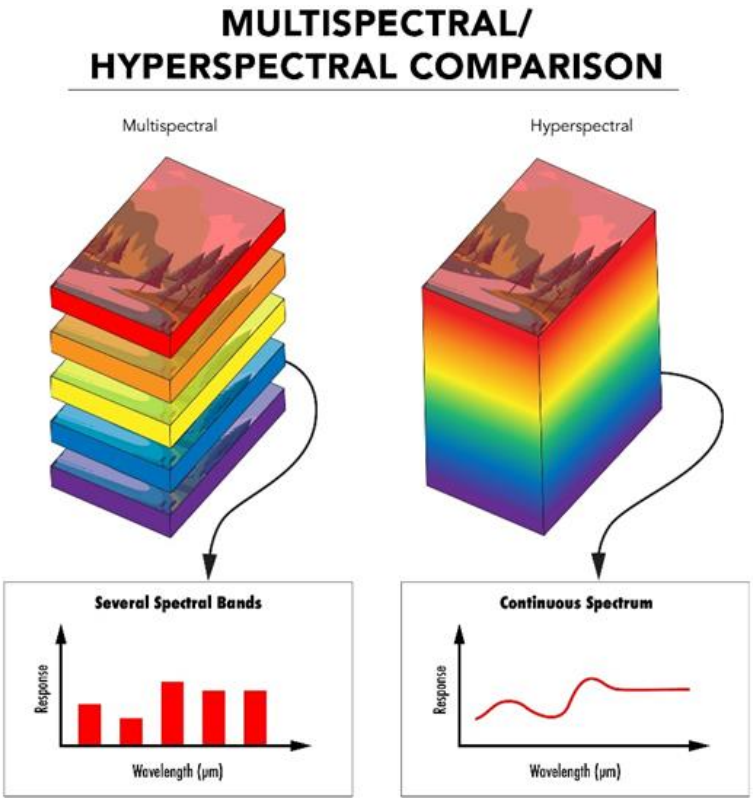

Figure 3.1 In multispectral imaging, image stacks consist of multiple images captured in different spectra, while hyperspectral imaging involves image stacks with a much larger number of images taken in numerous spectra.

2.3 Conclusions

Hyperspectral imaging has revolutionized remote sensing by providing a high level of detail and precision in the acquisition of data. Its ability to detect radiation across a vast wavelength range, construct a detailed hyperspectral data-cube, and identify unique spectral signatures have proven to be invaluable in various applications from environmental monitoring to military usage and medical diagnostics. Importantly, the advances in hyperspectral imaging have enabled a superiority over multispectral imaging, offering numerous advantages including

high spatial resolution, the detection of subtle variations, and the ability to analyze complex scenes accurately.

The recent ease of acquiring high-resolution hyperspectral remote sensing images has increased the application of this technology in various fields. The primary focus of ongoing research in this domain is the classification of hyperspectral images. However, several challenges exist, such as high dimensionality, limited availability of labeled samples, spatial variability of spectral information, and image quality. A plethora of classification methods and dimension reduction techniques are continually being explored and developed, including the use of machine learning techniques such as support vector machines, random forests, neural networks, and more recently, deep learning networks.

Despite these challenges, the potential of hyperspectral imaging is vast and continues to expand with technological advancements. The continuous improvements in image acquisition, processing, and classification techniques will further enhance the quality of data derived from hyperspectral imaging and broaden its application range.

3 MODERN SYSTEMS AND METHODS OF OBJECT CLASSIFICATION

In recent years, the acquisition of hyperspectral remote sensing images with high spatial and spectral resolution has become relatively easier, finding wide applications in environmental, military, mining, and medical fields. These images, captured using imaging spectrometers, possess high spectral resolution, numerous bands, and abundant information. Hyperspectral image processing includes image correction, noise reduction, transformation, dimensionality reduction, and classification. Classification [13] remains the most active research area within the hyperspectral domain, as the rich spectral information reflects the physical structure and chemical composition of objects.

However, hyperspectral image classification faces challenges such as high dimensionality, lack of labeled samples, spatial variability of spectral information, and image quality. Researchers have developed various classification methods, including support vector machines, random forests, and neural networks, as well as dimension reduction techniques like principal component analysis and linear discriminant analysis. More recently, the incorporation of spatial context information has gained attention, with deep learning networks like convolutional neural networks and deep belief networks being used in remote sensing image processing. Hyperspectral image classification methods are broadly categorized into supervised, unsupervised, and semisupervised classifications.

3.1 Supervised classification

Supervised classification is a frequently employed method for hyperspectral image classification. The fundamental procedure involves establishing discriminant

criteria based on known sample categories and prior knowledge, followed by calculating the discriminant function. Widely used supervised classification techniques encompass support vector machine, artificial neural network classification, decision tree classification, and maximum likelihood classification methods.

3.1.1 Support vector machines

The Support Vector Machine (SVM) [12], a supervised classification approach, was formulated by Boser and his team. It leverages statistical theory and the principle of structural risk minimization and is instrumental in the realms of image and signal processing and recognition. SVM finds the optimal classification surface by applying structural risk minimization to linear classifiers. In practice, not all situations are linearly separable, so slack variables are introduced. For nonlinear cases, kernel functions [13] are used, transforming the input space into a high-dimensional space, and finding the optimal linear classification surface in the new space. Commonly used kernel functions include linear, polynomial, and Gaussian kernel functions. Figure 3.1 illustrates a conceptual representation of a support vector machine using a kernel function.

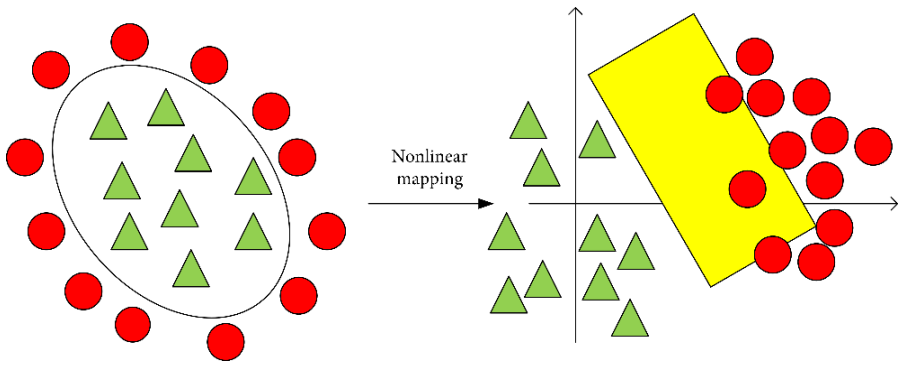

Fig. 3.1 Kernel function support vector machine diagram.

3.1.2 Minimum distance classification

The Minimum Distance Classifier (MDC) [14] is a supervised classification technique that operates based on the proximity of pixels within a feature space. It assumes that feature points of the same class cluster in space, using the mean vector as the category center and the covariance matrix to describe dispersion. Various distance calculations, such as Mahalanobis and Barth-Parametric distances, are used to measure similarity. MDC is an early method for image classification research, and its simplicity and intuitiveness make it widely used even today. For classifications with limited training samples, it can yield better results than more complex classifiers. Figure 3.2 is a flowchart of the minimum distance classification method.

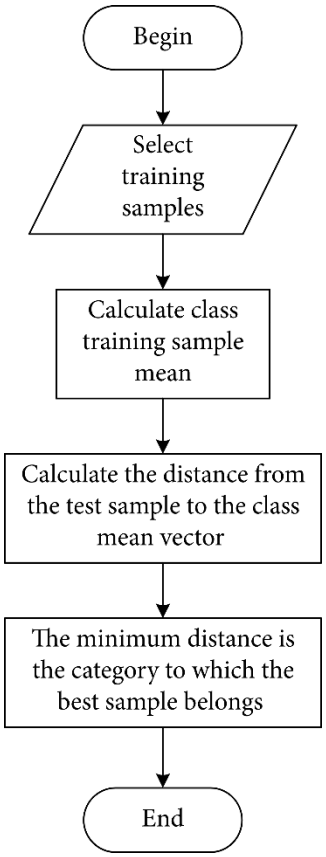

Figure 3.2 Schematic diagram of minimum distance classification.

3.1.3 Maximum likelihood classification

Maximum Likelihood Classifier (MLC) [15] is a nonlinear classification method based on the Bayesian criterion. It calculates statistical feature values of training samples to establish a discriminant function, which is used to determine the probability of each pixel in a hyperspectral image belonging to various classes. The test sample is classified into the category with the highest probability. MLC generally obtains better results, especially when training samples are normally distributed. It assumes a normal distribution of hyperspectral data and uses a likelihood decision function to determine conditional probability.

3.1.4 Neural network classification

Artificial Neural Networks (ANN) are prevalent artificial intelligence classification systems that mimic the information processing of human neurons. They find utility in intelligent control, information processing, and combinatorial optimization. Nevertheless, they come with certain limitations such as the need for vast amounts of training data, reduced processing speeds, and challenges in deriving decision boundaries in the feature space. Backpropagation [17] neural networks are the most widely used ANN model, consisting of input, hidden, and output layers. The implementation process includes two stages: network self-learning to optimize connection weights and using learning results to classify image data.

Compared to other methods, SVM requires fewer training samples but struggles with large-scale samples and multi-classification problems. Minimum distance classification is fast but less accurate, while maximum likelihood, minimum distance, and neural network methods can be used in practice with human supervision to ensure accuracy.

3.2 Deep learning

In recent years, hyperspectral image classification techniques have incorporated spatial information from hyperspectral images, leading to the development of methods based on combined spatial-spectral features. Deep learning [18], derived from artificial neural networks, offers more robust feature extraction capabilities compared to its predecessor. Deep learning models possess multiple layers, further enhancing feature information extraction. This section primarily explores deep learning techniques, such as convolutional neural networks (CNN), deep belief networks (DBN), and stacked autoencoders (SAE).

3.2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [18] are analogous to traditional Artificial Neural Networks (ANNs) as they consist of self-optimizing neurons. Each neuron takes an input, performs an operation, and contributes to the final class score, just like in ANNs. The final layer contains class-associated loss functions, and all common ANN strategies still apply.

However, CNNs stand out in their primary use for pattern recognition in images, enabling us to incorporate image-specific features into the network architecture. This makes CNNs more suitable for image-related tasks and reduces the parameters needed for the model.

A key limitation of ANNs is their struggle with the computational complexity of image data. For instance, they can handle datasets like the MNIST database of handwritten digits, with its manageable 28 × 28 image dimensionality. However, for a larger colored image input of 64 × 64, the number of weights for a single neuron

in the first layer jumps to 12,288, necessitating a significantly larger network. This showcases the challenges of using such models for larger, more complex image data.

3.2.1.1 CNN architecture

CNNs are designed primarily for image inputs, structuring the architecture to handle this data type effectively. Neurons in CNNs are organized in three dimensions: height, width, and depth of the input. The depth refers to the third dimension of an activation volume, not the total number of layers. Neurons in a layer connect only to a small region of the preceding layer. So, for a 64 × 64 × 3 input volume (height, width, depth), the final output layer will have a 1 × 1 × n dimensionality, where n represents possible classes.

3.2.1.2 Overall architecture

CNNs consist of three primary types of layers: convolutional layers, pooling layers, and fully connected layers. The stacking of these layers creates a CNN architecture. Figure 3.3 showcases a simplified representation of a CNN architecture designed for MNIST classification.

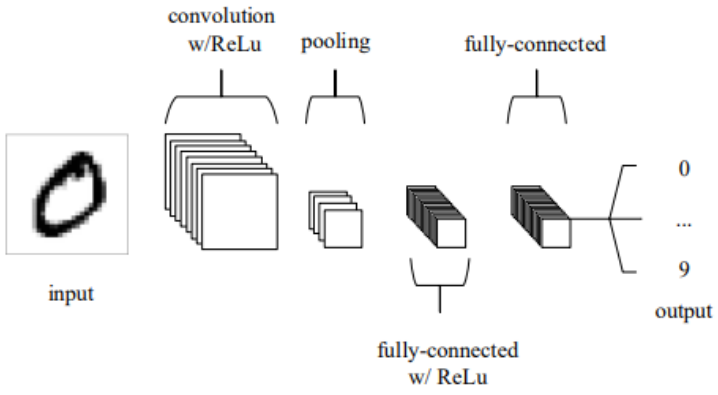

Figure 3.3 represents a basic CNN structure composed of merely five layers.

The primary functions of the aforementioned CNN example can be divided into four main aspects:

1. The input layer holds the image's pixel values.

2. Within the convolutional layer, neurons that are linked to local areas of the input perform the dot product of their weights and the region tied to the input volume. The rectified linear unit (ReLu) applies an activation function such as the sigmoid function to the output originating from the preceding layer.

3. The pooling layer downsamples along the spatial dimensionality of the input, reducing the number of parameters within the activation.

4. The fully connected layers perform standard ANN tasks, producing class scores from the activations for classification. ReLu may improve performance between these layers.

Through these transformations, CNNs can process the original input using convolutional and downsampling techniques to produce class scores for classification and regression. However, understanding the overall architecture isn't enough. Creating and optimizing these models takes time and can be complex. Next, we'll explore the individual layers, their hyperparameters, and connectivities in detail.

3.2.1.2 Convolutional layer

Convolutional layers, essential in CNNs, revolve around learnable kernels. These kernels, small in spatial size but spanning the input's depth, produce a 2D activation map when convolved across the input's spatial dimension. This process calculates the scalar product for each kernel value, letting the network learn kernels that activate upon detecting specific features, known as activations. Each kernel

generates an activation map, stacked along the depth dimension to form the layer's full output volume.

To address the challenge of large model sizes in ANNs due to fully connected neurons, each neuron in a convolutional layer only connects to a small input volume region, or the neuron's receptive field size. For instance, in an RGB image of 64 × 64 × 3, setting the receptive field size as 6 × 6 results in 108 weights per neuron in the convolutional layer, a dramatic reduction compared to standard ANNs.

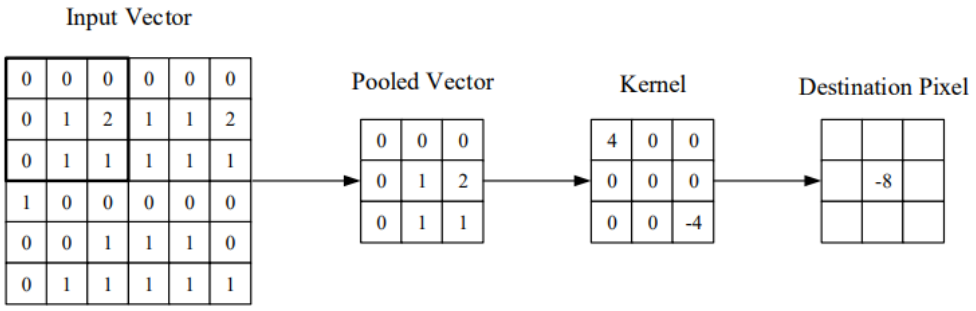

Figure 3.4 Visual representation of a convolutional layer. The kernel's central element is positioned over the input vector, which is then computed and replaced by a weighted sum of itself and the neighboring pixels.

Convolutional layers reduce model complexity through output optimization, managed via three hyperparameters: depth, stride, and zero-padding. Depth, or the output volume's dimension, can be manually set by the number of neurons within the layer. Stride determines the depth setting around the input's spatial dimensionality to position the receptive field. Zero-padding, or input border padding, helps control the output volumes' dimensionality.

Adjusting these hyperparameters alters the convolutional layer's output spatial dimensionality. Parameter sharing, a technique assuming a feature useful in one spatial region will be useful in others, further reduces parameters by constraining

each activation map within the output volume to share the same weights and bias. Consequently, during backpropagation, each output neuron represents the total gradient across the depth, updating only a single weight set.

3.2.1.3 Pooling layer

Pooling layers aim to progressively downscale the representation's dimensionality, thereby decreasing the model's parameters and computational complexity. They operate on each input's activation map, using the "MAX" function to resize it. Most CNNs employ max-pooling layers with 2 × 2 kernels and a stride of 2, reducing the activation map to 25% of its original size while keeping the depth unchanged.

Given its destructive nature, max pooling typically employs two methods: using both 2 × 2 stride and filters to cover the input's entire spatial dimension, or using overlapping pooling with a stride of 2 and kernel size of 3. A kernel size above 3 generally hampers model performance due to the destructive aspect of pooling.

Apart from max-pooling, CNNs can utilize general-pooling layers that perform multiple operations such as L1/L2-normalisation and average pooling. Nonetheless, this description primarily focuses on the concept of max-pooling.

3.2.1.4 Fully connected layer

In the fully-connected layer, neurons are directly linked to neurons in the layers immediately preceding and succeeding them, without any interconnections within those layers.

3.2.1.5 Spectral feature-based classification

Now let's move on to looking at CNN on the classification of hyperspectral images.

Hyperspectral images possess abundant spectral data and incredibly high spectral resolution. Each pixel generates one-dimensional spectral vectors consisting of spectral details. Classifying solely based on these one-dimensional spectral vectors is known as spectral information-based classification. Typically, this approach involves extracting spectral information or specific features from a pixel's spectral data through feature extraction for classification purposes. To classify hyperspectral images' spectral features, one-dimensional convolutional neural networks (1D-CNN) [19] are employed to extract spectral features and perform classification. The process is illustrated in Figure 2.3.

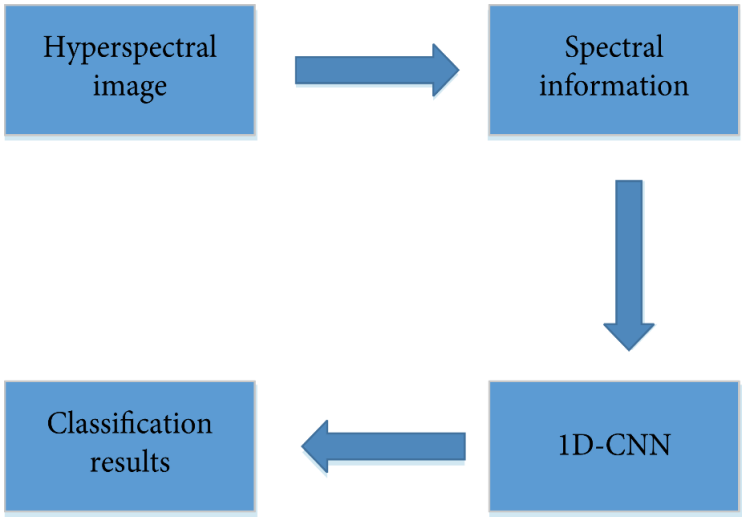

Figure 3.5 Schematic diagram of 1D-CNN.

The procedure involves feeding labeled hyperspectral data into the 1D-CNN, training the 1D-CNN using class labels, and iteratively updating the network weights using algorithms like SGD. Ultimately, the trained 1D-CNN is employed to classify each pixel, yielding classification outcomes. A one-dimensional convolution kernel

is utilized in the one-dimensional convolution operation to perform convolution on a one-dimensional feature vector. The operation is expressed as follows:

$$v_{l,j}^{x} = f\left(\sum_{m}\sum_{h=0}^{H_i-1} k_{(l-1),m}^{(x+h)} + b_{l,j}\right) \tag{3.1}$$

Among them, $k_{l,j,m}^{h}$represents the value of the l-th convolution kernel in the j-th layer at h, and the convolution kernel is connected to the m-th feature vector in the (l-1) layer network. H_i represents the length of the one-dimensional convolution kernel. $b_{l,j}$ represents the offset of the j-th feature map of the l-th layer. $v_{(l-1),m}^{(x+h)}$ represents the specific value of the m-th feature map at the $(x+h, y+w)$ position in the l-1st layer.

3.2.1.6 Spatial-feature-based classification method

This approach focuses on contextual or spatial information. In this classification process, rather than utilizing the spectral data obtained from individual pixels, the neighboring pixel's spatial details are employed. Owing to the high dimensionality of hyperspectral data, the common practice for extracting spatial information is to initially compress the dataset, followed by employing two-dimensional convolutional neural networks (2D-CNN) [20] to derive more profound spatial insights, which are then used for classification. The detailed procedure is illustrated in Figure 3.6.

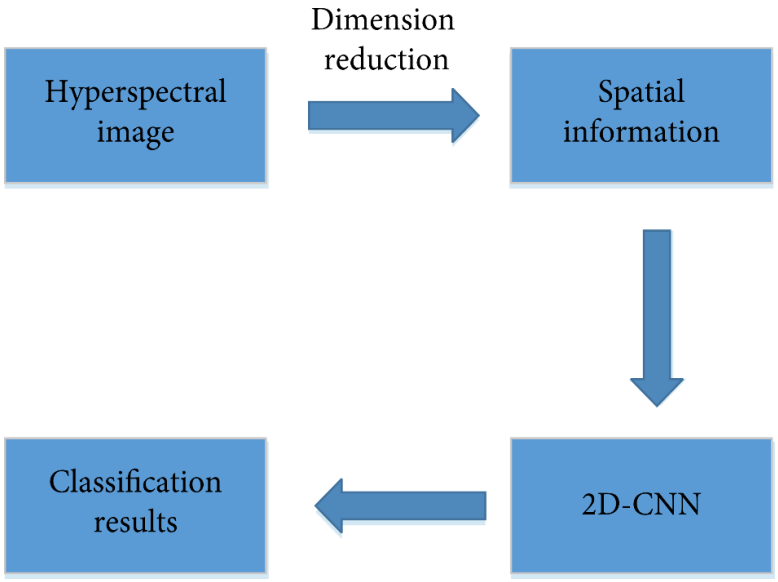

Figure 3.6 Schematic diagram of 2D-CNN.

The dimensions of the convolution layer and pooling layer are the primary distinction between the two-dimensional convolution operation and the one-dimensional convolution operation. In the case of two-dimensional convolution, a two-dimensional convolution kernel is employed to perform the convolution operation on two-dimensional data.

$$map_{l,j}^{x,y} = f\left(\sum_{m}\sum_{h=0}^{H_1-1}\sum_{w=0}^{W_l-1} k_{l,j,m}^{h,w} map_{(l-1),m}^{(x+h)(y,w)} + b_{l.j}\right) \tag{3.2}$$

Among them, $k_{l,j,m}^{h,w}$represents the value of the l-th convolution kernel in the l-th layer at (h,w), and this convolution kernel is connected to the m-th feature vector in the (l-1) layer network. H_i and W_l, respectively, represents the height and width of the convolution kernel, and $b_{l,j}$ represents the offset of the j-th feature map of the l-th layer. $map_{(l-1),m}^{(x+h),(y+w)}$ represents the specific value of the m-th feature

map at the $(x+h, y+w)$ position in the l-1st layer, and $map_{l,j}^{x,y}$ represents the output data of the j-th feature map at the l-th layer at (x, y).

3.2.1.7 Spectral-spatial feature-based classification method

Traditional hyperspectral image classification primarily relies on spectral data. Nevertheless, external environmental factors can cause identical ground features to exhibit different spectral curves, while distinct ground features may have the same spectral curve, leading to occurrences of heterospectrum within the same object and same-spectrum phenomena in foreign objects. For instance, when adjacent pixels are categorized as parking lots, those with spectral characteristics resembling metal are likely to represent cars. Similarly, if the surrounding pixels are grass, the central pixel is probably grass as well. Hyperspectral data comprises a three-dimensional structure, encompassing one-dimensional spectral and two-dimensional spatial details. A three-dimensional convolutional neural network (3D-CNN)[21] can extract both spectral and spatial information simultaneously. This specific procedure is depicted in Figure 2.4.

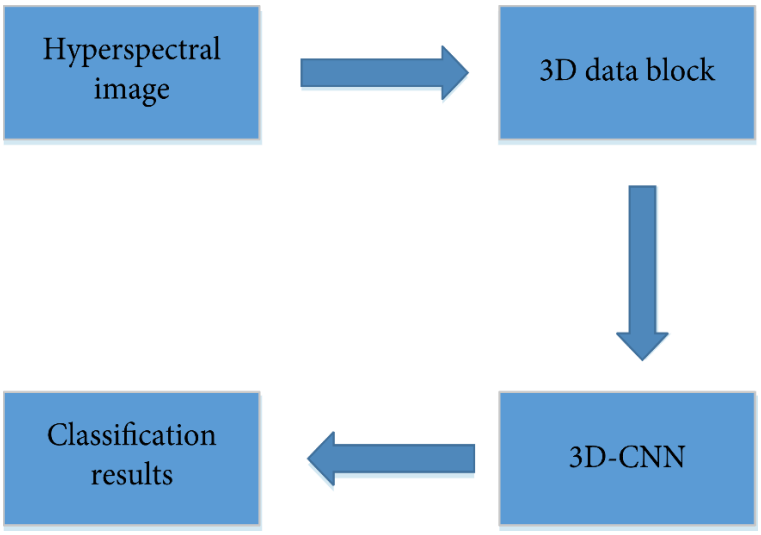

Figure 3.7 Schematic diagram of 3D-CNN.

3.2.2 Deep belief network

The implementation of a deep belief network (DBN) [22] relies on the utilization of restricted Boltzmann machines (RBMs). DBN is a network model that is built by sequentially stacking multiple RBM layers. A typical DBN consists of several RBMs and a backpropagation (BP) layer. The schematic diagram depicting its structure can be observed in Figure 3.4.

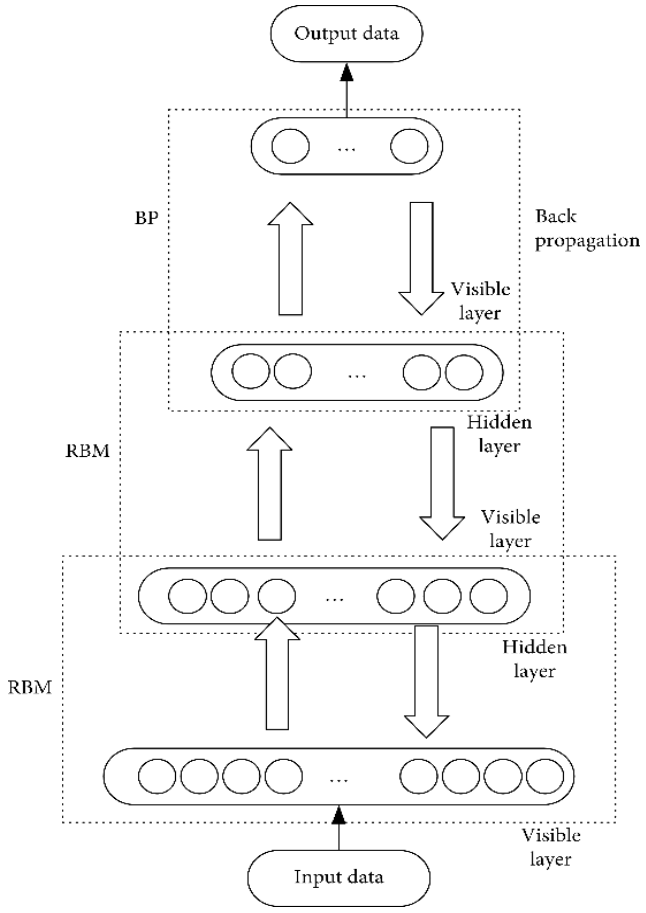

Figure 3.8 Classic DBN structure diagram.

Parameters are learned using an unsupervised approach that operates on a layer-by-layer basis during training. Initially, the data and the first hidden layer are treated as an RBM, with the parameters of this RBM being trained. Afterward, once

the parameters of the RBM are set, the first hidden layer is treated as a visible vector, while the second hidden layer is treated as a hidden vector. This process is repeated in a loop, with the following specific steps:

Train each layer of the RBM network independently and unsupervisedly, ensuring that feature vectors retain as much information as possible when mapped to different feature spaces.

Incorporate a backpropagation (BP) network at the final layer of the DBN, taking the RBM's output feature vector as input, and utilize it to supervise the training of the entity relationship classifier. Due to the limitations of each RBM layer in optimizing weights and feature vector mapping within its own layer, the backpropagation network propagates error information downwards across each RBM layer, refining both the DBN and RBM network training models. This process can be viewed as the initialization of a deep BP network's weight parameters, allowing the DBN to overcome the BP network's shortcomings of falling into local optimization and having lengthy training times due to random weight parameter initialization.

When using DBN to classify hyperspectral image spectral features, the primary approach is to employ DBN to extract deeper features from spectral information gathered from pixel locations to be classified, and then complete the classification using deep features. The classification method for hyperspectral image spatial features based on DBN is quite similar to the SAE-based method.

3.3 Unsupervised classification

The method of unsupervised classification relates to categorizing based on the spectral likeness of hyperspectral data, essentially a clustering approach that doesn't require any previous information. Given that it doesn't use any pre-existing

knowledge, unsupervised classification can merely presume initial parameters, create groups through preliminary classification procedures, and then repeatedly adjust until the related parameters fall within acceptable boundaries.

3.3.1 K-Means Classification

The fundamental concept underlying the K-means [23] clustering technique is to minimize the total sum of squared distances between each pixel within a cluster and the centroid of that particular cluster. The initial clustering process starts by randomly selecting a center point, and then other pixels are classified into one of the clusters based on set criteria, thereby completing the initial clustering. The next step involves recalculating the center point for each cluster, adjusting it, and reclassifying, repeating these steps until the clustering center points no longer shift. The optimal clustering center is then determined, yielding the best cluster results and ending the iteration process. Figure 8 illustrates the algorithm flow of K-means clustering. One limitation of K-means clustering is that the number of chosen categories remains fixed throughout the calculation, and the initially selected cluster center point position can influence the clustering outcome, leading to potentially significant variations in experimental results each time. To address this issue, auxiliary methods can be used to identify a more accurate initial clustering center, thus enhancing classification precision.

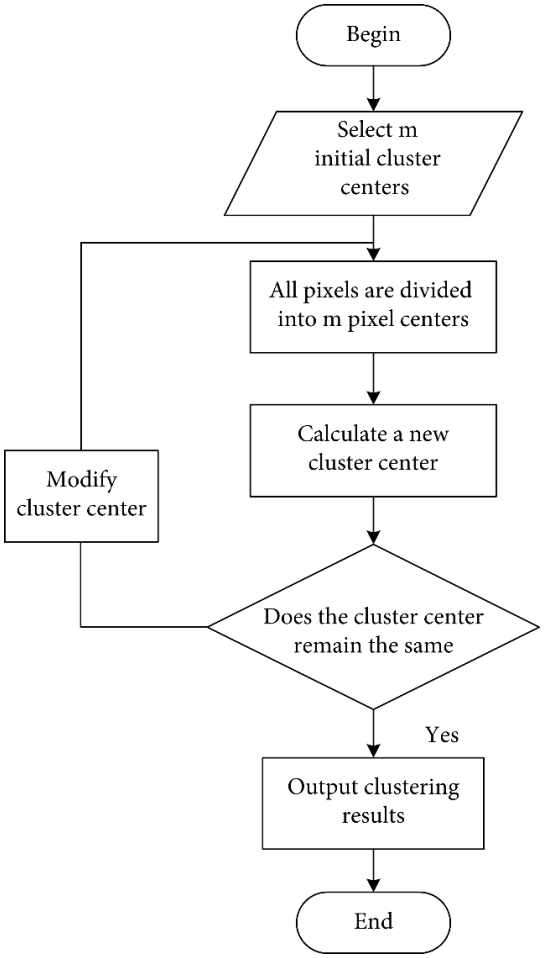

Figure 3.9 Schematic of -means algorithm.

3.3.2 Iterative Self-Organizing Method

The ISODATA [24] algorithm, like the K-means algorithm, is a frequently used clustering method. It's essentially an enhancement of the K-means classification technique. The ISODATA algorithm provides some clear advantages over K-means clustering. First, instead of continuously adjusting the cluster center during the calculation, all categories are computed and the samples are then collectively adjusted. Second, unlike K-means clustering, the ISODATA algorithm can automatically modify the number of categories during clustering based on the actual scenario, leading to more reasonable clustering results.

The primary benefits of these two classification methods include the lack of need for extensive understanding of the classification area; only sufficient knowledge is needed to interpret the classified cluster groups. This reduces the risk of human error and minimizes the initial parameters required for input. The clusters with small but distinctive spectral characteristics are more homogeneous than in supervised classification, and categories with unique and small coverage can be identified. The main drawbacks include the need for significant analysis and post-processing to achieve reliable classification outcomes. The classified clusters and land categories may or may not align due to the common phenomena of "same spectrum" and "foreign material," complicating the matching of cluster groups and categories. Moreover, as the spectral characteristics of each category vary with time and terrain, the spectral cluster groups across different images lack continuity and are challenging to compare.

3.4 Semisupervised Classification

The primary drawback of supervised methods is their reliance on the volume of training data sets with label points to determine the classification model and accuracy. Acquiring a significant amount of class labels for hyperspectral images is both time-consuming and expensive. Unsupervised methods aren't as affected by labeled samples, but their lack of prior knowledge makes the relationship between clustering categories and actual categories uncertain [19]. Semi-supervised classification addresses these limitations by utilizing a combination of labeled and unlabeled data for training the classifier. This approach is grounded in the assumption that in feature space, labeled and unlabeled samples of the same type are closer. Since numerous unlabeled samples provide a comprehensive depiction of the

data's overall characteristics, a classifier trained using both types of samples possess better generalization.

Semi-supervised classification is frequently employed in hyperspectral image classification. Notable semi-supervised classification methods encompass model generation algorithms, semi-supervised support vector machines, graph-based semi-supervised algorithms, and self-training, co-training, and tri-training.

Considering these issues, this paper presents a review of a semi-supervised classification method. Semi-supervised learning has garnered significant interest in the realm of hyperspectral image classification due to its requirement for only a minimal number of labeled samples. This learning approach merges labeled and unlabeled data to enhance classification accuracy.

3.4.1 Laplace Support Vector Machine

The Laplacian Support Vector Machine (LapSVM) [25] is an advancement of the conventional Support Vector Machine (SVM). By incorporating manifold regularization terms, LapSVM is able to leverage the geometric information derived from both labeled and unlabeled samples to construct a classifier that effectively predicts the labels of forthcoming test samples. Additionally, it is characterized by its robust adaptability and capacity for global optimization.

Given labeled samples and unlabeled samples $\{x_i\}_{i=l+1}^{l+u}, x_i \in R^m$, and $y_i \in \{-1, +1\}$, the decision function is f. The

$$L = \frac{1}{l}\sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_L \| f \|_H^2 + \gamma_M \| f \|_M^2 \tag{3.3}$$

In this context, V stands for the mis-segmentation cost function of labeled samples, while Y_L regulates the intricacy of function f within Hilbert space, and Y_M manages the complexity of the geometric features of the data distribution within the maximum distance of f. The architecture of LapSVM is elaborated further below. Initially, LapSVM employs the same loss function as the conventional SVM:

$$V(x_i, y_i, f) = max\{0,1 - y_i f(x_i)\}. \tag{3.4}$$

Among them, f represents the classification decision function $f(x) = \langle w, \varphi(x)\rangle + b$ of the selected classifier, where $\varphi(\cdot)$ denotes a non-linear mapping function that transforms data from a low-dimensional space to a high-dimensional Hilbert space, where

$$w = \sum_{i=1}^{l+N} \alpha_i \varphi(x_i) = \Phi\alpha, \Phi = [\varphi(x_1), \cdots, \varphi(x_{l+u})]^T, \tag{3.5}$$

$\alpha = [\alpha_1, \cdots, \alpha_{l+u}]$, is a decision function after finishing:

$$f(x) = \sum_{i=l}^{l+u} \alpha_i K(x_i, x) + b. \tag{3.6}$$

The kernel function K represents different learner functions, which can be achieved by choosing various kernel functions, so there are

$$\| f \|_H^2 = \| w \|^2 = (\Phi\alpha)^T(\Phi\alpha) = \alpha^T K\alpha. \tag{3.7}$$

The LapSVM algorithm emulates the geometric arrangement of data by creating a graph based on both labeled and unlabeled samples. By applying the smoothing assumption to normalize the graph, the penalty classification function undergoes adjustments, particularly in its rapidly changing segment.

$$\| f \|_H^2 = \frac{1}{(l+u)^2} \sum_{i,j=1}^{l+u} W_{ij} \left(f(x_i) - f(x_j) \right)^2 = f^T L f \tag{3.8}$$

Substituting the above formula into

$$\begin{aligned} &\min_{\xi_i \in R^1, \alpha \in R^{1+M}} \left\{ \frac{1}{l} \sum_{i=l}^{l} \xi_i + \gamma_L \alpha^T K \alpha + \frac{\gamma_M}{(l+u)^2} \alpha^T K L K \alpha \right\} \\ &\text{s.t. } y_i \left(\sum_{i,j=1}^{l+u} \alpha_i K(x_i, x_j) + b \right) \geq 1 - \xi_i \\ &\xi_i \geq 0, i = 1, \cdots, l, \end{aligned} \tag{3.9}$$

where ξ_i represents the relaxation factor of the labeled sample.

The LapSVM algorithm effectively incorporates the influence of unlabeled samples in the classification process by considering the geometric attributes of the data. However, it often necessitates significant computational resources due to its high computational cost.

3.4.2 Self-Training

Self-training [26] is a frequently employed semi-supervised classification algorithm. In executing this algorithm, a classifier is initially trained with labeled samples, followed by the labeling of a plethora of unlabeled samples using this classifier. High-confidence data is chosen from these labeled samples and added, along with their labels, to the initial training set for retraining the classifier. This

process is repeated until a termination condition is met. The general progression of self-training is as follows:

(1) Train the classifier using the initial set of labeled samples

(2) Apply the classifier to label the data within the set of unlabeled samples, select the samples with the greatest confidence, and record them

(3) Retrain the classifier with the newly acquired sample set

(4) Repeat steps 2) and 3) until the termination condition is satisfied

Self-training algorithms are extensively utilized. Although this classification approach is simple and convenient, it becomes challenging to train a classifier with strong generalization capabilities and high accuracy due to the initially limited number of training samples. Additionally, when unlabeled samples are labeled, a significant number of mislabeled samples may be generated. These samples act as noise samples when added to the original training set, and as the iteration proceeds, errors accumulate, invariably leading to a degradation in the classifier's classification performance.

3.5 Evaluation measures

Within the domain of hyperspectral image categorization, three key accuracy measures, namely, OA, AA, and Kappa coefficient, are typically employed for impartial assessment. Herein, we provide a detailed explanation of these three accuracy assessment indicators.

OA represents the ratio of correctly categorized instances to the total count of test instances. The computation is detailed below:

$$\mathrm{OA} = \sum_{i=1}^{C} \mathbf{M}_{ii}/N \tag{3.10}$$

C denotes the total count of categories. The confusion matrix, M, is derived by juxtaposing the classification map against the actual results. M_{ii} signifies the count of instances that are part of class i and are also classified as such. N stands for the aggregate count of instances in the test set.

AA symbolizes the average proportion of correctly identified pixels per class, as defined below:

$$\text{AA} = \left(\sum_{i=1}^{C} \left(\mathbf{M}_{ii} / \sum_{i=1}^{C} \mathbf{M}_{ij} \right) \right) / C \tag{3.11}$$

The Kappa coefficient signifies the proportion of agreement adjusted by the count of concurrences that could randomly occur, coupled with the accuracy specific to each class.

$$\text{Kappa} = \left(N \left(\sum_{i=1}^{C} \mathbf{M}_{ii} \right) - \sum_{i=1}^{C} \left(\sum_{j=1}^{C} \mathbf{M}_{ij} \sum_{j=1}^{C} \mathbf{M}_{ji} \right) \right) / \left(N^2 - \sum_{i=1}^{C} \left(\sum_{j=1}^{C} \mathbf{M}_{ij} \sum_{j=1}^{C} \mathbf{M}_{ji} \right) \right). \tag{3.11}$$

The Kappa coefficient holds the benefit of considering the impact of uncertainty on the classification outcomes when determining accuracy. The above-mentioned accuracy measures are all computed through the juxtaposition of classification maps and actual results. Hence, it can be easily deduced that the actual results will affect the precision of the measurements obtained.

3.6 Conclusion

The categorization and identification of hyperspectral images constitute a crucial aspect of hyperspectral image processing. This paper has examined several techniques for hyperspectral image classification, encompassing supervised, unsupervised, and semi-supervised classification. While the supervised and unsupervised methods presented in this discussion each offer varying degrees of benefits, there are inherent constraints when implementing these methods. For instance, supervised classification necessitates specific preconditions, and human influences can notably affect the outcomes of the classification. Hence, depending on the particular application requirements and considering the vast information obtained through hyperspectral images, a combination of multiple methods is required to achieve the desired classification results. As hyperspectral image technology continues to evolve, its classification has found widespread application. However, existing theories and techniques still encounter certain limitations when dealing with more complex hyperspectral image classifications. Therefore, in the future, it will be crucial to focus on researching and developing more specialized methods for hyperspectral image classification.

4 SOFTWARE IMPLEMENTATION OF OBJECT CLASSIFICATION ON HYPERSPECTRAL IMAGES

Convolutional Neural Networks have recently gained a lot of popularity thanks to their dramatic performance improvement over manually created features. In many applications where processing of visual information is necessary, such as image classification, object identification, semantic segmentation, colon cancer classification, depth estimation, face anti-spoofing, etc., the CNN has demonstrated highly promising performance. Deep learning for hyperspectral image analysis has made significant advancements in recent years as well. For the HSI classification, a dual-path network (DPN) is proposed by fusing the residual network and dense convolutional network. To represent the remote sensing images in unsupervised training, Yu et al. developed a greedy layer-wise technique. A pixel-block pair (PBP) based data augmentation strategy was presented by Li et al. to extend deep learning for HSI classification. Deep feature fusion network was proposed by Song et al. while Cheng et al. employed pre-built CNN models for HSI classification. In essence, they retrieved the deep spatial features in a hierarchical fashion and utilized SVM for training and classification.

The literature makes it clear that utilizing only 2D-CNN or 3D-CNN had several drawbacks, such as lacking channel relationship information or requiring very complex models, respectively. Additionally, it hindered these techniques from improving their accuracy when used with hyperspectral pictures. The primary cause is that hyperspectral images are volumetric data with a second spectral dimension. The spectral dimensions cannot be effectively extracted into appropriate discriminating feature maps by the 2D-CNN alone. A deep 3D-CNN is similarly more computationally intensive and appears to perform worse on its own for classes

with similar textures over numerous spectral bands. This is what inspired me to suggest a **Dual Convolution HIS Net (**DualConvHSINet**)** model that corrects these earlier models' flaws. For the proposed model, the 3D-CNN and 2D-CNN layers are put together in a way that fully utilizes both the spectral and spatial feature maps to reach the highest level of accuracy.

4.1 Proposed DualConvHSINet model

Let $\mathbf{I} \in \mathcal{R}^{M \times N \times D}$ be the symbol for the spectral-spatial hyperspectral data cube, where **I** stands for the initial input, M for the width, N for the height, and D for the quantity of spectral bands/depth. Each HSI pixel in **I** comprises D spectral measurements, which together create the one-hot label vector $Y = (y_1, y_2, \ldots y_C) \in \mathcal{R}^{1 \times 1 \times C}$, where C stands for the various types of land cover.

The mixed land-cover classes in the hyperspectral pixels, however, introduce considerable intra-class variability and inter-class similarity into **I**. Any model must overcome a huge challenge to solve this issue. The original HSI data (**I**) along spectral bands are initially subjected to the conventional principal component analysis (PCA) to reduce the spectral redundancy. The PCA keeps the same spatial dimensions (i.e., width M and height N) while reducing the number of spectral bands from D to B.

The spectral bands have been selectively minimized to maintain the crucial spatial information required for object recognition. The data cube, which has undergone PCA reduction, can be represented as $\mathbf{X} \in \mathcal{R}^{M \times N \times B}$. In this representation, X is the adjusted input following the PCA process, M signifies the width, N stands for the height, and B represents the count of spectral bands post-PCA.

The Hyper Spectral Imaging (HSI) data cube is segmented into minute, intersecting 3D sections, the authentic labels of these are decided by the middle pixel's label to apply image categorization methods. We have constructed 3D adjacent patches $P \in \mathcal{R}^{S\times S\times B}$ from $\mathbf{X}$, situated at the spatial point (α,β), encapsulating the $S \times S$ window or spatial range and all B spectral bands. The aggregate quantity of created 3D patches (n) from X is given by $(M-S+1)\times(N-S+1)$. Hence, the 3D patch situated at position (α,β), denoted by $P_{\alpha,\beta}$, covers the width from $\alpha-(S-1)/2$ to $\alpha+(S-1)/2$, height from $\beta-(S-1)/2$ to $\beta+(S-1)/2$ and includes all B spectral bands of the Principal Component Analysis (PCA) condensed data cube X.

In 2D Convolutional Neural Networks (2D-CNN), the incoming data are processed with 2D kernel functions. This convolution operation involves calculating the aggregate of the dot product between the input data and the kernel. The kernel slides across the input data to encompass its complete spatial dimensions. The output from this convolution, also known as convolved features, are fed into an activation function to incorporate nonlinearity into the model. In 2D convolution, the activation value at spatial coordinate (x,y) in the j^{th} feature map of the i^{th} layer, denoted as $v_{i,j}^{x,y}$, is computed based on the subsequent equation,

$$v_{i,j}^{x,y}=\phi\left(b_{i,j}+\sum_{\tau=1}^{d_{l-1}}\sum_{\rho=-\gamma}^{\gamma}\sum_{\sigma=-\delta}^{\delta}w_{i,j,\tau}^{\sigma,\rho}\times v_{i-1,\tau}^{x+\sigma,y+\rho}\right) \tag{4.1}$$

where ϕ is the activation function, $b_{i,j}$ signifies the bias parameter for the j^{th} feature map of the i^{th} layer, d_{l-1} is the number of feature map in $(l-1)^{th}$ layer and the depth of kernel $w_{i,j}$ for the j^{th} feature map of the i^{th} layer, $2\gamma+1$ is the width

of kernel, $2\delta + 1$ is the height of kernel, and $w_{i,j}$ is the value of weight parameter for the j^{th} feature map of the i^{th} layer.

The process of 3D convolution involves convolving a 3D kernel with 3D data. In the suggested model tailored for Hyper Spectral Imaging (HSI) data, the convolution layer's feature maps are created by applying a 3D kernel across several adjacent bands in the input layer, thereby encompassing the spectral data. During 3D convolution, the activation value located at the spatial coordinates (x, y, z) in the j^{th} feature map of the i^{th} layer, denoted as $v_{i,j}^{x,y,z}$, is generated as follows,

$$v_{i,j}^{x,y,z} = \phi\left(b_{i,j} + \sum_{\tau=1}^{d_{l-1}} \sum_{\lambda=-\eta}^{\eta} \sum_{\rho=-\gamma}^{\gamma} \sum_{\sigma=-\delta}^{\delta} w_{i,j,\tau}^{\sigma,\rho,\lambda} \times v_{i-1,\tau}^{x+\sigma,y+\rho,z+\lambda}\right) \tag{4.2}$$

where $2\eta + 1$ is the depth of kernel along spectral dimension and other parameters are the same as in (Eq. 4.1).

CNN parameters, including the bias b and the kernel weight w, are commonly trained using supervised methods with the aid of gradient descent optimization techniques. Traditional 2D CNNs perform convolutions exclusively across the spatial dimensions, incorporating all the feature maps of the preceding layer to derive the 2D discriminative feature maps. However, when it comes to HSI classification, it's crucial to capture not only spatial information but also spectral data, which is distributed across multiple bands. This is something that 2D-CNNs fall short in managing. On the other hand, a 3D-CNN kernel can simultaneously extract both spectral and spatial features from HSI data, but this comes with the drawback of elevated computational complexity. To leverage the automatic feature learning strengths of both 2D and 3D CNN, we introduce a mixed feature learning framework dubbed DualConvHSINet for HSI classification. The flow diagram of the proposed DualConvHSINet network is shown in Figure 4.1. It includes three 3D

convolutions (Eq. 4.2), a single 2D convolution (Eq. 4.1), and three fully connected layers.

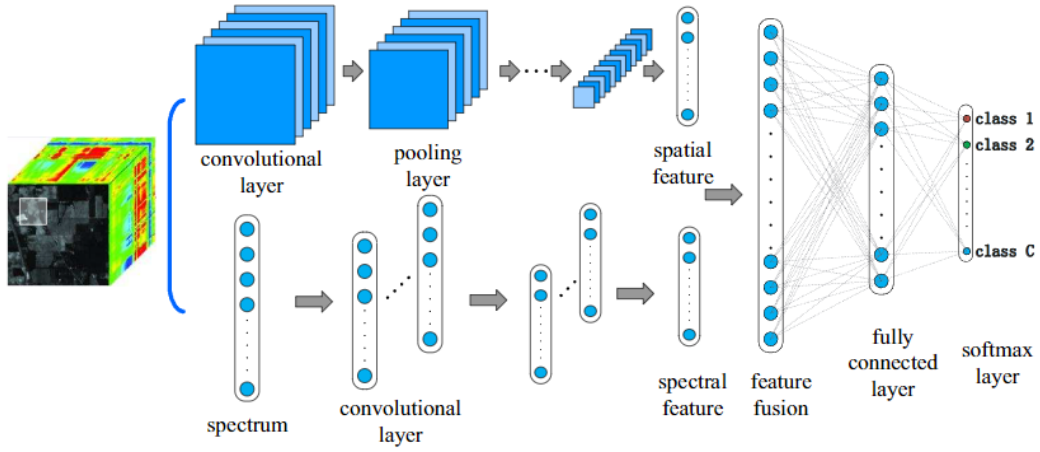

Figure 4.1 The DualConvHSINet model is suggested, which combines 3D and 2D convolution methods for the classification of hyperspectral images (HSI).

The software implementation for hyperspectral image classification is primarily done in Python and leverages a number of machine learning and data processing libraries. Key among them is TensorFlow, a powerful machine learning library used for creating the Convolutional Neural Network model. Additionally, the Scikit-learn library is used for Principal Component Analysis (PCA), an operation essential for reducing the spectral redundancy in the initial hyperspectral data cube. Other utility libraries like Matplotlib and NumPy are also employed for data visualization and manipulation, respectively.

To summarize, the key functions in the program include:

1. load_data(name): This function is implemented to load the initial hyperspectral data (I) and labels using the Scipy library. This process corresponds to the initial theoretical definitions of I and Y.

2. apply_pca(X, numComponents=75): This function implements PCA via Scikit-learn to reduce the spectral redundancy of the initial hyperspectral data (I), in

accordance with the theoretical concept. It transforms the initial data cube I to X with reduced spectral bands.

3. pad_with_zeros(X, margin=2): This function is used to pad the adjusted input X with zeros, a step which allows the formation of 3D adjacent patches that have their boundaries outside the actual spatial dimensions of X.

4. generate_image_cubes(X, y, windowSize=5, removeZeroLabels = True): This function generates the 3D adjacent patches P for image categorization, based on the definitions given in the theoretical part.

5. build_model(input_shape, output_units): This function constructs the DualConvHSINet model, incorporating both 2D and 3D convolutions along with fully connected layers. It employs the TensorFlow library to create the layers, which helps extract both spatial and spectral features from HSI data.

The last part of the code integrates the above functions and follows the process of loading the data, applying PCA, generating image cubes, building the model, and then training it using the TensorFlow library. It further saves the model in format h5 and plots the loss and accuracy curves using Matplotlib.

The developed model complies with the theoretical framework, by successfully addressing the issue of high intra-class variability and inter-class similarity in hyperspectral pixels and effectively extracting both spatial and spectral information for HSI classification.

The full Python implementation of the hyperspectral images classification model training is provided in Appendix A.

4.2 Dataset description and training details

We utilized three hyperspectral image datasets that are openly accessible: University of Pavia, Indian Pines, and Salinas Scene. The Indian Pines (IP) dataset

includes images with spatial dimensions of 145 x 145 and 224 spectral bands spanning from 400 to 2500 nm wavelengths, but we excluded 24 spectral bands that overlap with water absorption regions. This dataset is categorized into 16 different vegetation classes according to the available ground truth. The University of Pavia (UP) dataset comprises images with spatial dimensions of 610x340 pixels and 103 spectral bands ranging from 430 to 860 nm in wavelength. The ground truth here is partitioned into 9 urban land-cover categories. Lastly, the Salinas Scene (SA) dataset consists of images with spatial dimensions of 512x217 and 224 spectral bands covering the wavelength range of 360 to 2500 nm. We removed 20 spectral bands that were absorbing water. This dataset has a total of 16 different classes. The network was trained using mini-batches, each consisting of 256 examples, and the training process was repeated for a total of 100 epochs. This was done without the use of batch normalization or data augmentation techniques.

All experimental work is performed with the help of Colab Research, using a computing environment with an A100 GPU and 24 GB RAM. It was identified the optimal learning rate to be 0.001, as determined by the classification results. To ensure a balanced comparison, we have maintained consistent spatial dimensions in 3D-patches of input volume across various datasets, with dimensions being 25x25x30 for IP, and 25x25x15 for both UP and SA, respectively.

Let's delve into the implications of executing the code found in Appendix A, specifically with reference to the Pavia University dataset. This is an image captured by the ROSIS sensor during a flight campaign over Pavia, located in northern Italy. The image from Pavia University comprises 103 spectral bands and measures 610*340 pixels. However, certain samples in these images lack valuable information and need to be excluded prior to analysis. The geometric resolution of the image stands at 1.3 meters. Each image's ground truth distinguishes 9 unique classes. The figures illustrate the omitted samples as broad black strips.

Initially, the console will yield an output akin to that depicted in Figure 4.1, providing an in-depth description and analysis of the model parameters.

This output is a summary of a convolutional neural network (CNN) architecture specifically designed for the classification of hyperspectral images.

Let's delve into the description:

- InputLayer receives the hyperspectral images, which have a dimensionality of 25x25 spatial pixels, 15 spectral bands (or channels), and 1 to indicate grayscale (if images were colored, it would typically be 3). The choice of 25x25 based on empirical results suggesting that patches of this size contain enough spatial context to make accurate predictions while still being small enough to be computationally manageable. In other words, these patches provide a balance between computational efficiency and model performance. The patches are extracted from the entire hyperspectral image and used to train the model. This is often done in order to manage the high dimensionality of hyperspectral images and to generate more training examples. In this case, the model is designed to work with images that have 15 spectral bands.

- Conv3D layers apply convolution operation in 3D, spatially and spectrally. They extract features from the input data and reduce their dimensions. There are 3 Conv3D layers in the model with an increasing number of filters (8, 16, and 32) used to capture more complex patterns as the data progresses through the network. The kernel size used by these convolution operations is implicitly set to (3,3,3), since the output dimensions reduce by 2 at each step.

- Reshape layer converts the 3D output of the last Conv3D layer into 2D. It combines the last two dimensions, reducing it from (19,19,3,32) to (19,19,96).

- Conv2D is a convolution layer that operates in 2 dimensions (height and width). It is used here for further spatial feature extraction from the reshaped data.

- Flatten layer is used to flatten the output of the Conv2D layer into a single dimension vector, which can be inputted into Dense layers.

- Dense layers, also called fully connected layers, perform classification on the features extracted by the convolutional layers. The model uses two Dense layers with 256 and 128 neurons respectively, followed by dropout layers to prevent overfitting.

- The final Dense layer with 9 neurons is the output layer, corresponding to the 9 classes that the model is expected to classify. This would indicate that there are 9 different classes in the hyperspectral image dataset.

- The dropout layers are used for regularization and reducing overfitting. During training, they randomly set a fraction of input units to 0 at each update, which helps prevent overfitting.

This network has a total of 4,844,793 trainable parameters, meaning that these weights and biases are updated during training. There are no non-trainable parameters in this network, which would otherwise be kept constant during training.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 25, 25, 15, 1)]   0

 conv3d (Conv3D)             (None, 23, 23, 9, 8)      512

 conv3d_1 (Conv3D)           (None, 21, 21, 5, 16)     5776

 conv3d_2 (Conv3D)           (None, 19, 19, 3, 32)     13856

 reshape (Reshape)           (None, 19, 19, 96)        0

 conv2d (Conv2D)             (None, 17, 17, 64)        55360

 flatten (Flatten)           (None, 18496)             0

 dense (Dense)               (None, 256)               4735232

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 9)                 1161

=================================================================
Total params: 4,844,793
Trainable params: 4,844,793
Non-trainable params: 0
```

Figure 4.1 Results of model parameters for PU dataset

After carrying out the training, we can observe the accuracy and loss convergence over 100 epochs for both training and validation sets, as depicted in Fig. 4.2 for the suggested approach. Notably, convergence is reached roughly around the 50th epoch, indicating the method's swift convergence rate.

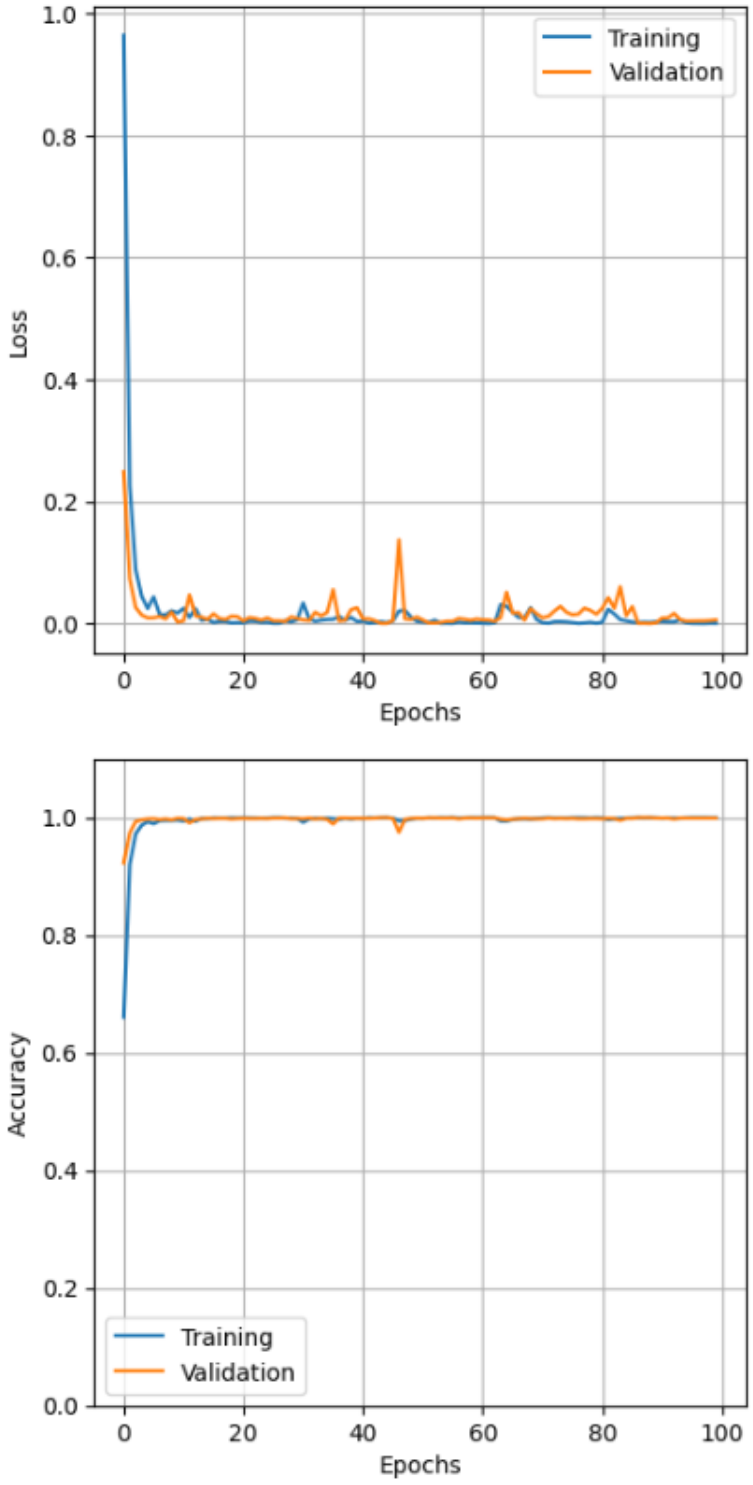

Figure 4.2 The convergence of accuracy and loss across epochs on the Indian Pines dataset.

4.3 Classification results

Full implementation is places in Appendix B. The general approach there is to prepare the data, load the model, make predictions, evaluate performance, and visualize the results.

As we already remember hyperspectral imaging is an image consisting of many spectral bands, each reflecting the intensity of light of a particular wavelength. These bands represent a wide range of the electromagnetic spectrum, often beyond the limits of visible light. Our ROSIS-03 hyperspectral sensor covers the spectrum from 430 to 860 nm, dividing it into 103 spectral bands. The width of each band is approximately 4.174 nm. The image on Figure 4.3 is based on data from three spectral bands: 54, 33, and 14.

The 54th band is the red region of the spectrum corresponding to the wavelength range of approximately 625 - 740 nm.

The 33rd band is the green region of the spectrum corresponding to a wavelength range of approximately 520 to 570 nm.

The 14th band is the blue region of the spectrum, corresponding to a wavelength range of approximately 440 - 490 nm.

Thus, the result is an image where each pixel is displayed with a color based on the light intensity of these three wavelengths (red, green and blue). This allows spectral data that would otherwise be invisible to the eye to be visualized and analyzed.

Figure 4.3 Visual display of hyperspectral images that would otherwise be invisible to the eye

Now let's move on to classifying our image. In general, the algorithm works as follows:

1. First, a two-dimensional array is created which will be used to store the prediction results for each pixel in the image.

2. The algorithm then looks at each pixel in the image in turn.

3. If a pixel does not belong to the classes of interest (usually designated as class 0), it is skipped.

4. For each pixel of interest, the algorithm extracts the corresponding portion of the image. This portion of the image is the area around the pixel, and its dimensions are determined in advance.

5. This section of the image is then fed to the input of the deep learning model, which performs prediction, predicting which class the section belongs to.

6. Since the model produces a probability distribution for all classes, we select the class with the highest probability as the predicted class.

This process is repeated for each pixel in the image.

The result is a two-dimensional classification map for the entire image as on Figure 4.4. This map shows which class each pixel in the image belongs to, according to our deep learning model.

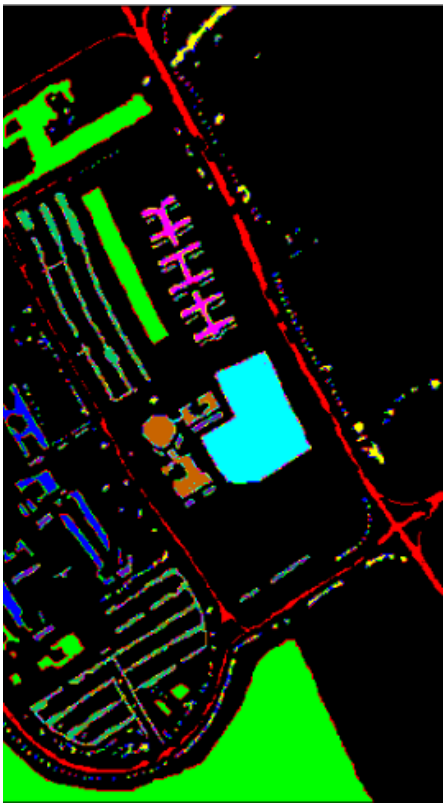

Figure 4.4 Predicted classification map for Pavia University dataset

Figure 4.5 Legend for predicted classification map for Pavia University dataset

Figure 4.6 Visual display of the predicted classification map from the original image

4.4 Performance evaluation

In this correspondence, we've applied three different evaluation metrics - Overall Accuracy (OA), Average Accuracy (AA), and the Kappa Coefficient (Kappa) - to assess the performance of Hyperspectral Image (HSI) classification. OA gives us the ratio of accurately classified samples to the total sample count, while AA is the mean accuracy calculated across various classes. Kappa, on the other hand, is a statistical measure offering mutual insights into the high-level correspondence between the real-world and classified maps. The outcomes from the DualConvHSINet model that we propose are juxtaposed with prevalent supervised techniques like SVM, 2D-CNN, 3D-CNN, M3D-CNN, and SSRN. The dataset is divided arbitrarily into training (30%) and testing (70%) segments. The computations of results were performed using the publicly accessible code corresponding to the methods being compared.

As shown in Table 4.1, the OA, AA, and Kappa coefficient results for various methods4 are presented. DualConvHSINet, as indicated by Table 4.1, surpasses all other comparative methods across each dataset, all while maintaining the lowest standard deviation. The design of DualConvHSINet is predicated on the layered depiction of a spectral-spatial 3D CNN, succeeded by a spatial 2D CNN. These two are mutually beneficial. An observation from these findings shows that the 3D-CNN underperforms compared to the 2D-CNN on the Salinas Scene dataset. To our understanding, this may be due to the existence of two classes in the Salinas dataset (specifically Grapes-untrained and Vinyarduntrained) which predominantly have similar textures across the majority of spectral bands. As such, with the heightened redundancy across the spectral bands, the 2D-CNN outdoes the 3D-CNN on the Salinas Scene dataset. In addition, the performance of both SSRN and DualConvHSINet consistently outmatches that of M3D-CNN. The implication is

clear that solo 3D or 2D convolution cannot provide the same level of discriminative feature representation as a hybrid of 3D and 2D convolutions.

Figure 4.7 depicts a classification map of a sample hyperspectral image, created using SVM, 2D-CNN, 3D-CNN, M3D-CNN, SSRN, and DualConvHSINet methods. The classification map quality for SSRN and DualConvHSINet noticeably exceeds that of the other techniques. Among SSRN and DualConvHSINet, the maps created by DualConvHSINet in smaller sections are superior to those by SSRN. The computational efficiency of the DualConvHSINet model is evident in the training and testing durations outlined in Table 4.2, demonstrating its increased efficiency over the 3D-CNN model. Table 4.3 reflects the impact of spatial dimension on the performance of the DualConvHSINet model, revealing that a 25 × 25 spatial dimension is most fitting for the proposed method. We further conducted experiments with even less training data, specifically only 10% of total samples, and encapsulated the results in Table 4.4. It is notable from this experiment that each model's performance dips slightly, yet the proposed method continues to surpass the other techniques in nearly all instances.

4.5 Conclusion

This correspondence presents a hybrid 3D and 2D model intended for hyperspectral image categorization. The suggested DualConvHSINet model essentially merges the mutually beneficial data of spatio-spectral and spectral elements via 3D and 2D convolutions, respectively. Benchmark tests across three datasets, contrasted with recent advanced methods, substantiate the proposed method's superior effectiveness. Not only is the proposed model more computationally efficient than the 3D-CNN model, but it also demonstrates outstanding performance when working with limited training data.

CONCLUSIONS

Hyperspectral images from remote sensing offer distinct advantages over traditional multispectral images. This includes providing more accurate and detailed analyses of satellite data, which is significant for various applications like geological, agricultural, environmental, and military purposes.

There is a wide array of techniques currently being used for object classification in hyperspectral images, all of which have proven to be precise in delivering high-resolution data.

The work introduces a novel software approach for object classification on hyperspectral images, utilizing advanced machine learning algorithms. This represents a significant contribution in the field as it demonstrates the potential for enhancing current remote sensing capabilities.

The implications of this study are broad and promising, potentially leading to significant advancements in remote sensing, object classification, and hyperspectral image analysis. These advancements could lead to improved accuracy and efficiency in the mentioned applications, supporting progress in several critical areas.

REFERENCE

1. Lillesand, T., Kiefer, R. W., & Chipman, J. (2007). Remote Sensing and Image Interpretation. John Wiley & Sons.

2. Richards, J. A., & Jia, X. (2006). Remote sensing digital image analysis: an introduction. Springer.

3. Remote sensing of the Earth: Its future and role in understanding the Earth system by Andrew K. Skidmore, Piers J. Sellers, and Richard A. Myneni. International Journal of Remote Sensing, Vol. 39, Issue 23, 2018.

4. Satellite remote sensing for water erosion assessment: A review by N. Baghdadi, M. Bernier, R. Gauthier, and I. Neeson. Catena, Vol. 64, Issue 2, 2005.

5. Pohl, C., Van Genderen, J.: Remote sensing image fusion: A practical guide. CRC Press (2016).

6. Ferraris, V., Dobigeon, N., Wei, Q., Chabert, M.: Detecting changes between optical imagesof different spatial and spectral resolutions: A fusion-based approach. IEEE Trans. on Geo-science and Remote Sensing 56(3), 1566–1578 (2018)

7. Goetz, A.F.H., Three decades of hyperspectral remote sensing of the Earth: A personal view. Remote Sensing of Environment, 2010. 113: p. S5-S16.

8. Feng X, He L, Cheng Q, Long X, Yuan Y. Hyperspectral and Multispectral Remote Sensing Image Fusion Based on Endmember Spatial Information. Remote Sensing. 2020; 12(6):1009.

9. B. Lu, Y. He and P. D. Dao, “Comparing the Performance of Multispectral and Hyperspectral Images for Estimating Vegetation Properties,” in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 12, no. 6, pp. 1784–1797, June 2019

10. Sluiter, R., and E. J. Pebesma. “Comparing techniques for vegetation classification using multi-and hyperspectral images and ancillary environmental data.”

11. Y. Ma, R. Li, G. Yang, L. Sun, and J. Wang, “A research on the combination strategies of multiple features for hyperspectral remote sensing image classification,” Journal of Sensors, vol. 2018, Article ID 7341973, 14 pages, 2018.

12. H. Binol, “Ensemble learning based multiple kernel principal component analysis for dimensionality reduction and classification of hyperspectral imagery,” Mathematical Problems in Engineering, vol. 2018, Article ID 9632569, 14 pages, 2018.

13. S. Li, W. Song, L. Fang, Y. Chen, P. Ghamisi, and J. A. Benediktsson, “Deep learning for hyperspectral image classification: an overview,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 9, pp. 6690–6709, 2019.

14. B. E. Boser, I. M. Guyon, and V. N. Vapnik, “A training algorithm for optimal margin classifiers,” in Proceedings of the Fifth Annual Workshop on Computational Learning Theory, pp. 144–152, ACM Press, 1992.

15. C. Li, P. Hsieh, and B. Kuo, “Multiple SVMS based on random subspaces from kernel feature importance for hyperspectral image classification,” in 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), pp. 574–577, Fort Worth, TX, USA, 2017.

16. H. Kwon, X. Hu, J. Theiler, A. Zare, and P. Gurram, “Algorithms for multispectral and hyperspectral image analysis,” *Journal of Electrical and Computer Engineering*, vol. 2013, Article ID 908906, 2 pages, 2013.

17. Y. Lei, “Fusion method of PCA and BP neural network for face recognition,” in 2011 International Conference on Computer Science and Service System (CSSS), pp. 3256–3259, Nanjing, China, 2011.

18. K. Makantasis, K. Karantzalos, A. Doulamis, and N. Doulamis, “Deep supervised learning for hyperspectral data classification through convolutional neural networks,” in 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), pp. 4959–4962, Milan, Italy, 2015.

19. Z. Cheng and F. Xie, “Semi-supervised classification of hyperspectral images based on spatial features and texture information,” Bulletin of Surveying and Mapping, vol. 51, no. 12, pp. 56–59, 2016.

20. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, 2012, pp. 1097–1105.

21. X. Kang, B. Zhuo, and P. Duan, “Dual-path network-based hyperspectral image classification,” IEEE Geoscience and Remote Sensing Letters, 2018.

22. Yu, Z. Gong, C. Wang, and P. Zhong, “An unsupervised convolutional feature fusion network for deep representation of remote sensing images,” IEEE Geoscience and Remote Sensing Letters, vol. 15, no. 1, pp. 23–27, 2018.

23. W. Li, C. Chen, M. Zhang, H. Li, and Q. Du, “Data augmentation for hyperspectral image classification with deep cnn,” IEEE Geoscience and Remote Sensing Letters, 2018.

24. W. Song, S. Li, L. Fang, and T. Lu, “Hyperspectral image classification with deep feature fusion network,” IEEE Transactions on Geoscience and Remote Sensing, vol. 56, no. 6, pp. 3173–3184, 2018.

25. G. Cheng, Z. Li, J. Han, X. Yao, and L. Guo, “Exploring hierarchical convolutional features for hyperspectral image classification,” IEEE Transactions on Geoscience and Remote Sensing, no. 99, pp. 1–11, 2018.

26. 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, 2013.

APPENDIX A

Software implementation of DualConvHSINet model training for object classification

```
import os
import numpy as np
import scipy.io as sio
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import Input, Conv3D, Conv2D,
Dense, Flatten, Reshape, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from collections import Counter
from sklearn.model_selection import StratifiedShuffleSplit

## GLOBAL VARIABLES
DATASET = 'PU'
TEST_RATIO = 0.7
WINDOW_SIZE = 25
DATA_PATH = '/content/dataset/'
MODEL_PATH =
"/content/drive/MyDrive/trained_models/{}_hybrid_sn.h5".for
mat(DATASET)

def load_data(name):
    """
    Load data and labels for a given dataset name.

    Parameters:
    name (str): Name of the dataset to load.
                Acceptable inputs: 'IP', 'SA', 'SA_S',
'PU'.

    Returns:
    data (ndarray): Multidimensional array containing the
loaded data.
    labels (ndarray): Multidimensional array containing the
corresponding labels.
```

```
    """
    if name == 'IP':
        data = sio.loadmat(os.path.join(DATA_PATH,
'Indian_pines_corrected.mat'))['indian_pines_corrected']
        labels = sio.loadmat(os.path.join(DATA_PATH,
'Indian_pines_gt.mat'))['indian_pines_gt']
    elif name == 'SA':
        data = sio.loadmat(os.path.join(DATA_PATH,
'Salinas_corrected.mat'))['salinas_corrected']
        labels = sio.loadmat(os.path.join(DATA_PATH,
'Salinas_gt.mat'))['salinas_gt']
    elif name == 'SA_S':
        data = sio.loadmat(os.path.join(DATA_PATH,
'SalinasA_corrected.mat'))['salinasA_corrected']
        labels = sio.loadmat(os.path.join(DATA_PATH,
'SalinasA_gt.mat'))['salinasA_gt']
    elif name == 'PU':
        data = sio.loadmat(os.path.join(DATA_PATH,
'PaviaU.mat'))['paviaU']
        labels = sio.loadmat(os.path.join(DATA_PATH,
'PaviaU_gt.mat'))['paviaU_gt']
    return data, labels

def apply_pca(X, numComponents=75):
    """
    Apply PCA (Principal Component Analysis) to the input
data.

    Parameters:
    X (ndarray): Input data to which PCA will be applied.
    numComponents (int): Number of principal components to
return.

    Returns:
    newX (ndarray): Transformed data after applying PCA.
    pca (PCA): The PCA model fitted on the data.
    """
    newX = np.reshape(X, (-1, X.shape[2]))
    pca = PCA(n_components=numComponents, whiten=True)
    newX = pca.fit_transform(newX)
    newX = np.reshape(newX, (X.shape[0],X.shape[1],
numComponents))
    return newX, pca
```

```
def pad_with_zeros(X, margin=2):
    """
    Pad the input array with zeros around the border.

    Parameters:
    X (ndarray): Input array.
    margin (int): Width of the zero-padding.

    Returns:
    newX (ndarray): The zero-padded array.
    """
    newX = np.zeros((X.shape[0] + 2 * margin, X.shape[1] +
2* margin, X.shape[2]))
    x_offset = margin
    y_offset = margin
    newX[x_offset:X.shape[0] + x_offset,
y_offset:X.shape[1] + y_offset, :] = X
    return newX

def plot_model_history(history):
    """
    Plot the training history of a model.

    Parameters:
    history (History): History object obtained from the fit
method of a model.

    Returns:
    None
    """
    # Plotting the Loss Curve
    plt.figure(figsize=(5,5))
    plt.grid()
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.ylabel('Loss')
    plt.xlabel('Epochs')
    plt.legend(['Training','Validation'], loc='upper
right')
    plt.savefig("loss_curve.png")
    plt.show()

    # Plotting the Accuracy Curve
    plt.figure(figsize=(5,5))
```

```
    plt.ylim(0,1.1)
    plt.grid()
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.ylabel('Accuracy')
    plt.xlabel('Epochs')
    plt.legend(['Training','Validation'])
    plt.savefig("acc_curve.png")
    plt.show()

def build_model(input_shape, output_units):
    """
    Build a Convolutional Neural Network (CNN) model.

    Parameters:
    input_shape (tuple): Shape of the input data.
    output_units (int): Number of output units (number of
classes).

    Returns:
    model (Model): Compiled CNN model.
    """
    ## input layer
    input_layer = Input(input_shape)

    ## convolutional layers
    conv_layer1 = Conv3D(filters=8, kernel_size=(3, 3, 7),
activation='relu')(input_layer)
    conv_layer2 = Conv3D(filters=16, kernel_size=(3, 3, 5),
activation='relu')(conv_layer1)
    conv_layer3 = Conv3D(filters=32, kernel_size=(3, 3, 3),
activation='relu')(conv_layer2)

    conv3d_shape = tf.keras.backend.int_shape(conv_layer3)
    conv_layer3 = Reshape((conv3d_shape[1],
conv3d_shape[2],
conv3d_shape[3]*conv3d_shape[4]))(conv_layer3)
    conv_layer4 = Conv2D(filters=64, kernel_size=(3,3),
activation='relu')(conv_layer3)

    flatten_layer = Flatten()(conv_layer4)

    ## fully connected layers
```

```
    dense_layer1 = Dense(units=256,
activation='relu')(flatten_layer)
    dense_layer1 = Dropout(0.4)(dense_layer1)
    dense_layer2 = Dense(units=128,
activation='relu')(dense_layer1)
    dense_layer2 = Dropout(0.4)(dense_layer2)
    output_layer = Dense(units=output_units,
activation='softmax')(dense_layer2)

    # Define the model and print the summary
    model = Model(inputs=input_layer, outputs=output_layer)
    model.summary()

    return model

def generate_image_cubes(X, y, windowSize=5,
removeZeroLabels = True):
    """
    Generate 3D image cubes from the input data.

    Parameters:
    X (ndarray): Input data.
    y (ndarray): Corresponding labels of the data.
    windowSize (int): Size of the spatial window.
    removeZeroLabels (bool): If True, patches corresponding
to zero labels are not returned.

    Yields:
    (patch, patch_label): Tuples of image patches and
corresponding labels.
    """
    margin = int((windowSize - 1) / 2)
    zeroPaddedX = pad_with_zeros(X, margin=margin)
    # generate patches
    for r in range(margin, zeroPaddedX.shape[0] - margin):
        for c in range(margin, zeroPaddedX.shape[1] -
margin):
            patch = zeroPaddedX[r - margin:r + margin + 1,
c - margin:c + margin + 1]
            patch_label = y[r-margin, c-margin]
            if removeZeroLabels and patch_label > 0:
                yield (patch, patch_label - 1)
            elif not removeZeroLabels:
                yield (patch, patch_label)
```

```
if __name__ == '__main__':
    if not os.path.exists(MODEL_PATH):
        X, y = load_data(DATASET)
        K = 30
        X, _ = apply_pca(X, numComponents=K)
        patchesGenerator = generate_image_cubes(X, y,
windowSize=WINDOW_SIZE)
        X_patches = []
        y_patches = []
        for (patch, label) in patchesGenerator:
            X_patches.append(patch)
            y_patches.append(label)
        X = np.array(X_patches)
        y = np.array(y_patches)
        X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=TEST_RATIO, stratify=y)
        X_train = X_train.reshape(-1, WINDOW_SIZE,
WINDOW_SIZE, K, 1)
        print("Unique labels before to_categorical:",
np.unique(y_train))
        le = LabelEncoder()
        y_train_encoded = le.fit_transform(y_train)
        y_train = to_categorical(y_train_encoded)
        print("Unique labels after encoding:",
np.unique(y_train_encoded))
        # Determine the number of unique classes
        output_units = len(np.unique(y_train_encoded))
        model = build_model((WINDOW_SIZE, WINDOW_SIZE, K,
1), output_units)
        #compiling the model
        adam = tf.keras.optimizers.legacy.Adam(lr=0.001,
decay=1e-06)
        model.compile(loss='categorical_crossentropy',
optimizer=adam, metrics=['accuracy'])
        history = model.fit(x=X_train, y=y_train,
batch_size=256, epochs=100, validation_split=0.2)
        model.save(MODEL_PATH)
        plot_model_history(history)
    else:
        print(f'Model {MODEL_PATH} already exists.')
```

APPENDIX B

Software implementation of using DualConvHSINet model for object classification

```
import os
import numpy as np
import scipy.io as sio
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import Input, Conv3D, Conv2D,
Dense, Flatten, Reshape, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,
accuracy_score, classification_report, cohen_kappa_score
from keras.utils import np_utils
import spectral
from matplotlib import colors

DATASET = 'PU'
DATA_PATH = '/content/dataset/'
MODEL_PATH =
"/content/drive/MyDrive/trained_models/{}_hybrid_sn.h5".for
mat(DATASET)
WINDOW_SIZE = 25
TEST_RATIO = 0.7

def load_data(name):
    if name == 'IP':
        data = sio.loadmat(os.path.join(DATA_PATH,
'Indian_pines_corrected.mat'))['indian_pines_corrected']
        labels = sio.loadmat(os.path.join(DATA_PATH,
'Indian_pines_gt.mat'))['indian_pines_gt']
    elif name == 'SA':
        data = sio.loadmat(os.path.join(DATA_PATH,
'Salinas_corrected.mat'))['salinas_corrected']
        labels = sio.loadmat(os.path.join(DATA_PATH,
'Salinas_gt.mat'))['salinas_gt']
    elif name == 'PU':
```

```
        data = sio.loadmat(os.path.join(DATA_PATH,
'PaviaU.mat'))['paviaU']
        labels = sio.loadmat(os.path.join(DATA_PATH,
'PaviaU_gt.mat'))['paviaU_gt']
    return data, labels

def splitTrainTestSet(X, y, testRatio, randomState=345):
    X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=testRatio, random_state=randomState,
stratify=y)
    return X_train, X_test, y_train, y_test

def applyPCA(X, numComponents=75):
    newX = np.reshape(X, (-1, X.shape[2]))
    pca = PCA(n_components=numComponents, whiten=True)
    newX = pca.fit_transform(newX)
    newX = np.reshape(newX, (X.shape[0],X.shape[1],
numComponents))
    return newX, pca

def padWithZeros(X, margin=2):
    newX = np.zeros((X.shape[0] + 2 * margin, X.shape[1] +
2* margin, X.shape[2]))
    x_offset = margin
    y_offset = margin
    newX[x_offset:X.shape[0] + x_offset,
y_offset:X.shape[1] + y_offset, :] = X
    return newX

def createImageCubes(X, y, windowSize=5, removeZeroLabels =
True):
    margin = int((windowSize - 1) / 2)
    zeroPaddedX = padWithZeros(X, margin=margin)
    # split patches
    patchesData = np.zeros((X.shape[0] * X.shape[1],
windowSize, windowSize, X.shape[2]))
    patchesLabels = np.zeros((X.shape[0] * X.shape[1]))
    patchIndex = 0
    for r in range(margin, zeroPaddedX.shape[0] - margin):
        for c in range(margin, zeroPaddedX.shape[1] -
margin):
            patch = zeroPaddedX[r - margin:r + margin + 1,
c - margin:c + margin + 1]
            patchesData[patchIndex, :, :, :] = patch
```

```
            patchesLabels[patchIndex] = y[r-margin, c-
margin]
            patchIndex = patchIndex + 1
    if removeZeroLabels:
        patchesData = patchesData[patchesLabels>0,:,:,:]
        patchesLabels = patchesLabels[patchesLabels>0]
        patchesLabels -= 1
    return patchesData, patchesLabels
def Patch(data,height_index,width_index):
    height_slice = slice(height_index,
height_index+PATCH_SIZE)
    width_slice = slice(width_index,
width_index+PATCH_SIZE)
    patch = data[height_slice, width_slice, :]
    return patch

X, y = load_data(DATASET)
X.shape, y.shape
K = 30
X,pca = applyPCA(X,numComponents=K)
K = X.shape[2]
X, y = createImageCubes(X, y, windowSize=WINDOW_SIZE)
Xtrain, Xtest, ytrain, ytest = splitTrainTestSet(X, y,
TEST_RATIO)
Xtrain.shape, Xtest.shape, ytrain.shape, ytest.shape
Xtrain = Xtrain.reshape(-1, WINDOW_SIZE, WINDOW_SIZE, K, 1)
ytrain = np_utils.to_categorical(ytrain)
S = WINDOW_SIZE
L = K
output_units = 16
model = load_model(MODEL_PATH)
adam = tf.keras.optimizers.legacy.Adam(lr=0.001, decay=1e-
06)
model.compile(loss='categorical_crossentropy',
optimizer=adam, metrics=['accuracy'])
Xtest = Xtest.reshape(-1, WINDOW_SIZE, WINDOW_SIZE, K, 1)
ytest = np_utils.to_categorical(ytest)
Y_pred_test = model.predict(Xtest)
y_pred_test = np.argmax(Y_pred_test, axis=1)

# load the original image
X, y = load_data(DATASET)
height = y.shape[0]
width = y.shape[1]
```

```
PATCH_SIZE = WINDOW_SIZE
numComponents = K
X,pca = applyPCA(X, numComponents=numComponents)
X = padWithZeros(X, PATCH_SIZE//2)

from operator import truediv

def AA_andEachClassAccuracy(confusion_matrix):
    counter = confusion_matrix.shape[0]
    list_diag = np.diag(confusion_matrix)
    list_raw_sum = np.sum(confusion_matrix, axis=1)
    each_acc = np.nan_to_num(truediv(list_diag,
list_raw_sum))
    average_acc = np.mean(each_acc)
    return each_acc, average_acc
def reports (X_test,y_test,name):
    #start = time.time()
    Y_pred = model.predict(X_test)
    y_pred = np.argmax(Y_pred, axis=1)
    #end = time.time()
    #print(end - start)
    if name == 'IP':
        target_names = ['Alfalfa', 'Corn-notill', 'Corn-
mintill', 'Corn'
                        ,'Grass-pasture', 'Grass-trees',
'Grass-pasture-mowed',
                        'Hay-windrowed', 'Oats', 'Soybean-
notill', 'Soybean-mintill',
                        'Soybean-clean', 'Wheat', 'Woods',
'Buildings-Grass-Trees-Drives',
                        'Stone-Steel-Towers']
    elif name == 'SA':
        target_names =
['Brocoli_green_weeds_1','Brocoli_green_weeds_2','Fallow','
Fallow_rough_plow','Fallow_smooth',
                        'Stubble','Celery','Grapes_untraine
d','Soil_vinyard_develop','Corn_senesced_green_weeds',
                        'Lettuce_romaine_4wk','Lettuce_roma
ine_5wk','Lettuce_romaine_6wk','Lettuce_romaine_7wk',
                        'Vinyard_untrained','Vinyard_vertic
al_trellis']
    elif name == 'PU':
```

```
        target_names =
['Asphalt','Meadows','Gravel','Trees', 'Painted metal
sheets','Bare Soil','Bitumen',
                        'Self-Blocking Bricks','Shadows']

    classification =
classification_report(np.argmax(y_test, axis=1), y_pred,
target_names=target_names)
    oa = accuracy_score(np.argmax(y_test, axis=1), y_pred)
    confusion = confusion_matrix(np.argmax(y_test, axis=1),
y_pred)
    each_acc, aa = AA_andEachClassAccuracy(confusion)
    kappa = cohen_kappa_score(np.argmax(y_test, axis=1),
y_pred)
    score = model.evaluate(X_test, y_test, batch_size=32)
    Test_Loss =  score[0]*100
    Test_accuracy = score[1]*100

    return classification, confusion, Test_Loss,
Test_accuracy, oa*100, each_acc*100, aa*100, kappa*100

classification, confusion, Test_loss, Test_accuracy, oa,
each_acc, aa, kappa = reports(Xtest,ytest,DATASET)
classification = str(classification)
confusion = str(confusion)
file_name = "classification_report.txt"

with open(file_name, 'w') as x_file:
    x_file.write('{} Test loss (%)'.format(Test_loss))
    x_file.write('\n')
    x_file.write('{} Test accuracy
(%)'.format(Test_accuracy))
    x_file.write('\n')
    x_file.write('\n')
    x_file.write('{} Kappa accuracy (%)'.format(kappa))
    x_file.write('\n')
    x_file.write('{} Overall accuracy (%)'.format(oa))
    x_file.write('\n')
    x_file.write('{} Average accuracy (%)'.format(aa))
    x_file.write('\n')
    x_file.write('\n')
    x_file.write('{}'.format(classification))
    x_file.write('\n')
    x_file.write('{}'.format(confusion))
```

```
outputs = np.zeros((height,width))
for i in range(height):
    for j in range(width):
        target = int(y[i,j])
        if target == 0 :
            continue
        else :
            image_patch=Patch(X,i,j)
            X_test_image =
image_patch.reshape(1,image_patch.shape[0],image_patch.shap
e[1], image_patch.shape[2],
1).astype('float32')
            prediction = (model.predict(X_test_image))
            prediction = np.argmax(prediction, axis=1)
            outputs[i][j] = prediction+1

X2, y2 = load_data(DATASET)

spectral.imshow(X2, (54, 33, 14), stretch=(0.02,
0.98),figsize =(7,7))

spectral.imshow(classes = outputs.astype(int),figsize
=(7,7))

predict_image = spectral.imshow(X2, (54, 33, 14),
stretch=(0.02, 0.98), classes = outputs.astype(int),figsize
=(7,7))
predict_image.set_display_mode('overlay')
predict_image.class_alpha = 0.6
```

APPENDIX C

Methods	Indian Pines Dataset			University of Pavia Dataset			Salinas Scene Dataset		
	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA
2D-CNN	86.90 ± 1.3	85.01 ± 1.6	82.70 ± 1.0	96.02 ± 0.4	96.04 ± 0.3	95.10 ± 0.1	96.15 ± 0.6	95.71 ± 0.7	98.27 ± 0.2
3D-CNN	89.23 ± 0.2	87.70 ± 0.3	87.87 ± 0.1	97.30 ± 0.3	96.22 ± 0.1	97.02 ± 0.1	94.54 ± 0.5	93.81 ± 0.3	96.79 ± 0.6
M3D-CNN	93.67 ± 0.1	92.70 ± 0.3	93.60 ± 0.6	97.41 ± 0.2	96.05 ± 0.6	98.22 ± 0.1	94.92 ± 0.3	94.40 ± 0.1	97.28 ± 0.2
SSRN	99.23 ± 0.1	99.12 ± 0.1	92.52 ± 0.1	99.77 ± 0.1	99.69 ± 0.2	99.71 ± 0.1	99.88 ± 0.0	99.87 ± 0.0	99.84 ± 0.0
DualConvHSINet	99.47 ± 0.1	99.40 ± 0.1	99.38 ± 0.1	99.86 ± 0.1	99.82 ± 0.0	99.71 ± 0.1	100 ± 0.0	100 ± 0.0	100 ± 0.0

Table 4.1 The classification accuracies (in percentages) on Indian Pines, University of Pavia, and Salinas Scene datasets using proposed and state-of-the-art methods.

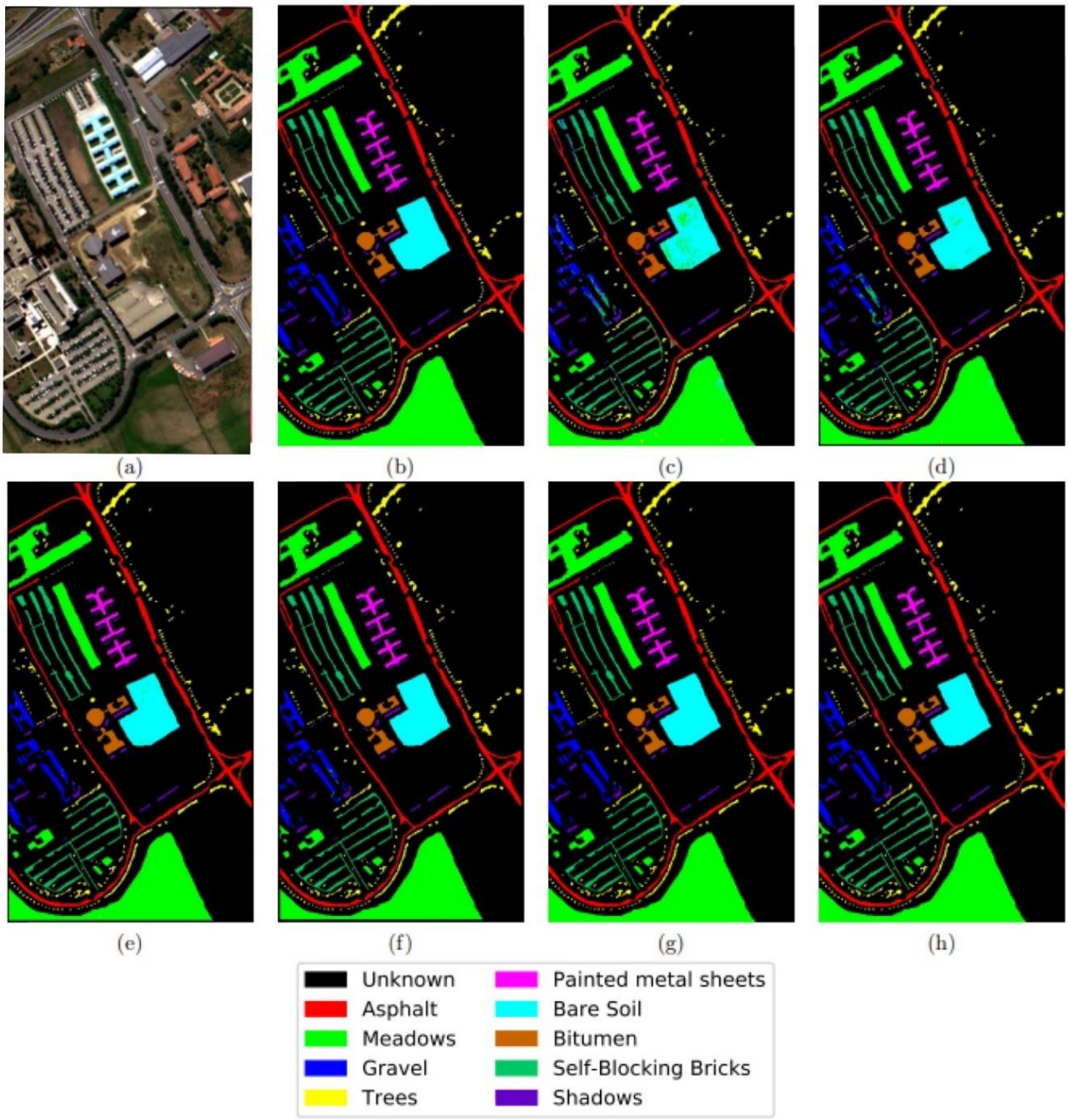

Figure 4.7 The Classification Map for Pavia University (a) False color image (b) Ground Truth (c)-(h) Predicted Classification Maps for SVM, 2D-CNN, 3D-CNN, M3D-CNN, SSRN, and DualConvHSINet

Data	2D CNN		3D CNN		HybridSN	
	Train(m)	Test(s)	Train(m)	Test(s)	Train(m)	Test(s)
IP	1.9	1.1	15.2	4.3	14.1	4.8
UP	1.8	1.3	58.0	10.6	20.3	6.6
SA	2.2	2.0	74	15.2	25.5	9.0

Table 4.2 The duration spent on training (expressed in minutes, m) and testing (expressed in seconds, s) using the 2D-CNN, 3D-CNN, and DualConvHSINet models across the IP, UP, and SA datasets.

Window	IP(%)	UP(%)	SA(%)	Window	IP(%)	UP(%)	SA(%)
19×19	99.74	99.98	99.99	23×23	99.31	99.96	99.71
21×21	99.73	99.90	99.69	25×25	99.75	99.98	100

Table 4.3 The impact of spatial window size over the performance of DualConvHSINet

Methods	Indian Pines			Univ. of Pavia			Salinas Scene		
	OA	Kappa	AA	OA	Kappa	AA	OA	Kappa	AA
2D-CNN	80.27	78.26	68.32	96.63	95.53	94.84	96.34	95.93	94.36
3D-CNN	82.62	79.25	76.51	96.34	94.90	97.03	85.00	83.20	89.63
M3D-CNN	81.39	81.20	75.22	95.95	93.40	97.52	94.20	93.61	96.66
SSRN	98.45	98.23	86.19	99.62	99.50	99.49	99.64	99.60	99.76
HybridSN	98.39	98.16	98.01	99.72	99.64	99.20	99.98	99.98	99.98

Table 4.4 The classification precision rates (expressed as percentages) attained through the use of both proposed and leading-edge techniques with a reduced volume of training data, specifically just 10%.