

DISSERTATION

CONTINUAL LEARNING FOR GEOSTATIONARY SATELLITE REMOTE SENSING OF
PRECIPITATION

Submitted by

Yifan Yang

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Summer 2026

Doctoral Committee:

Advisor: Mahmood Azimi-Sadjadi

Co-Advisor: Haonan Chen

Rocky Luo

Michael Kirby

Copyright by Yifan Yang 2026

All Rights Reserved

ABSTRACT

CONTINUAL LEARNING FOR GEOSTATIONARY SATELLITE REMOTE SENSING OF PRECIPITATION

Accurate precipitation retrieval from geostationary satellites remains a formidable challenge, primarily due to sensor limitations and the complexity of mapping satellite observations to surface precipitation. To address these challenges, this research proposes a deep learning (DL)-based framework for precipitation estimation using GOES-R satellite measurements. However, precipitation characteristics can vary substantially across geographic regions, storm types, and climatic conditions, causing DL models trained in one domain to perform poorly in another. Therefore, continual learning (CL) is needed to enable the model to learn from new precipitation domains while preserving knowledge acquired from previous domains. In this work, new CL algorithms have been developed to improve model generalization across multiple precipitation retrieval tasks. Three different incremental algorithms are introduced to efficiently update the learned representation as new domains are incorporated, enabling scalable adaptation without retraining from scratch. Building upon this framework, we design a CL-based precipitation retrieval system that explicitly accounts for multi-domain variability by allowing a single model to sequentially learn from different study domains while maintaining performance on previously learned domains. Experimental results demonstrate that: (1) the proposed DL-based framework enhances current operational precipitation products derived from GOES-R satellites; (2) the proposed CL algorithm improves the generalization capability of DL models, allowing them to perform robustly across multiple tasks; and (3) the integrated CL-based precipitation framework further strengthens performance across diverse study domains spanning the southeastern United States. These findings highlight the potential of the proposed approach as an important step toward scalable DL-based precipitation estimation systems for broader geostationary satellite applications.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Dr. Azimi-Sadjadi, for his guidance, dedication, and unwavering support throughout my research. He devoted an extraordinary amount of time carefully reviewing my dissertation and papers and providing insightful feedback that significantly strengthened this work. I am also deeply grateful to my co-advisor, Dr. Chen, for his invaluable mentorship and continuous support, as well as for providing numerous research opportunities and project experiences that greatly enriched my academic journey. I especially appreciate his assistance during challenging times, including his help with resources and his thoughtful guidance in both my research and student life. I would also like to thank my family and friends for their constant emotional support, encouragement, and understanding, which have been essential to my success.

This research was supported by the National Oceanic and Atmospheric Administration (NOAA) under Grant NA24OARX432C0007 and the Office of Naval Research (ONR) under Grant N00014-2512025.

DEDICATION

I would like to dedicate this dissertation to my lovely family and friends who have been supporting me.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGEMENTS	iii
DEDICATION	iv
LIST OF TABLES	viii
LIST OF FIGURES	ix
Chapter 1 Introduction	1
1.1 Background and Motivation	1
1.2 Review of Existing Sensors and Limitations	2
1.3 Problem Statement and Challenges	4
1.4 Research Objectives and Contributions	5
1.5 Organization of the Dissertation	7
Chapter 2 Satellite Observations and Feature Selection	8
2.1 Introduction	8
2.1.1 GOES-16 ABI Measurement	8
2.1.2 GOES-16 GLM Measurement	9
2.2 Ground-based Radar Measurement	10
2.3 Data Pre-processing	11
2.3.1 Brightness Temperature	11
2.3.2 Hourly Flash Rate	12
2.3.3 Spatial and Temporal Matching	12
2.4 Feature Normalization	14
2.5 Features Representation Analysis	15
2.6 Conclusion	17
Chapter 3 Precipitation Estimation using Deep Learning	18
3.1 Introduction	18
3.2 Model Architecture	19
3.3 Evaluation Metrics	22
3.4 Experimental Setup and Results	24
3.5 Discussion and Analysis	31
3.5.1 Comparison with Other DL Methods	32
3.5.2 Spatial and Temporal Resolution Mismatch	33
3.5.3 Parallax Effects in GOES-16 ABI Observations	33
3.5.4 Data Availability and Measurement Limitations	34
3.5.5 Additional Features	34
3.5.6 Effect of Lightning Data	34
3.6 Multi-Domain Limitations	35
3.7 Conclusion	36

Chapter 4	A Review of Continual Learning	38
4.1	Introduction	38
4.2	Brief Review of Key Continual Learning Methods	39
4.2.1	Regularization-based Methods	39
4.2.2	Replay-based Methods	42
4.2.3	Projection-based Methods	43
4.3	Conclusion	46
Chapter 5	Continual Learning using Incremental SVD	47
5.1	Introduction	47
5.2	Problem Formulation and Notation	47
5.3	Condition for Learning without Catastrophic Forgetting	49
5.4	Null-Space Projected Parameter Updates	50
5.5	Incremental SVD for Null-Space Estimation	52
5.6	CL-ISVD Algorithm	54
5.7	Computational Complexity Analysis	56
5.8	Experimental Setup	58
5.8.1	Benchmarking Datasets and Network Architecture	58
5.8.2	Evaluation Metrics	60
5.8.3	Baseline and Benchmark Methods	61
5.9	Quantitative Evaluation Results	63
5.10	Performance and Efficiency Analysis	66
5.10.1	Empirical Rank Determination	66
5.10.2	Impact of ISVD Update Group Size	67
5.10.3	Computational Efficiency	68
5.11	Conclusion	70
Chapter 6	Continual Learning using Subspace Tracking	72
6.1	Introduction	72
6.2	PCA Formulation of Null-Space Estimation	73
6.3	Incremental PCA for Null-Space Estimation	74
6.4	RLS for Null-Space Estimation	78
6.5	Continual Learning using Subspace Tracking	80
6.6	Computational Complexity Analysis	81
6.7	Experimental Results and Comparison	84
6.8	Conclusion	88
Chapter 7	Continual Learning for Multi-Domain Precipitation Estimation	90
7.1	Introduction	90
7.2	Study Domains and Model Training	91
7.3	Experimental Results and Evaluation	94
7.4	Domain Transition and Forgetting Analysis	97
7.4.1	Performance of Baseline Model without CL	97
7.4.2	Comparative Analysis of Null-Space-Based CL Methods	99
7.5	Conclusion	100

Chapter 8	Conclusion and Future Work	102
8.1	Conclusion	102
8.2	Future Work	103
Bibliography		105

LIST OF TABLES

2.1	Maximum and Minimum Values for Normalization.	14
3.1	Evaluation of Hourly Precipitation Estimation	26
5.1	Labels for Split SVHN and Split CIFAR-10	60
5.2	Performance Evaluation on Permuted MNIST	64
5.3	Performance Evaluation on Split MNIST	65
5.4	Performance Evaluation on Split SVHN	66
5.5	Performance Evaluation on Split CIFAR-10	67
5.6	Study of Batch Size for ISVD	70
5.7	Computational Speed Comparison	70
6.1	Comparison of computational complexity.	83
6.2	Compact Comparison of Benchmark and Proposed Methods	85
7.1	Geolocation of Five Study Domains for MDCL-GSPE	92
7.2	Testing Performance of MDCL-GSPE	96

LIST OF FIGURES

2.1	FDA Feature Representation Analysis.	16
3.1	Architecture of the baseline DL model for satellite precipitation estimation	20
3.2	Study domain of the baseline DL model.	22
3.3	Satellite Precipitation Estimation Results on 23:00 UTC, May 17, 2020.	25
3.4	Quantitative Evaluation Scores of the DL-based Precipitation Estimation on Hourly Scale	28
3.5	Daily Accumulation of Satellite Precipitation Estimation Results on Aug 20, 2020. . .	29
3.6	Quantitative Evaluation Scores of the DL-based Precipitation Estimation on Daily Scale	30
5.1	Weight updating of layer l for a new task $\mathcal{T}_t$	48
5.2	ACC and BWT vs. Tasks	68
5.3	Study of Hyperparameter η	69
5.4	Study of Empirical Rank Determination	69
6.1	ACC and BWT Comparison of CL-ISVD, CL-IPCA, and CL-RLS	86
6.2	Sensitivity to Hyperparameter η	87
7.1	Five Study Domains for MDCL-GSPE	92
7.2	Architecture of the DL model for MDCL-GSPE	94
7.3	Satellite Precipitation Estimation Results across Five Domains	95
7.4	Average Detection Metrics vs Domains	97
7.5	Average Estimation Metrics vs Domains	98
7.6	Metrics of Baseline Model Trained without Continual Learning	99
7.7	Satellite Precipitation Estimation without Continual Learning	100

Chapter 1

Introduction

1.1 Background and Motivation

Precipitation is a fundamental component of the regional and global hydrological cycle [1, 2]. Accurate precipitation estimation is essential for understanding water availability, predicting floods and droughts, and assessing the impacts of climate change [3–6]. As climate change intensifies the Earth’s water cycle and increases the frequency and severity of extreme weather events, reliable precipitation information becomes even more critical [2, 7]. Precipitation plays a central role in the analysis of extreme weather events, climate variability, and hydrological hazards [5, 8]. High-quality precipitation data support water resource management, agricultural decision-making, disaster mitigation, and infrastructure planning [3, 4, 9, 10]. Furthermore, accurate observations are indispensable for the development, calibration, and validation of climate models used to project long-term hydrological changes.

Precipitation measurements are generally categorized into two main types: ground-based observations, such as weather radars [11] and rain gauges [12], and satellite-based measurements [13–15]. Both types of observations are important and complementary for precipitation monitoring. While ground-based systems provide high accuracy at local scales, their spatial coverage is limited, especially over oceans, mountainous regions, and areas with sparse observing networks. In contrast, satellite observations offer near-global coverage with relatively high temporal frequency, making them essential for large-scale precipitation monitoring. However, the indirect nature of satellite measurements introduces inherent limitations in precipitation estimation. That is, satellites do not measure surface rainfall directly; instead, they observe cloud-top temperature, emitted or scattered radiation, or other atmospheric signals, which are related to precipitation. The relationship between these satellite observations and actual surface rainfall is complex, nonlinear, and often uncertain, particularly for different cloud types, precipitation regimes, and geographic re-

gions. These limitations motivate the need for improved precipitation estimation methods. These are briefly reviewed in the next section.

1.2 Review of Existing Sensors and Limitations

Rain gauges are among the most traditional instruments for measuring precipitation. While they provide direct and reliable point-based measurements, their spatial coverage is sparse and unevenly distributed. This limitation is particularly severe over oceans, mountainous terrain, and remote regions [12]. Moreover, rain gauges require continuous maintenance and calibration. Hence, the measurement accuracy can be affected by wind-induced undercatch, evaporation losses, mechanical blockage, and overflow errors [16]. As a result, rain gauge networks alone are insufficient for capturing the spatial variability of precipitation at regional and global scales.

To address these limitations, integrated systems such as the Multi-Radar Multi-Sensor (MRMS) system [11] developed by the National Oceanic and Atmospheric Administration (NOAA) combine ground-based weather radar, rain gauge data, and numerical weather prediction outputs to produce quantitative precipitation estimates (QPE). While MRMS significantly improves precipitation estimation over the contiguous United States (CONUS), its performance remains constrained by radar coverage. Radar beam blockage and terrain-induced limitations reduce accuracy in complex terrain, particularly in mountainous regions such as the western United States. In these areas, mountains can partially or completely block radar beams, and the radar beam may overshoot low-level precipitation as it travels farther from the radar site. These issues make it difficult to accurately capture precipitation intensity and spatial variability in complex terrain. In addition, radar-based products are generally unavailable over oceans because ground-based radar networks are located on land and have limited offshore coverage. As a result, precipitation monitoring over mountains and oceans remains highly challenging, further highlighting the need for satellite-based precipitation estimation.

Satellite remote sensing provides extensive spatial coverage with relatively high temporal resolution, making it indispensable for global precipitation monitoring. Both geostationary Earth

orbit (GEO) and low Earth orbit (LEO) satellites have been widely used for precipitation retrieval [17–19]. LEO passive microwave (PMW) sensors offer physically direct precipitation signals but suffer from limited temporal coverage [17, 18]. In contrast, GEO satellites provide continuous high-frequency observations using several infrared (IR) measurements, but require indirect retrieval approaches because IR sensors mainly observe cloud-top properties rather than surface rainfall. Therefore, rainfall must be inferred from cloud-top temperature, cloud structure, and related atmospheric features, whose relationship with surface precipitation is often complex and nonlinear [15, 19].

Several satellite-based precipitation products have been developed, including the Tropical Rainfall Measuring Mission (TRMM) [20], Precipitation Estimation from Remotely Sensed Information using Artificial Neural Networks (PERSIANN) [21], Climate Prediction Center Morphing Technique (CMORPH) [22], and the Global Precipitation Measurement (GPM) mission [14]. Although these products have significantly advanced global precipitation monitoring, trade-offs remain between spatial resolution, temporal resolution, and retrieval accuracy.

The latest generation of Geostationary Operational Environmental Satellites (GOES) provides high temporal resolution observations through the Advanced Baseline Imager (ABI), with full-disk observations available every 10 minutes [15, 23, 24]. The operational Rainfall Rate Quantitative Precipitation Estimate (RRQPE) product aboard GOES delivers precipitation estimates at approximately 2 km spatial resolution and 10-minute temporal resolution [15, 19]. The RRQPE algorithm employs a two-stage approach: precipitation detection followed by rainfall rate estimation using a combination of linear and nonlinear predictors derived from IR measurements [15, 19].

Despite its operational value, the RRQPE product exhibits limited performance in certain precipitation regimes, particularly mesoscale convective systems (MCSs) and orographic precipitation [25–29]. This limitation is mainly because IR-based predictors describe cloud-top properties rather than surface rainfall. For example, cold cloud tops may not always correspond to intense surface precipitation, while warm-rain processes or low-level precipitation may be missed or underestimated. In addition, MCSs contain complex cloud structures, mixed convective and stratiform

precipitation, and strong spatial variability, making the cloud-top-to-rainfall relationship highly nonlinear [25, 30]. Therefore, the regression-based formulation of RRQPE may not fully capture these complex cloud–precipitation relationships, often resulting in underestimation or overestimation of rainfall intensity. These limitations highlight the need for more advanced modeling frameworks capable of extracting complex precipitation signals from geostationary satellite observations.

1.3 Problem Statement and Challenges

Despite decades of research and operational advancements, accurate and robust precipitation retrieval remains a challenging problem, particularly at high spatial and temporal resolutions [15, 17, 19]. The main difficulty is that precipitation is highly variable in space and time, and satellite sensors do not usually measure surface rainfall directly. Instead, precipitation must be inferred from satellite-observed signals such as cloud-top temperature, emitted or scattered radiation, cloud structure, and other atmospheric features [15, 19]. Therefore, an effective precipitation retrieval framework must be able to extract reliable precipitation information from indirect observations while maintaining accuracy across different precipitation intensities, storm structures, and environmental conditions [25, 30].

One main challenge is sensor and data selection. Different satellite sensors provide different types of precipitation-related information. Passive microwave observations are more directly related to precipitation-sized hydrometeors, but their temporal resolution is limited and coverage gaps remain [17, 18]. In contrast, geostationary IR observations provide continuous high-frequency measurements, but they mainly describe cloud-top properties rather than surface rainfall [15, 19]. As a result, IR-based precipitation retrieval is an indirect inference problem. The relationship between cloud-top temperature, cloud texture, vertical structure, and surface rainfall intensity is highly nonlinear and often ambiguous. For example, cold cloud tops may not always produce intense surface rainfall, while warm-rain or low-level precipitation may be missed [15, 19, 31].

Effective precipitation estimation therefore requires careful feature design and representation learning. Important predictors may include multi-channel brightness temperatures, temporal gradients, spatial texture features, and lightning information [24, 32, 33]. However, selecting and integrating these features is nontrivial. The model must learn how to combine complementary information from different satellite observations while reducing uncertainty caused by indirect measurements. This makes model architecture design, feature extraction, and training strategy particularly important for geostationary satellite precipitation retrieval [27, 28, 34–36].

Another major challenge is the multi-domain generalization. Precipitation processes vary substantially across regimes such as convective and stratiform systems, and precipitation characteristics also differ across geographic and climatic domains due to variations in thermodynamic conditions, topography, and dominant weather systems [25, 29, 30]. As a result, models trained in one region often fail to generalize well to other regions because of distribution shifts. Traditional deep learning (DL) models are usually developed in a static training setting. When deployed in new domains, they either require retraining, which is computationally expensive, or fine-tuning, which may lead to catastrophic forgetting, where performance on previously learned domains deteriorates [37–41].

Therefore, there is a need for advanced methodologies that can leverage high-frequency satellite observations, integrate multi-source information, capture complex nonlinear cloud–precipitation relationships, and adapt to different precipitation regimes without sacrificing previously learned knowledge. Such capabilities are essential for developing accurate, robust, and scalable precipitation estimates across diverse conditions.

1.4 Research Objectives and Contributions

To address the challenges discussed earlier, this dissertation focuses on the following research objectives:

- Develop an effective DL-based framework for accurate and robust geostationary satellite precipitation estimation.

- Improve the ability of DL models to learn from new precipitation domains while reducing catastrophic forgetting.
- Develop a scalable multi-domain CL framework for quantitative precipitation estimation across different precipitation regimes and geographic regions.

Motivated by these objectives, this research developed a unified framework that integrates DL-based precipitation estimation with CL techniques [34–36,40,41]. The proposed approach enables a single model to sequentially learn from multiple precipitation regimes while preserving knowledge acquired from previously learned domains, addressing the catastrophic forgetting problem commonly observed in sequential learning [37–39]. To achieve this goal, new CL algorithms are developed based on null-space constrained parameter updates, allowing the model to adapt to new data without degrading previously learned representations [42–44].

The main contributions of this work are summarized as follows:

- Development of a duo-CNN precipitation estimation architecture using selected ABI and GLM observations.
- Development of a new CL methodology based on null-space constrained parameter updates to mitigate catastrophic forgetting during sequential learning while maintaining performance on previous tasks.
- Design of three incremental algorithms for efficiently computing and updating the null space during CL, ensuring computational scalability as new domains are introduced.
- Comprehensive experimental evaluation of the proposed CL approaches and comparison with representative state-of-the-art CL methods.
- Application of the proposed methods to multi-domain precipitation estimation using GOES-R satellite observations, along with extensive experiments and analysis to assess performance across different precipitation regimes and geographic domains.

1.5 Organization of the Dissertation

The remainder of this dissertation is organized as follows. Chapter 2 describes the satellite observations, ground reference data, preprocessing procedures, and feature selection used in this study. Chapter 3 presents a DL framework for precipitation estimation without continual learning (CL) and analyzes its limitations in multi-domain applications. Motivated by these limitations, Chapter 4 reviews the fundamental concepts and representative methods of CL. Chapter 5 develops a null-space-based CL algorithm using incremental SVD, referred to as CL-ISVD, and evaluates its performance on benchmark datasets. Chapter 6 presents two additional subspace-tracking-based CL methods, namely CL-IPCA and CL-RLS, and compares them with CL-ISVD and other representative CL approaches. Chapter 7 applies the proposed CL methods to multi-domain precipitation estimation using GOES-R satellite observations and evaluates their performance across multiple geographic domains. Finally, Chapter 8 summarizes the main findings of this dissertation and outlines directions for future work.

Chapter 2

Satellite Observations and Feature Selection

2.1 Introduction

The GOES-16 (current GOES-East) satellite was jointly developed and operated by NOAA and the National Aeronautics and Space Administration (NASA). It covers North and South America and the Atlantic Ocean to the west coast of Africa. In this work, we utilize the ABI (Advanced Baseline Imager) and GLM (Geostationary Lightning Mapper) measurements acquired from the GOES-16 satellite as inputs to the DL model. This chapter introduces the satellite observations, ground reference data, data preprocessing procedures, and feature selection analysis used in this study. First, the selected GOES-16 ABI infrared bands and GLM lightning measurements are described, with emphasis on their relevance to precipitation retrieval. Next, the ground-based MRMS precipitation product is introduced as the reference dataset for model training and evaluation. The chapter then presents the preprocessing steps used to construct the input features and target labels, including brightness temperature conversion, brightness temperature difference calculation, hourly flash rate generation, spatial and temporal matching, voxel extraction, and feature normalization. Finally, feature representation analysis is performed to examine the relative contribution of each input feature to precipitation retrieval at different rainfall intensities.

2.1.1 GOES-16 ABI Measurement

The full-disk ABI measurement is an imaging radiometer [24,32,45] with 10-minute temporal resolution, which images Earth’s weather, oceans, and environment using 16 spectral bands associated with different center wavelengths and spatial resolutions, including 2 visible bands, 4 near-IR bands, and 10 IR bands. For the purpose of precipitation retrieval, we select data of 5 IR spectral bands 8, 10, 11, 14, and 15, with 6.2 μm , 7.3 μm , 8.4 μm , 11.2 μm , and 12.3 μm center wavelength, respectively. These wavelengths are sensitive to water vapor, cloud-top properties, and thermal IR

signals that are useful for precipitation-related retrievals [15, 24, 32]. In contrast, the near-IR and visible bands in ABI measurements are less sensitive to water vapor and precipitation compared to these selected five IR bands. All these five bands have a spatial resolution of 2 km [32, 45]. The radiance data from these selected bands are converted into brightness temperatures (BTs), which are incorporated into the input features of the DL model [45].

Previous studies have demonstrated [32, 46, 47] that the brightness temperature differences (BTDs) derived from the GOES-16 ABI bands can be utilized to obtain precipitation retrieval products, as well as land surface temperature and cloud-top pressure products. For instance, the BTD between 6.2 and $7.3 \mu\text{m}$ (band 8 - 10) approximates the concentration and distribution of water vapor [24, 32]. The BTD between $8.4 \mu\text{m}$ and $11.2 \mu\text{m}$ (band 11- 14) can differentiate between thick and thin cirrus, and between clouds made of ice and those made of liquid [32, 46]. Consequently, the methodology described herein incorporates five BTD features in addition to the BT features.

2.1.2 GOES-16 GLM Measurement

The GLM measurement uses a single-channel, near-infrared optical transient detector to identify and indicate the presence of lightning. It measures the in-cloud, cloud-to-cloud, and cloud-to-ground lightning activities continuously over the Americas and adjacent ocean regions with near-uniform 8 km spatial resolution and 20 seconds refresh rate [33]. In the GLM measurement, the lightning data are ingested and assembled into three categories of high-level products: event, group, and flash, which can be easily and conveniently utilized for scientific research and broader operational applications. For instance, the flash product from the GOES-16 GLM measurement has been employed in [48] to improve convective precipitation forecasting. The GLM measurement has also been found to be valuable for estimating locations of strong radar echoes [49].

In the GOES-16 GLM measurement, an event, which is the basic unit of data from the GLM measurement, is defined as the occurrence of a single pixel exceeding the background threshold during a single frame. A group is defined as multiple simultaneous adjacent events in pixels

that occur in the same time integration frame. The reason for defining groups is that a lightning discharge is usually present in more than one pixel during a single time integration frame. The location information for the groups is stored in Earth-based latitude and longitude coordinates. A flash is defined as a set of groups sequentially separated in time by 330 ms or less and in space by no more than 16.5 km (nominally two pixels). Thus, a flash may include one group with a single event or it may consist of many groups that contain many events. The spatial information for flashes is in latitude and longitude to be consistent for the groups and flashes. In this work, we apply the high-level flash product as an additional input feature to our DL model in order to identify areas of deep convection with associated heavy rain rates.

2.2 Ground-based Radar Measurement

The ground-based Multi-Radar/Multi-Sensor System (MRMS) product was operationalized [11] in 2015 by NOAA, which integrates multiple radars, satellites, surface observations, upper air observations, lightning reports, rain gauges, and numerical weather prediction models in the continental United States (CONUS) and Canada, and generates quantitative precipitation estimates. The integration of radar with multi-sensor data provides more accurate inference of physical processes in the atmosphere than using radar data alone. Furthermore, a bias correction algorithm was later developed by Ware [50], which takes the density of the gauge network and the distance from the gauge into consideration. It properly weights the radar estimates and the gauge observations at each grid point to improve the quantitative precipitation estimates. The bias correction process for MRMS data can be summarized into the following three stages: a) calculation of hourly radar-gauge differences; b) interpolation of differences onto the MRMS grid via an inverse-distance-weighted mean scheme; c) subtraction of interpolated differences from radar QPE. The bias correction is performed separately for different subdomains within the CONUS region. The results by Zhang et al. in [11] demonstrated that this bias correction approach consistently improves the radar QPE accuracy throughout the year across the CONUS domain. This bias-corrected MRMS product has a spatial resolution of 1 km and includes the accumulation results ranging from 1 to

72 hours. In this work, we use the hourly bias-corrected MRMS rainfall estimates as the target or desired labels to train the DL model. This is due to the fact that MRMS has been identified as the most precise precipitation rainfall rate product at CONUS during the period of this study. This MRMS product is also used for performance evaluation in the testing phase.

2.3 Data Pre-processing

The data pre-processing phase comprises four key parts: (a) converting the radiance data from the GOES-16 ABI measurement into BT and BTD features; (b) forming the hourly flash rate feature from the GOES-16 GLM measurement; (c) partitioning the features into voxels; (d) labeling the partitioned voxels using the MRMS data in the training phase. In the testing phase, we repeat steps (a)-(c) for data pre-processing. Each of them will be demonstrated elaborately in the following.

2.3.1 Brightness Temperature

To convert the spectral radiance data of the selected bands in the GOES-16 ABI measurement into the BT, we use the transformation equation [45]

$$BT_i = [fk_2 / (\ln((fk_1/L_i) + 1)) - bc_1] / bc_2, \quad (2.1)$$

where $i \in \{8, 10, 11, 14, 15\}$ is the index for the GOES-16 ABI band; BT_i is the brightness temperature converted from the band i spectral radiance; L_i is the spectral radiance data from the band i ; bc_1 and bc_2 are the spectral response function offset and the scale correction term, respectively; fk_1 and fk_2 are the Planck function coefficients, which are derived as $fk_1 = 2hc\nu$, and $fk_2 = 2h\epsilon_b\nu$, where h is the Planck constant; b is the Boltzmann constant; c is the velocity of light; ν is the band central wavenumber. The default values of bc_1 , bc_2 , fk_1 , and fk_2 are used from the GOES-16 ABI measurement data files throughout this work.

Once the BT features are calculated, we also include the following BTD features:

$$\begin{aligned}
\text{BTD}_{10,8} &= \text{BT}_{10} - \text{BT}_8, \\
\text{BTD}_{11,10} &= \text{BT}_{11} - \text{BT}_{10}, \\
\text{BTD}_{14,10} &= \text{BT}_{14} - \text{BT}_{10}, \\
\text{BTD}_{11,14} &= \text{BT}_{11} - \text{BT}_{14}, \\
\text{BTD}_{14,15} &= \text{BT}_{14} - \text{BT}_{15},
\end{aligned} \tag{2.2}$$

where BTD_{i_1, i_2} , $i_1, i_2 \in \{8, 10, 11, 14, 15\}$ is the brightness temperature difference of BT_{i_1} and BT_{i_2} .

2.3.2 Hourly Flash Rate

In GOES-16 GLM measurement, flashes are recorded and stored with the associated longitude, latitude, and radiant energy. To form the hourly flash rate feature, we initialize a grid with a 2 km spatial resolution over the selected study domain for each hour. We then find the coordinates of the flashes within this hour and assign the grid pixel value with the occurrence of the flashes at the corresponding location. It should be noted that the flash occurrence is counted regardless of the released flash energy. Also, the spatial resolution formed hourly flash rate feature is 2 km while the spatial resolution of the GOES-16 GLM measurement is 8 km. When the coordinates of the flashes are not perfectly matched to the grid coordinates, we find the closest locations in the feature grid as the corresponding locations for the flashes.

2.3.3 Spatial and Temporal Matching

In summary, a total of 11 features are stacked together as the input features to the DL model, including 5 BT features, 5 BTD features, and the GLM flash rate. To apply these features to the model, we partition them into $11 \times 29 \times 29$ voxels as data cube or voxel with stride size 1 in both horizontal and vertical dimension and feed them as input to model. The selection of 29×29 voxel size in each band is based on the following consideration. CNNs excel at capturing

spatial information from data. By utilizing nearby information surrounding the voxel center, we aim to effectively determine the rainfall rate at that specific location while considering the local neighborhood. However, if the voxel size is too large, it may encompass numerous precipitation patterns that could potentially confuse the CNN and yield inaccurate results. If the voxel size is too small, the input may not contain sufficient spatial context to represent the surrounding cloud structure and precipitation pattern, which can limit the model’s ability to distinguish precipitating and non-precipitating regions. We have conducted a series of experiments involving different voxel sizes to determine the optimal choice. After careful evaluation, we experimentally confirm that a voxel size of 29×29 yields the best performance.

To align the spatial resolution of GOES measurements and MRMS, we down-sample the MRMS data from 1 km to 2 km to match the resolution of the input features and generate desired labels for the partitioned voxels. Specifically, we retain every other pixel on the MRMS grid, both column-wise and row-wise. This sub-sampling approach allows us to reduce the spatial resolution of the MRMS data while still capturing the essential information needed for our analysis. The MRMS estimates at the center of the partitioned voxels are then utilized as label for the data cube $11 \times 29 \times 29$. In the case of precipitation detection, the partitioned data cubes are labeled as precipitation if the MRMS rainfall rates exceed 0.1 mm/hr, and as non-precipitation otherwise. For precipitation estimation, the MRMS estimates at the center of the partitioned voxels are taken as the target values of the partitioned data cubes.

To accommodate the 1-hour temporal resolution of the bias-corrected MRMS data, the BTs and BTDs with a temporal resolution of 10 minutes are averaged over the six scans within an hour. By doing so, the resulting averaged BTs and BTDs are treated as the representative BT and BTD features for that specific hour.

2.4 Feature Normalization

Before apply to the DL model, the pixel values in the data cubes were normalized between 0 and 1 for each individual feature using

$$BT_{i,norm} = \frac{BT_i - BT_{i,min}}{BT_{i,max} - BT_{i,min}}, \quad (2.3)$$

where $BT_{i,max}$ and $BT_{i,min}$ are the maximum and the minimum values of the BT for band i in the training set. The normalization for the BTD and flash rate features were done in a similar manner. It should be noted that the BT features, the BTD features, and the flash rate features had different maximum and minimum values, which needed to be taken into account in normalization. The maximum and minimum values for the input features used here are presented in Table 2.1. Moreover, the MRMS label is also normalized to ensure all the label values are between 0 and 1.

Table 2.1: Maximum and Minimum Values used for normalizing the input features and labels.

feature	Maximum	Minimum
BT ₈ (K)	257.44	200.37
BT ₁₀ (K)	273.46	199.81
BT ₁₁ (K)	310.46	198.95
BT ₁₄ (K)	314.75	198.63
BT ₁₅ (K)	308.96	198.19
BTD _{10, 8} (K)	29.08	-5.89
BTD _{11,10} (K)	49.86	-1.37
BTD _{14,10} (K)	54.40	-1.80
BTD _{11,14} (K)	10.83	-7.40
BTD _{14,15} (K)	8.32	-0.43
Lightning (Count/hr)	5	0
MRMS (mm/hr)	80	0

2.5 Features Representation Analysis

To investigate the impact of each input feature on the precipitation retrieval for different levels of precipitation rainfall rates, Fisher discriminant analysis (FDA) [51] was applied to the 11 input features across four distinct rainfall rate thresholds, including 1, 4, 7, and 10 mm/hr. By employing FDA, we were able to quantitatively measure the discrimination of the features based on the class-dependent statistics of the dataset, where the highest Fisher measure indicates the most discriminatory feature.

For each rainfall rate threshold, the voxels were divided into two classes. Class 0 represents the voxels whose precipitation labels are below the selected threshold, while Class 1 represents the voxels whose precipitation labels are above the selected threshold. For example, when the threshold is 10 mm/hr, Class 1 corresponds to heavy-precipitation voxels with rainfall rates greater than 10 mm/hr, and Class 0 corresponds to voxels with rainfall rates less than or equal to 10 mm/hr. Therefore, the FDA score measures how well each input feature separates the two precipitation classes at a given rainfall intensity threshold.

For each threshold, the Fisher measure $FDA(i)$ for the band i feature was calculated as

$$FDA(i) = \frac{|\mu_0(i) - \mu_1(i)|^2}{\sigma_0^2(i) + \sigma_1^2(i)}, \quad (2.4)$$

where $\mu_0(i)$ and $\mu_1(i)$ are the pixel mean of the band i feature over all the pixels in voxels whose precipitation estimation target labels are above and below the threshold, respectively; $\sigma_0^2(i)$ and $\sigma_1^2(i)$ are the pixel variance of the band i feature over all the pixels in voxels whose precipitation estimation target labels are above and below the threshold, respectively. In this analysis, we selected 64,000 voxels randomly from the data in 2019 and obtained the Fisher measures of the 11 input features. Figure 2.1 illustrates the Fisher measures of the input features at different rainfall intensities determined by the rainfall rate thresholds.

As can be observed from Figure 2.1, we note that the Fisher measure of both BT and BTD features remained relatively constant across varying rainfall rate thresholds. In general, BT features

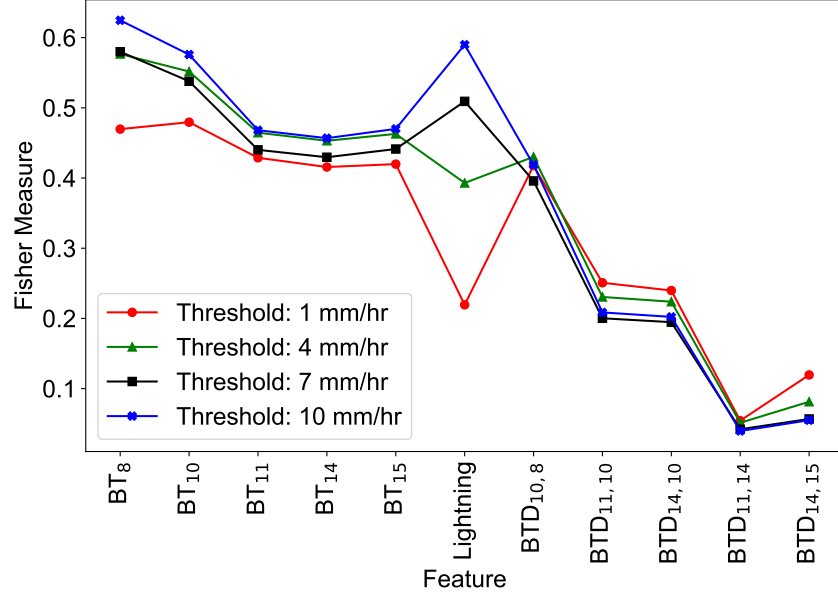


Figure 2.1: Fisher measures of the input features at different rainfall intensities determined by the rainfall rate thresholds. The x-axis includes all the input features to the DL model, where BT_i and BTD_{i_1, i_2} , $i, i_1, i_2 \in \{8, 10, 11, 14, 15\}$ refer to the BT from the GOES-16 ABI band i , and the BTD from band i_1 subtracting band i_2 , respectively. Lightning refers to the hourly flash rate obtained from the GOES-16 GLM measurement.

exhibited greater impact than BTD features, with the BT feature from ABI band 8, having a center wavelength of $6.2 \mu\text{m}$, being the most informative. Nonetheless, with regards to the hourly flash rate feature (denoted as *Lightning* in Figure 2.1), its Fisher measure exhibited remarkable variations for different rainfall rate thresholds. As the rainfall rate threshold increased from 1 mm/hr to 10 mm/hr, the Fisher measure of the hourly flash rate feature increased from 0.22 to 0.59 approximately. Nonetheless, the hourly flash rate became the second most informative feature when the rainfall rate threshold was 10 mm/hr. The Fisher measure results for the Lightning feature are consistent with the fact that higher rain rates are usually associated with deeper and more intense convection compared to lower rain rates. These findings suggest that the lightning data could potentially play a pivotal role in precipitation retrieval, particularly for heavy precipitation.

2.6 Conclusion

This chapter introduced the satellite observations, ground-based reference data, and data preparation procedures used in this study. The geostationary satellite observations from GOES-16 ABI were selected as the primary input data source due to their high temporal resolution and continuous coverage, which are essential for monitoring rapidly evolving precipitation systems. In addition, lightning observations from the GOES-16 GLM were incorporated to provide complementary information on convective activity. For ground reference precipitation measurements, the MRMS product was used as the ground-truth for labeling and evaluation because of its high accuracy, reliability, and high spatial resolution and comprehensive radar-based precipitation estimates.

To ensure consistency between satellite observations and ground reference data, several preprocessing steps were performed, including brightness temperature extraction, lightning flash rate computation, and spatial-temporal matching between satellite inputs and MRMS precipitation labels. These steps enable the construction of a reliable dataset for training and evaluating machine learning models for precipitation estimation.

Finally, a feature importance analysis was conducted to examine the contribution of different satellite-derived variables. The results indicate that lightning-related features play a significant role in identifying strong precipitation events, highlighting the value of incorporating lightning observations in satellite-based precipitation retrieval. The processed dataset and identified key features provide the foundation for the DL precipitation estimation framework presented in the following chapters.

Chapter 3

Precipitation Estimation using Deep Learning

3.1 Introduction

This chapter presents the baseline DL-based framework for geostationary satellite precipitation estimation. The objective is to retrieve surface precipitation rates from GOES-16 satellite observations by learning the nonlinear relationship between satellite-derived input features and ground-based precipitation reference data. As discussed in previous chapters, geostationary satellite observations provide high temporal resolution and broad spatial coverage, but precipitation retrieval from these measurements remains challenging because the satellite sensors observe cloud-top and atmospheric properties rather than surface rainfall directly. Therefore, an effective DL framework is needed to extract useful spatial and physical information from multi-channel satellite observations for accurate precipitation estimation.

As stated in Chapter 2, the input vector $\mathbf{x} \in \mathbb{R}^M$ to the model consists of features from multi-channel IR brightness temperatures and brightness temperature differences of the GOES-16 ABI and GLM. The corresponding surface precipitation rate is obtained from the MRMS product, and serves as the ground truth in this learning task. The objective is to learn a mapping function $f(\mathbf{x}; \boldsymbol{\theta}) : \mathbb{R}^M \rightarrow \mathbb{R}$, where $f(\cdot)$ represents a nonlinear model parameterized by $\boldsymbol{\theta}$. The model aims to estimate surface precipitation from satellite observations by minimizing the sum square error (or other measures of performance) between the predicted precipitation $\hat{y}$ and the target precipitation $y \in \mathbb{R}$ over a training dataset $(\mathbf{x}_i, y_i)_{i=1}^N$.

To model the complex spatial relationships between satellite observations and precipitation, this chapter proposes a DL-based framework based on CNNs. CNNs are particularly suitable for this task because they can effectively capture spatial structures and hierarchical patterns in satellite imagery through convolutional filters and pooling operations [52, 53]. The proposed framework employs two CNN models: one for precipitation detection and another for rainfall rate estimation.

By integrating ABI-derived IR features and GLM lightning measurements, the framework aims to better characterize cloud structures, convective activity, and precipitation intensity.

The remainder of this chapter is organized as follows. Section 3.2 describes the proposed DL framework, including the detection CNN and estimation CNN modules, and introduces the study domain used in this chapter. Section 3.3 presents the evaluation metrics used to assess precipitation detection and estimation performance. Section 3.4 describes the experimental configuration, including the training, validation, and testing datasets, and presents the experimental results comparing the DL-based retrievals with the operational GOES-16 RRQPE product. Section 3.5 provides further discussion and analysis of the model performance, input features, observational uncertainties, and limitations. Section 3.6 discusses the limitations of the baseline DL framework in multi-domain applications and motivates the CL methods introduced in the following chapters. Finally, Section 3.7 concludes this chapter.

3.2 Model Architecture

The proposed architecture is based on deep convolutional neural networks (CNNs), which have achieved significant success in applications such as land cover classification, cloud detection, and atmospheric parameter retrieval, where complex nonlinear relationships exist between satellite observations and geophysical variables. We demonstrate the potential of DL models to extract meaningful features from multi-spectral and multi-channel satellite imagery that enable accurate precipitation estimation.

Figure 3.1 details the proposed architecture. Once the GOES-16 ABI and GLM measurements have been pre-processed, the partitioned data cubes are applied to the DL model. The DL model consists of two components: a detection CNN module (CNN-1) and an estimation CNN module (CNN-2), both of which have three convolutional layers, two average pooling layers, and one dense fully connected layer. For all the convolutional layers in CNN-1 and CNN-2, the kernel size is set to 4×4 with stride size 1. The numbers of feature maps in the three convolutional layers for both CNN-1 and CNN-2 are set to 64, 128, and 128, respectively. All the pooling layers in CNN-1

and CNN-2 use 2×2 average pooling mask with stride 2. In CNN-1, the feature maps at the third convolutional layer are fully connected to the dense layer, which has 2 neurons to predict the detection results, i.e., precipitation and non-precipitation. In CNN-2, the feature maps at the third convolutional layer are fully connected to a single-neuron dense layer for precipitation estimation.

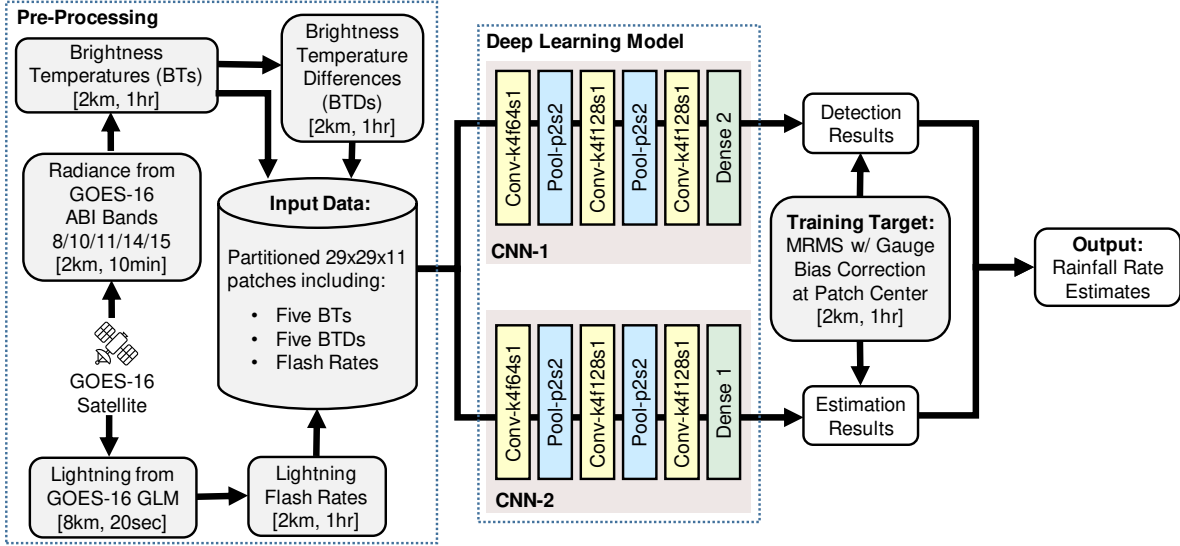


Figure 3.1: The DL framework for precipitation retrievals using GOES ABI and GLM data. In the convolutional (Conv-) and pooling (Pool-) layers, k, f, s, and p represent kernel size, number of feature maps, stride size, and pooling size, respectively. The spatio-temporal resolution of the data and data pre-processing is also illustrated. The rainfall rate estimates are determined by combining the detection results from CNN-1 and estimation results from CNN-2.

Both CNN modules used ReLU as the activation function at each neuron. The DL model was optimized using the Adam optimizer with learning rates $2e-6$ and $2e-7$ for CNN-1 and CNN-2, respectively. The training batch size was set to 100 and the training epoch was set to 100. All experiments were implemented using the Tensorflow platform [54].

In the training phase, the losses between the predicted results and the actual target labels provided from the MRMS data are calculated for both modules, in which the CNN-1 uses the binary cross-entropy loss, and the CNN-2 uses a weighted sum squared error (SSE) loss to estimate the precipitations according to their target rainfall rates. The weighted MSE loss in CNN-2 is defined

as

$$\text{Loss}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N (1 + y_n)(\hat{y}_n(\boldsymbol{\theta}) - y_n)^2, \quad (3.1)$$

where $\boldsymbol{\theta}$ denotes the model parameter set in CNN-2, N is the total number of training voxels; y_n and $\hat{y}_n$ are the actual and the predicted label for the n th training voxel, respectively. One can notice that the heavier the precipitation in a voxel is, the larger weight it gets. Since the actual rainfall rate y_n is always non-negative, $(1 + y_n)$ ensures the weight is always greater than or equal to 1. The losses are then back-propagated to the CNN modules to update the network parameters individually [53, 55].

In the testing phase, the test data features are pre-processed into $11 \times 29 \times 29$ voxels similar to the training data, and then applied to the DL model. The final precipitation retrievals are determined by fusing the results from the two CNNs. Specifically, if CNN-1 detects precipitation for a given voxel, the rainfall rate at the center location of that voxel is estimated based on the outcome of CNN-2. Conversely, if CNN-1 does not detect precipitation in a voxel, the corresponding rainfall rate at the center location for that voxel is set 0 mm/hr.

In this dissertation, a rectangular region in the southeastern United States is selected as the study domain to examine the effectiveness of the DL-based precipitation estimation framework, which spans from 81°W to 97°W in longitude and 31°N to 39°N in latitude, covering an area of approximately $1600 \text{ km} \times 800 \text{ km}$ as shown by the red rectangle in Figure 3.2. This region was chosen due to the fact that it primarily consists of relatively flat terrains with similar characteristics as the digital elevation model (DEM) information in Fig. 3.2 presents. The deployment of radars and rain gauge sites across this area has been extensive, owing to the absence of complex terrain features like high mountains or lakes [11]. As a result, the MRMS precipitation estimates in this region are considered reliable and accurate with the bias-correction algorithm, making them suitable as target labels for our DL model. Although the main challenge is the mesoscale convective precipitation in western and the mid-western United States as described above, this chapter is describing a prototype algorithm and the selected region is relatively “simple” in terms of terrain and has accurate QPE for training.

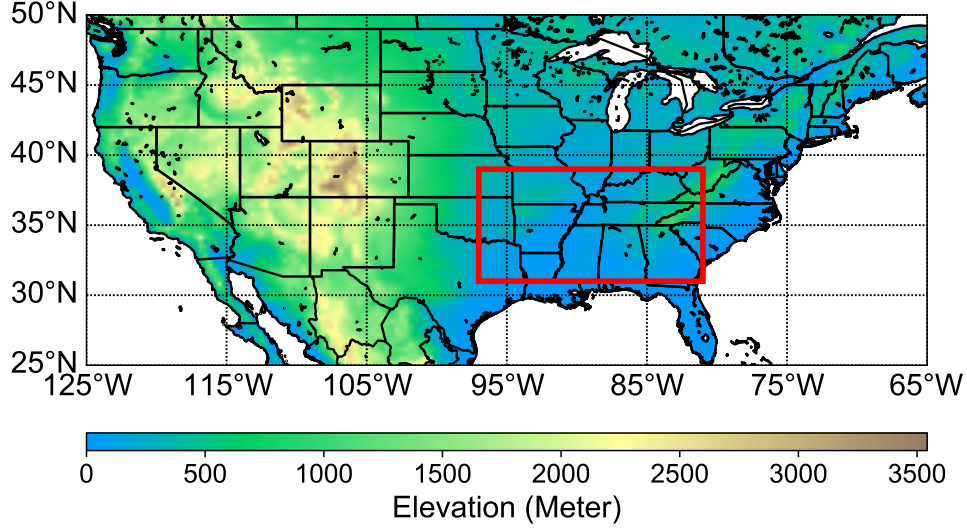


Figure 3.2: Location of the study domain (denoted by the red rectangle) in the CONUS. The background color indicates the digital elevation model (DEM) information.

To train the model, the ground-based MRMS product is used for the target labels. The data from May to September in 2019 are divided into training and validation data. The data from May to September in 2020 are used for the independent test. The period of May to September is selected mainly because it is the rainy season in our study domain. Detailed information on each dataset used in this dissertation will be provided in the following.

3.3 Evaluation Metrics

The detection performance of the proposed system is evaluated in terms of Heidke skill score (HSS), critical success index (CSI) [56], probability of detection (POD), and false alarm ratio (FAR), which are defined as

$$\text{HSS} = \frac{2(\text{TP} \times \text{TN} - \text{FN} \times \text{FP})}{(\text{TP} + \text{FN})(\text{FN} + \text{TN}) + (\text{TP} + \text{FP})(\text{FP} + \text{TN})}, \quad (3.2)$$

$$\text{CSI} = \frac{\text{TP}}{\text{TP} + \text{FN} + \text{FP}}, \quad (3.3)$$

$$\text{POD} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad (3.4)$$

$$\text{FAR} = \frac{\text{FP}}{\text{TN} + \text{FP}}, \quad (3.5)$$

where TP (true positive) is the number of pixels predicted and labeled as precipitation; FP (false positive) is the number of pixels labeled as non-precipitation, but predicted as precipitation; FN (false negative) is the number of pixels labeled as precipitation, but predicted as non-precipitation; TN (true negative) is the number of pixels predicted and labeled as non-precipitation.

The integrated HSS and CSI metrics directly reflect the model performance. Ideally, we want higher POD, HSS, CSI, and lower FAR. Furthermore, to investigate the precipitation detection performance at different rainfall rates, the evaluation metrics are computed based on a number of rainfall rate thresholds. This is done by transforming the DL-based estimates and the ground-based MRMS product into binary arrays using a pre-defined threshold. Specifically, if the rainfall rate of the DL-based estimate or the ground-based MRMS product at a grid pixel is higher than the threshold, “1” is assigned to this grid pixel, otherwise “0” will be assigned. For each threshold, we compute the confusion matrix between the transformed binary arrays from the proposed DL-based rainfall rate estimates and the corresponding ground-based MRMS product to further obtain the four detection evaluation metrics above.

The performance of the precipitation estimation is evaluated [57, 58] in terms of SSE, mean absolute error (MAE), normalized mean error (NME), and the normalized mean absolute error (NMAE), which are calculated as

$$\text{MSE} = \frac{1}{M} \sum_{m=1}^M (\hat{y}_m - y_m)^2, \quad (3.6)$$

$$\text{MAE} = \frac{1}{M} \sum_{m=1}^M |\hat{y}_m - y_m|, \quad (3.7)$$

$$\text{NME} = \frac{\sum_{m=1}^M \hat{y}_m - y_m}{\sum_{m=1}^M y_m}, \quad (3.8)$$

$$\text{NMAE} = \frac{\sum_{m=1}^M |\hat{y}_m - y_m|}{\sum_{m=1}^M y_m}, \quad (3.9)$$

where $m \in [1, M]$ is the testing voxel index; y_m is the rainfall rate from MRMS ground truth, $\hat{y}_m$ is the predicted rainfall rate estimates; M is the total number of testing voxels. It is important to compute and show all of these metrics to gain a comprehensive understanding of the model’s performance. Each metric captures different aspects of the errors, providing valuable insights. Specifically, MSE and MAE quantify the squared and the absolute errors, respectively, while NME and NMAE provide a relative perspective by normalizing the errors.

To assess the efficacy of precipitation estimation under varying intensities, we adopt the identical rainfall rate thresholds employed in evaluating the detection performance of precipitation. At each threshold level, we determine the grid pixel indices where the ground-based MRMS product exceeds the threshold value. We subsequently calculate the four categories of estimation evaluation errors between the predicted rainfall rates and the ground-based MRMS product exclusively for these selected pixel indices.

It should be noted that all evaluation metrics reported in this chapter are computed using the final fused precipitation estimates from the complete DL system. Thus, the results reflect the overall performance of the proposed framework on the entire testing set, rather than the individual performance of CNN-1 or CNN-2 alone.

3.4 Experimental Setup and Results

Once the GOES-16 ABI and GLM measurements were pre-processed and partitioned into $11 \times 29 \times 29$ voxels, a total of 6.4 million voxels from the data in 2019 were randomly selected for training the designed DL model, for which the numbers of precipitation voxels and non-precipitation voxels are equal. Then, the DL model was validated on 6.4 million validation voxels, which were also randomly selected from the data in 2019. The trained DL model was then tested on the test voxels in 2020. For the testing set, we sampled 12.8 million voxels from the dataset collected in 2020, for which the numbers of precipitation and non-precipitation are equal. The reason for the sampling was to ensure a well-balanced representation of both precipitation and non-precipitation voxels in the testing dataset. Including all non-precipitation samples from

the dataset would have led to an imbalance in the testing set, potentially biasing certain evaluation metrics such as TN, MSE, and MAE. By using a testing set with a balanced representation of precipitation and non-precipitation samples, we were able to obtain reliable and unbiased evaluation results for the framework’s performance.

In the following figures and tables, the proposed framework is referred to as “CNN w/ GLM”. In order to experimentally quantify the impact of the GLM data, another model with the same architecture in Figure 3.1 was trained except without using the GLM measurement for comparison, hence the input voxels for this model were of size $10 \times 29 \times 29$. Otherwise, the loss functions, activations, optimizer, hyper-parameters, and the network architecture beside the input layer remained the same. This model is referred to as “CNN w/o GLM”.

The experimental results for both accumulation periods were then evaluated using the metrics in Section 3.3 with different thresholds, including the performance comparison of the proposed DL-based precipitation retrieval with and without GLM measurement, as well as the currently operational GOES-16 RRQPE product.

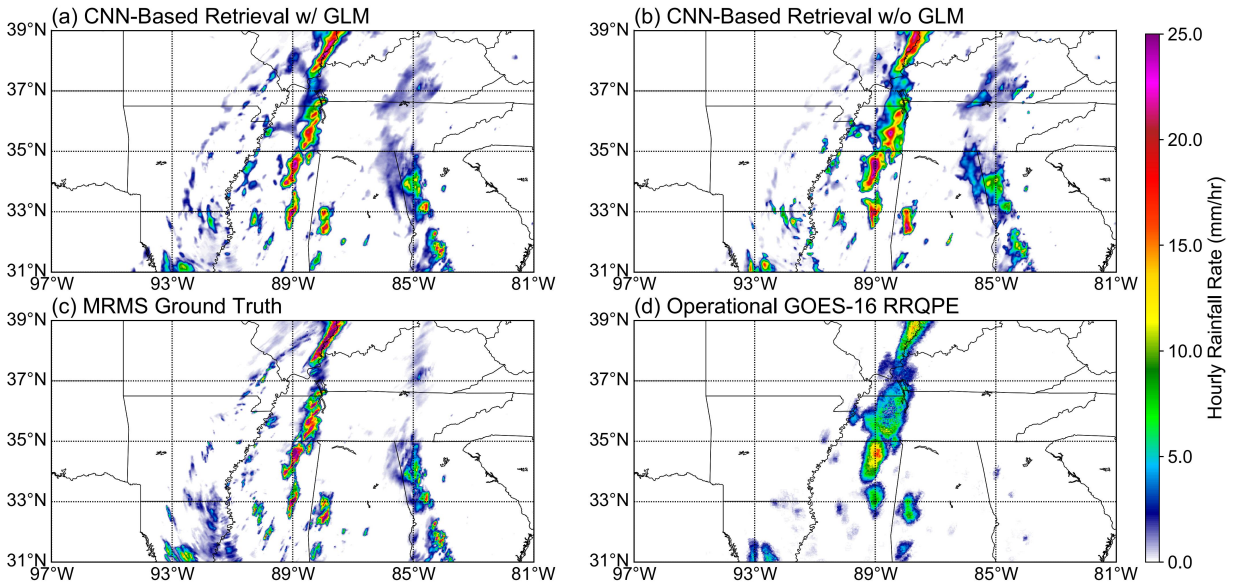


Figure 3.3: Example of hourly precipitation estimates at 23:00 UTC, May 17, 2020: (a) CNN-based retrieval with GLM data; (b) CNN-based retrieval without GLM data; (c) ground-based MRMS benchmark product; (d) operational GOES-16 RRQPE product.

Table 3.1: Performance of hourly precipitation retrievals from DL models and the operational GOES-16 RRQPE product on validation and test data.

		HSS	CSI	POD	FAR	MSE (mm ²)	MAE (mm)	NME	NMAE
Valid.	CNN w/ GLM	0.73	0.74	0.78	0.06	27.62	1.98	-0.04	0.48
	CNN w/o GLM	0.70	0.74	0.91	0.23	48.32	2.76	-0.10	0.66
	Operational	0.52	0.54	0.55	0.03	73.97	3.55	-0.52	0.75
Test	CNN w/ GLM	0.71	0.73	0.77	0.06	31.79	2.49	-0.09	0.52
	CNN w/o GLM	0.67	0.74	0.91	0.24	50.59	3.56	-0.11	0.74
	Operational	0.53	0.55	0.57	0.03	71.49	3.65	-0.53	0.76

Figure 3.3 shows a visual example of the hourly precipitation estimates at 23:00 UTC, May 17, 2020, including the CNN-based precipitation retrieval with and without the hourly flash rate feature from the GLM measurement, the MRMS ground truth, and the currently operational GOES-16 RRQPE product. As can be seen, comparing to the benchmark, the CNN-based precipitation retrieval outperformed the operational RRQPE product as it was able to identify heavy precipitation areas that the operational RRQPE product missed. The CNN-based retrieval provided better estimates for both weak and intense precipitation, particularly in areas with intense precipitation when using GLM lightning data compared to the model without lightning data. Despite some false alarmed areas, such as the region near 85°W, 37°N, the estimates from the CNN-based retrieval with the lightning data had lower errors than the estimates from the CNN-based retrieval without the GLM lightning data and the operational RRQPE product, and the operational RRQPE product completely failed to indicate the precipitation over this area. These visual and comparative observations between the CNN-based retrieval with and without GLM corroborated our conclusion in Section 2.5 that the lightning data from the GLM measurement played a crucial role in precipitation retrieval.

The quantitative evaluation results of the CNN-based precipitation retrieval and the operational product on the validation data and the test data were summarized in Table 3.1, where the best scores were indicated in bold. The evaluation metrics for detection focused on distinguishing be-

tween precipitation and non-precipitation, whereas the assessment metrics for estimation encompassed all levels of precipitation events. In general, the CNN-based retrieval approach outscored the operational GOES-16 RRQPE product. For the integrated HSS and CSI metrics, the CNN-based retrieval with the lightning data had at least 32% improvement on the testing data. For the estimation errors, it reduced the MSE, MAE, and NMAE by at least 31%. Although the CNN-based model without GLM had higher POD, it also had a relatively high FAR. This indicates that the model tended to classify more pixels as precipitation, which increased the number of correctly detected precipitation pixels but also introduced more false alarms. Without the GLM lightning feature, the model relied only on ABI-derived cloud-top information and was more likely to over-detect precipitation from cold or extensive cloud regions that did not correspond to rainfall at the surface.

To highlight the precipitation retrieval performance for different rainfall rates, seven thresholds were applied to calculate the evaluation metrics, including 1, 2, 4, 7, 10, 15, and 20 mm/hr. These metrics versus thresholds were plotted in Figure 3.4. It can be seen that, in general, the CNN-based retrieval approach produced higher performance scores compared to the operational GOES-16 RRQPE product. From Figure 3.4 (a)-(c) and (e)-(f), we can observe that when the rainfall rate thresholds increased from 10 to 20 mm/hr, the detection performance of all models dropped and the estimation error increased, which implied that the precipitation retrieval for heavy rain was more challenging than the retrieval for weak and moderate rain. This is because heavy precipitation is often associated with deep convection, stronger spatial variability, and faster temporal evolution. These characteristics make the relationship between satellite-observed cloud-top features and surface rainfall more nonlinear and uncertain than for weak or moderate precipitation. Similarly, Figure 3.4 (e)-(f) reveals that an increase in the rainfall rate thresholds from 1 to 20 mm/hr led to a decline in estimation performance for all models. This was expected as higher rain rates are typically associated with more intense convective features that are often transient and more challenging to retrieve from a satellite remote sensing perspective. Notice that the normalized errors in Figure 3.4 (g)-(h) dropped as the thresholds increased because of the normalization

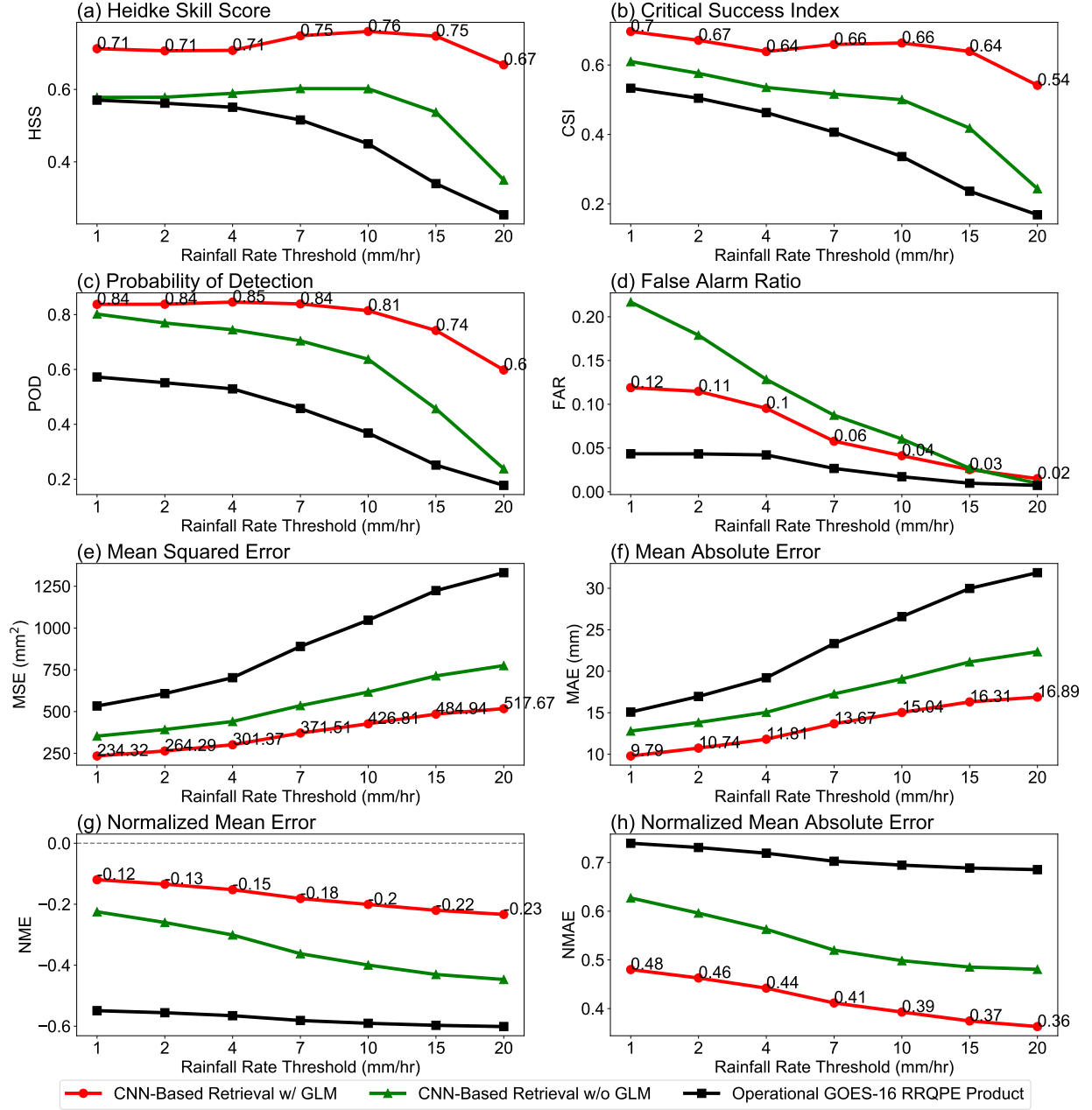


Figure 3.4: The quantitative evaluation scores of the operational GOES-16 RRQPE product and DL-based retrievals with and without using GLM feature. The evaluation scores are calculated for hourly rainfall accumulations based on all the test data, using different thresholds: (a) HSS, (b) CSI, (c) POD, (d) FAR, (e) MSE, (f) MAE, (g) NME, (h) NMAE. The evaluation scores of the CNN-based retrievals with GLM data are marked along the red curves.

in the calculation. It is remarkable that the CNN-based retrieval model with GLM exhibited a more reliable performance than both the CNN-based model without GLM and the operational product, particularly for heavy precipitation events where the hourly rainfall rate exceeded 10 mm/hr. This

conclusion is consistent with the visual example illustrated in Figure 3.3 and the FDA investigation in Figure 2.1.

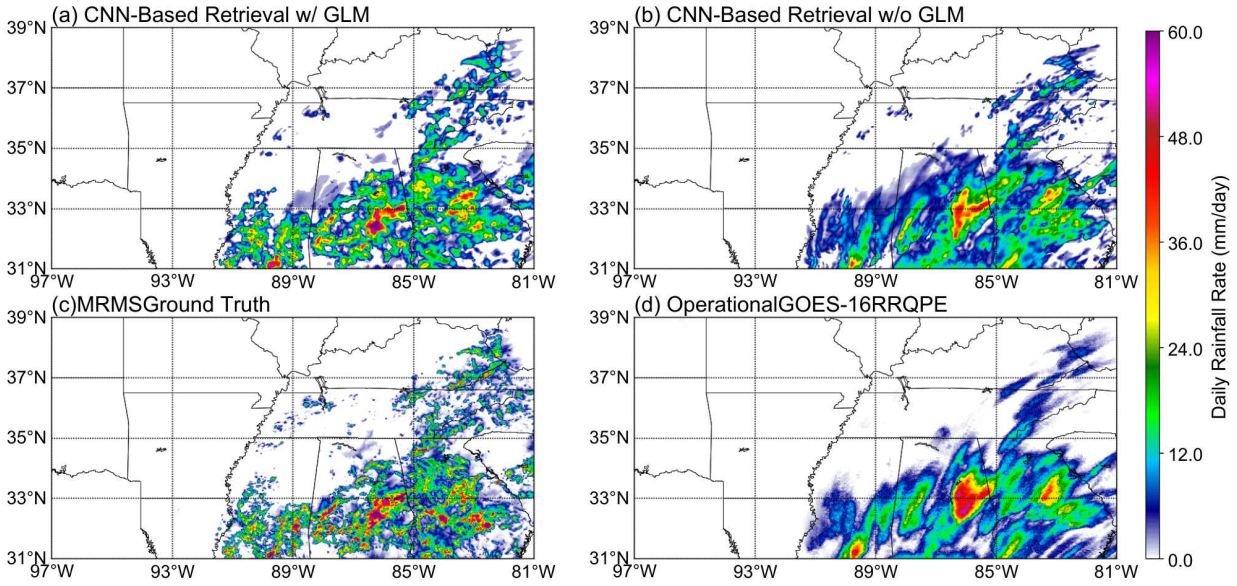


Figure 3.5: Example of daily precipitation accumulations on August 20, 2020: (a) CNN-based retrieval with GLM data; (b) CNN-based retrieval without GLM data; (c) ground-based MRMS product; (d) operational GOES-16 RRQPE product.

The daily precipitation retrieval results were obtained by aggregating the precipitation estimates over 24 hours of a day. An example of the precipitation estimates on August 20, 2020, is presented in Figure 3.5, including the CNN-based precipitation retrieval with and without the hourly flash rate feature from GLM measurement, the MRMS ground truth, and the currently operational GOES-16 RRQPE product. In Figure 3.5 (c), an area of precipitation with a number of embedded convective cells can be observed in the southeastern part of the study domain. Most of these rainfall patterns were revealed by the CNN-based retrieval with the lightning data in Figure 3.5 (a), albeit with some underestimation. Conversely, these heavy precipitation areas were largely missed by the CNN-based retrieval without the lightning data and the operational product, leading to substantial inaccuracy and underestimation. In contrast, for the moderate precipitation in the northeast part of the study domain, both of the CNN-based retrievals were able to delineate the precipitation areas more accurately and their intensities compared to the operational product. In

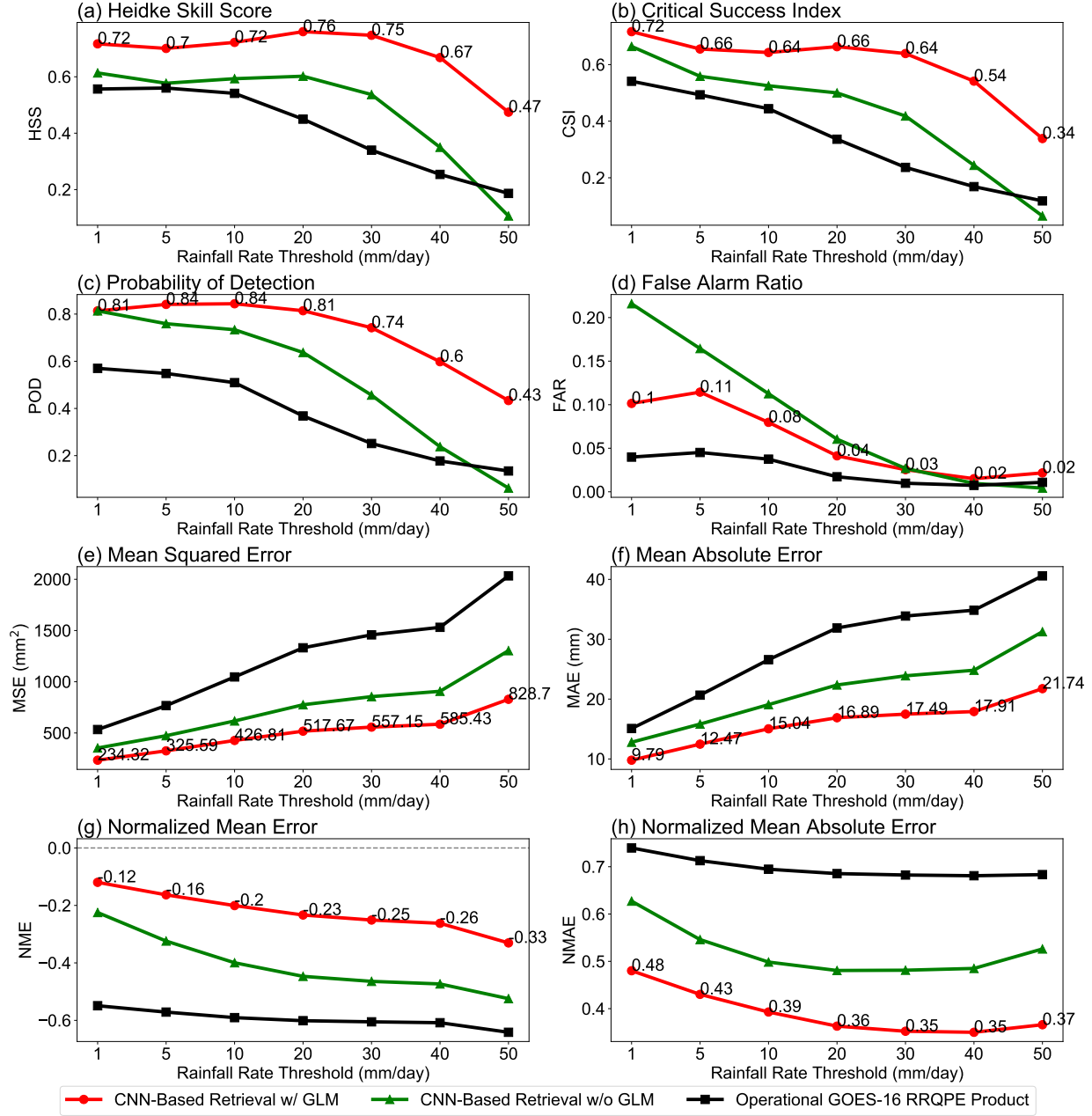


Figure 3.6: The quantitative evaluation scores of the operational GOES-16 RRQPE product and DL-based retrievals with and without using lightning information. The evaluation scores are calculated for daily rainfall accumulations based on all the test data, using different thresholds: (a) HSS, (b) CSI, (c) POD, (d) FAR, (e) MSE, (f) MAE, (g) NME, (h) NMAE. The evaluation scores of the CNN-based retrievals with GLM data are marked along the red curves.

Figure 3.5 (b), the estimates without using GLM measurement seemed to have a coarser resolution than the estimates in Figure 3.5 (a). Since the CNN-based retrieval model without GLM exhibits a relatively high false alarm rate for weak to moderate precipitation, after aggregating the estimates

to a daily scale, the delineation of areas with weak to moderate precipitation appeared less distinct compared to the estimates obtained from the CNN-based retrieval model without GLM model, making the estimates look smoother.

The daily precipitation evaluation with thresholds was carried out as follows. We first obtained the daily precipitation estimates of the days in the testing months in 2020. Note that we completely skipped the days with abnormal/missing data in ABI or GLM measurements, as well as the days without recorded precipitation events within the chosen study domain. We then calculated the daily precipitation evaluation metrics for each remaining day with thresholds 1, 5, 10, 20, 30, 40, and 50 mm/day. Finally, for each threshold, the daily precipitation evaluation metrics were averaged over the remaining days. The averaged daily precipitation evaluation metrics were plotted in Figure 3.6. It can be seen from Figure 3.6 that the CNN-based retrieval approach performed better than the operational GOES-16 RRQPE product with lower errors. For the more challenging heavy precipitation retrieval, the CNN-based retrieval with the lightning data exhibited large improvements when compared to the CNN-based retrieval without the GLM lightning data and the operational RRQPE product. Although the operational RRQPE product had the lowest FAR as shown in Figure 3.6 (d), the FAR of the CNN-based retrieval with GLM still maintained reasonable values. In Figure 3.6 (g), the negative values of NMEs imply that all three models underestimated the precipitation overall, with the CNN-based retrieval with GLM lightning data demonstrating the lowest NME. These observations are consistent with the underestimation of precipitations in Figure 3.5 using the CNN-based retrieval. Furthermore, our comparison of the CNN-based retrieval with and without GLM highlights the significance of the lightning data for precipitation retrieval, particularly as the rainfall rates increase.

3.5 Discussion and Analysis

In this section, we analyze the performance of the proposed DL framework for satellite-based precipitation estimation. The results are examined from several perspectives to better understand the strengths and limitations of the model. In particular, we investigate the contributions of differ-

ent input features, assess the impact of observational uncertainties, and analyze the behavior of the model under varying precipitation conditions. These analyses provide insights into the factors influencing precipitation retrieval accuracy and help identify potential areas for further improvement. The following subsections present detailed discussions on these aspects.

3.5.1 Comparison with Other DL Methods

The experimental results have demonstrated the efficacy and superiority of the DL model for precipitation retrieval using the ABI and GLM measurements on the GOES-R series satellite, particularly for moderate and heavy precipitation. From these results, one can conclude that the DL-based precipitation retrieval can significantly enhance the currently operational product to improve the precipitation retrieval accuracy, with brightness temperatures from the selected ABI bands and the flash rates from the GLM measurement as the input features. Thus, the proposed DL framework can potentially serve as an alternative approach for GOES precipitation retrievals.

A key design of the proposed framework is its hybrid structure with a CNN detector and a CNN estimator. Instead of directly estimating rainfall rates for all pixels, the framework first identifies whether precipitation is present and then estimates the rainfall intensity only for the detected precipitation pixels. This detector–estimator structure allows the model to separately address precipitation occurrence and precipitation intensity, which are related but different tasks. As a result, the framework can reduce false rainfall estimates over non-precipitating regions while improving rainfall rate estimation over precipitating areas.

In comparison to other DL models [59, 60] with wide usages, e.g., VGG-16 with 13 convolutional layers and 3 dense layers, our proposed DL model only has 3 convolutional layers and 1 dense layer for each CNN module. Consequently, the proposed DL model is lightweight and efficient, providing precipitation estimates without consuming excessive computational time. However, it is important to note that further investigation is necessary to determine the generalization capability of DL models for precipitation retrieval in multiple domains.

3.5.2 Spatial and Temporal Resolution Mismatch

From another perspective, the DL model learns a mapping from the feature space consisting of BTs, BTDs, and flash rates to the precipitation rainfall rate. These BT and BTD features are derived from the observations of the GOES-16 ABI measurement and are on the top or near the clouds with a 2 km spatial resolution. The flashes in the GOES-16 GLM measurement account for all in-cloud, cloud-to-cloud, and cloud-to-ground lightning activities with 8 km resolution. On the other hand, the target labels used for training the DL model are the ground-based MRMS dataset and have a 1 km spatial resolution. In the training phase, the MRMS data are re-sampled to 2 km spatial resolution to provide the target labels for the training voxels. Hence, errors may inevitably arise during sampling and providing the target labels of the training voxels. In spite of the possible errors raised from the different resolutions of datasets, the environments between the clouds and the grounds are variable. Any atmospheric changes may affect the learned mapping, and cause unreliable or inaccurate precipitation estimates.

3.5.3 Parallax Effects in GOES-16 ABI Observations

An additional concern regarding the expansion of the proposed framework pertains to the parallax shift in ABI measurements [61]. That is, the cloud-top features observed by the GOES-R satellite sensors appear to be displaced away from the satellite sub-point. This shift is important because it can create spatial mismatch between satellite-observed cloud-top features and the ground-based rainfall labels used for training and evaluation. If not corrected, the model may learn from misaligned inputs and targets, which can reduce precipitation detection accuracy and introduce location errors in the retrieved rainfall fields. The parallax shift becomes more noticeable as the study domain moves closer toward the edges of the ABI’s field of view. For the study domain considered here, the parallax effects were less significant compared to other areas, such as the western United States, due to its proximity to the satellite sub-point. Furthermore, it should be acknowledged that one of the advantages of CNNs lies in that they are less sensitive to spatial shift, which prompted us to overlook the parallax effect in our study. Nevertheless, it is important

to note that this parallax shift will pose a more substantial challenge when extending our work to encompass the western United States or the entire western hemisphere.

3.5.4 Data Availability and Measurement Limitations

Despite the missing or invalid data in GOES-R ABI and GLM measurements, the reliability and accuracy of the targets of the training data, i.e., ground-based MRMS product, are crucial when training a DL model for precipitation retrieval. However, as presented in [11], the coverage of the MRMS dataset is mostly limited to CONUS. Even so, the deployment of the ground radars over some complex terrains may not be as sufficient as those in plain areas which have relatively flat terrain. These complex areas include the regions where the deployment of radars is challenging due to various environmental obstacles such as mountains, oceans, and dense vegetation. For offering a global precipitation retrieval product, these complex terrains are unavoidable and further study on DL-based precipitation over complex terrains needs to be considered.

3.5.5 Additional Features

In this work, we only included BT, BTD, and lightning as input features for precipitation retrieval. Other environmental factors, such as terrain information, water vapor, temperature, and wind magnitude/direction, also impact the precipitation and therefore can be incorporated with the BT, BTD, and lightning features for a potentially more accurate precipitation retrieval. One way to include these additional features without starting the training process over is the network expansion for DL neural networks [62, 63]. For example, the terrain and elevation information has been shown to be important factors for the satellite-based precipitation retrieval in [29, 64, 65].

3.5.6 Effect of Lightning Data

From the experimental results, it can be seen that after including the hourly flash rate feature from the GOES-16 GLM measurement as an additional input to the DL model, the precipitation retrieval performance was improved. This finding supports the inference on the GLM lightning

data using FDA in Section 2.5, highlighting the crucial role of lightning information in precipitation retrievals.

However, here, only the flash rate from the GLM measurement was used, regardless of the energy released from the flashes. One main reason we did not use the released energy directly as the feature is that it would not provide clear information about the absence of lightning. If we used the hourly released energy from the flashes as a feature, we could not justify whether a small number of flashes occurred within a short period of time but released massive of energy intensely, or a large number of flashes occurred mildly within an hour. In contrast, the hourly flash rate can explicitly reflect the occurrence of the flashes, which usually take place with the precipitations. This also avoids possible errors in estimating the released energy from the flashes. Other than that, the GLM measurement provides three products, event, group, and flash, while in this work, we selected the most integrated flash product. A supplemental study on how each of them impacts the DL-based precipitation retrieval could be conducted thoroughly to further improve the retrieval performance. Lastly, in the data pre-processing for the GLM measurement, the grids with a 2 km spatial resolution were formed, while the spatial resolution for the GLM measurement is 8 km. This spatial resolution mismatch can cause inaccurate flash rate features.

3.6 Multi-Domain Limitations

It is anticipated that a satellite-based precipitation estimation product has a global or at least hemispheric coverage, while the study domain in this chapter was limited to only a region in the southeastern United States. When we expand the study domain to the other regions, further studies are required to justify whether multiple DL models or only one DL model is adequate to offer accurate hemispheric coverage of rainfall rate estimates. Although the DL model demonstrates promising performance for precipitation estimation within a single domain, several limitations arise when the model is applied across multiple domains. Satellite observations and precipitation characteristics can vary significantly across different regions, seasons, and meteorological condi-

tions. A model trained on data from one domain often experiences performance degradation when applied to another domain due to distribution shifts in the input features and precipitation patterns.

Furthermore, retraining the model with data from new domains using conventional training strategies may lead to the problem of catastrophic forgetting [66, 67], where the model loses previously learned knowledge from the earlier domains. This limitation restricts the scalability of traditional DL approaches when sequentially incorporating new datasets or adapting to new environments.

These challenges highlight the need for learning frameworks that can effectively adapt to new domains while preserving knowledge from previously learned domains. Continual learning (CL) provides a promising solution to address this issue by enabling models to incrementally learn from sequential data without suffering from catastrophic forgetting, for which the DL model can maintain the performance on the previous precipitation patterns but also be adaptive to the newly unseen patterns. Also, it is no longer a restrictive requirement that the training data must contain all intensities of precipitation data. The following chapter introduces a CL framework designed to support multi-domain precipitation estimation.

3.7 Conclusion

This chapter presented a DL framework for satellite-based precipitation estimation without the use of CL. Using GOES satellite observations as inputs and MRMS precipitation data as ground reference, a neural network model was developed to learn the nonlinear relationship between satellite measurements and surface precipitation. The model architecture, training procedure, and study domain were described, and baseline experiments demonstrated that the DL approach can effectively estimate precipitation from geostationary satellite observations.

Despite the promising performance achieved within a single domain, several limitations remain when extending the model to multi-domain scenarios. Variations in meteorological conditions, geographic characteristics, and seasonal patterns across different domains introduce distribution shifts in the input data. As a result, a model trained on one domain may experience significant

performance degradation when applied to new domains. Moreover, sequentially retraining the model with data from new domains using conventional training strategies can lead to catastrophic forgetting, where previously learned knowledge is overwritten.

These limitations highlight the challenges of applying standard DL approaches to large-scale, multi-domain precipitation estimation problems. To address these challenges, a CL framework is needed that can adapt to new domains while preserving knowledge learned from previous domains. Next chapter, we will introduce the concept of CL and design novel CL methods to support multi-domain precipitation estimation and mitigate catastrophic forgetting.

Chapter 4

A Review of Continual Learning

4.1 Introduction

Recent advances in DL have improved precipitation estimation. However, as discussed in previous chapters, DL models often struggle to generalize across multiple domains with varying data distributions. In practical applications, precipitation characteristics differ significantly across geographic regions, climatic regimes, seasons, and storm types. Therefore, a practical precipitation retrieval model should be able to adapt as new data and new domains become available. Conventional training strategies typically assume a fixed dataset and a single training session. This assumption is not well suited for multi-domain precipitation retrieval, where new regions or precipitation regimes may need to be incorporated sequentially. Directly updating a trained model with new data can lead to performance degradation on previously learned domains. This problem is known as catastrophic forgetting, where neural networks overwrite previously learned representations when trained on new tasks [37, 38].

CL provides a principled learning paradigm to address this issue. It enables models to learn incrementally from new data with potentially different distributions while preserving knowledge acquired from previous tasks. This capability has become increasingly important in real-world applications where data distributions evolve over time, including robotics [68, 69], autonomous systems [70, 71], healthcare [72, 73], and remote sensing [74, 75]. In these contexts, the ability to retain and build upon accumulated knowledge is crucial for developing intelligent systems capable of long-term operation in non-stationary environments.

The main challenge in CL is balancing plasticity and stability. Plasticity refers to the ability of a model to acquire new knowledge, while stability refers to the ability to retain previously learned knowledge. If a model is too plastic, it may quickly adapt to new tasks but forget previous ones. If it is too stable, it may preserve prior knowledge but fail to learn new information effec-

tively. Effective management of this stability–plasticity trade-off is essential for building robust CL systems [76]. For satellite precipitation retrieval, this challenge is particularly important because different geographic domains exhibit distinct precipitation characteristics, while a scalable retrieval system must adapt to new regions without sacrificing performance in previously learned regions.

This chapter briefly reviews the key concepts and representative methods of CL, including catastrophic forgetting, the stability–plasticity trade-off, and several common approaches for mitigating forgetting. The limitations of existing CL methods are then discussed to motivate the development of new algorithms for multi-domain satellite precipitation estimation.

4.2 Brief Review of Key Continual Learning Methods

This section reviews representative CL methods that aim to mitigate catastrophic forgetting while providing promising performance over the previous tasks. Existing approaches can be broadly categorized into three classes: regularization-based methods [39, 77, 78], replay-based methods [79–81], and projection-based methods [43, 44, 82]. These categories differ in how they preserve previously learned knowledge while adapting to new tasks. Regularization-based methods constrain parameter updates using importance measures, replay-based methods retain past information through stored or generated samples, and projection-based methods restrict updates to directions that do not interfere with prior knowledge. In the following subsections, we briefly review representative methods from each category.

4.2.1 Regularization-based Methods

(a) Elastic Weight Consolidation

Elastic Weight Consolidation (EWC) [39] is one of the earliest regularization-based CL methods. The main idea of EWC is to introduce a penalty term to the loss function for new tasks that prevents drastic updates to important model parameters corresponding to the previously learned

tasks. These important parameters are those that strongly influence the performance on earlier tasks, and large changes to them may lead to catastrophic forgetting.

Let us consider two tasks: A and B. After the model is trained on task A, the optimal parameter value is denoted by θ_A^* . When learning task B, EWC modifies the loss function by adding a quadratic regularization term:

$$L(\theta) = L_B(\theta) + \sum_i \frac{\lambda}{2} F_i (\theta_i - \theta_{A,i}^*)^2, \quad (4.1)$$

where θ represents the model parameters; $L_B(\theta)$ is the new loss function associated with task B (e.g., SSE loss); $\theta_{A,i}^*$ denotes the optimal value of the i th parameter after training on task A; λ is a hyperparameter that controls the strength of the regularization term; and F_i is the i th diagonal element of the estimated Fisher information matrix [83, 84] associated with parameter θ_i . The Fisher information measures the importance of each parameter to task A. Therefore, parameters with larger F_i values are more strongly constrained during training on task B. The Fisher information F is defined as

$$F_i = \mathbb{E} \left[\left(\frac{\partial \log(p_{\theta}(y | \mathbf{x}))}{\partial \theta} \right) \left(\frac{\partial \log(p_{\theta}(y | \mathbf{x}))}{\partial \theta} \right)^{\top} \right]_{i,i}, \quad (4.2)$$

where $\mathbb{E}[\cdot]$ is the expectation operation; $p_{\theta}(\mathbf{y} | \mathbf{x})$ is the conditional likelihood parametrized by θ ; $\mathbf{y}$ and $\mathbf{x}$ are the input and output to the DL model, respectively; $[\cdot]_{i,i}$ represents the i th diagonal element of the matrix. In this work, the Fisher information matrix is empirically approximated. More detailed calculation for the Fisher information matrix can be found in [84].

Although EWC is simple and effective in many settings, its performance may degrade as the number of tasks increases. The accumulated regularization constraints can reduce the model's ability to adapt to new tasks, especially when the new task requires parameter updates that conflict with those important for previous tasks.

(b) Riemannian Walk

Riemannian Walk (RWalk) [85] extends EWC by incorporating both the Fisher information and a path-based importance measure accumulated during training. In RWalk, the path-based importance measure records how much the movement of a parameter affects the training objective function by accumulating the contribution of each parameter to the loss reduction during the optimization. Parameters that produce large decreases in the loss function with relatively small changes are considered important and are penalized more strongly during future tasks. In this way, RWalk complements the Fisher-information-based importance measure with an optimization-path-based measure that reflects the parameter’s contribution during training.

Let us consider sequential learning over two tasks. After completing previous task A, the importance of each parameter θ_i is quantified by combining Fisher information F_i and a path-based score S_i . When learning a new task B, the loss function is defined as

$$L(\boldsymbol{\theta}) = L_B(\boldsymbol{\theta}) + \lambda \sum_i (F_i + S_i)(\theta_i - \theta_{A,i}^*)^2, \quad (4.3)$$

where $L_B(\boldsymbol{\theta})$ is the loss for the current task B, $\theta_{A,i}^*$ denotes the parameter after previous task A, F_i represents the Fisher information (which can be calculated similarly to EWC), S_i is the path-integral importance score, and λ controls the regularization strength. Suppose the model parameter is at training iteration t , the change of the i th parameter between two consecutive updates is defined as

$$\Delta\theta_i(t) = \theta_i(t+1) - \theta_i(t). \quad (4.4)$$

Using a first-order approximation, the contribution of parameter θ_i to the loss reduction at iteration t can be written as

$$\Delta L_t(\theta_i) = -\frac{\partial L(\boldsymbol{\theta})}{\partial \theta_i} \Delta\theta_i(t). \quad (4.5)$$

Here, a positive value of $\Delta L_t(\theta_i)$ indicates that the movement of parameter θ_i contributes to reducing the loss. Then, the path-based importance score for parameter θ_i accumulated from iteration t_1

to t_2 is computed as

$$S_i = \sum_{t=t_1}^{t_2} \frac{\Delta L_t(\theta_i)}{\frac{1}{2} F_i^t (\Delta \theta_i(t))^2 + \epsilon}, \quad (4.6)$$

where F_i^t is the diagonal Fisher information associated with parameter θ_i at iteration t , and $\epsilon > 0$ is a small constant used for numerical stability. Negative values of S_i are set to zero because only parameter movements that positively contribute to reducing the loss are considered important as: $S_i \leftarrow \max(S_i, 0)$.

Thus, a parameter receives a large S_i value if it produces a large decrease in the loss. Compared with EWC, RWalk considers not only the parameter importance after training is finished, but also how each parameter contributed during the training process. In other words, it tracks the learning behavior of each parameter along the optimization path, including whether its updates helped reduce the loss. However, it introduces additional computational overhead and still suffers from accumulated regularization constraints as the number of tasks increases.

4.2.2 Replay-based Methods

Replay-based methods mitigate catastrophic forgetting by retaining information from previous tasks and reusing it when learning new tasks. This retained information can be stored as real samples, generated samples, or representative memory examples. By replaying previous-task information during training, the model is reminded of earlier tasks while adapting to the current task.

For example, Experience Replay (ER) is a simple yet effective method that mitigates forgetting by storing samples from previous tasks and replaying them during the re-training. The main idea is to approximate joint training over all tasks by combining new data with stored past samples. Let us consider two tasks: task A and task B . After the model is trained on task A , a subset of training samples from task A is stored in a memory buffer, denoted as $\mathcal{M}_A$. When the model is later trained on task B , the training data from task B are combined with the replayed samples from $\mathcal{M}_A$. In this way, the model continues to learn the new task while still being reminded of the previous task.

The loss function for learning task B with experience replay can be written as

$$L(\boldsymbol{\theta}) = L_B(\boldsymbol{\theta}) + \lambda L_A^{\text{replay}}(\boldsymbol{\theta}), \quad (4.7)$$

where $L_B(\boldsymbol{\theta})$ is the loss computed using the training samples from task B , $L_A^{\text{replay}}(\boldsymbol{\theta})$ is the replay loss computed using the stored samples from task A , and λ is a hyperparameter that balances learning the new task and preserving the previous task.

The specific form of the loss depends on the learning problem. For example, in a classification problem, both $L_B(\boldsymbol{\theta})$ and $L_A^{\text{replay}}(\boldsymbol{\theta})$ can be defined using binary cross-entropy or cross-entropy loss. In a regression problem, such as precipitation rate estimation, they can be defined using mean squared error (MSE).

By including $L_A^{\text{replay}}(\boldsymbol{\theta})$, experience replay discourages the model from changing its parameters in a way that greatly increases the error on task A . Therefore, the model can adapt to task B while reducing forgetting on task A . However, experience replay requires storing past data, which may not be feasible for large-scale or high-dimensional datasets. In addition, storing previous samples may raise memory, privacy, or data-management concerns in some applications.

4.2.3 Projection-based Methods

(a) Orthogonal Gradient Descent

Orthogonal Gradient Descent (OGD) [43] is a projection-based CL method that constrains the current-task gradient to be orthogonal to the gradient subspace of previous task data. This helps reduce interference with previously learned knowledge.

Consider two tasks, task A and task B . After the model is trained on task A , representative samples from task A are used to compute gradients with respect to the model parameters. For a sample $\mathbf{x}_{A,j}$ from task A , the gradient vector can be written as $\mathbf{q}_{A,j} = \nabla_{\boldsymbol{\theta}} f(\mathbf{x}_{A,j}; \boldsymbol{\theta}_A^*)$, where $f(\mathbf{x}_{A,j}; \boldsymbol{\theta}_A^*)$ is the model output for sample $\mathbf{x}_{A,j}$, and $\boldsymbol{\theta}_A^*$ denotes the model parameters after learning task A .

These gradients describe how the previous-task outputs would change if the model parameters were updated. Therefore, they define important directions for preserving task A . The subspace

spanned by these gradients is given by $\mathcal{S}_A = \text{span} \{\mathbf{q}_{A,1}, \mathbf{q}_{A,2}, \dots, \mathbf{q}_{A,n_A}\}$, where n_A is the number of representative samples from task A .

To perform the projection efficiently, an orthonormal basis of this subspace is constructed. Let $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_K]$ denote the orthonormal basis of $\mathcal{S}_A$, where the basis vectors can be obtained using Gram–Schmidt orthogonalization, QR decomposition, or singular value decomposition (SVD). Since the columns of $\mathbf{U}$ are orthonormal, $\mathbf{U}^T \mathbf{U} = \mathbf{I}$. When learning task B , the ordinary gradient of the current-task loss is

$$\mathbf{g}_B = \nabla_{\boldsymbol{\theta}} L_B(\boldsymbol{\theta}), \quad (4.8)$$

where $L_B(\boldsymbol{\theta})$ is the loss function for task B . For example, $L_B(\boldsymbol{\theta})$ can be cross-entropy for classification or MSE for regression.

If the model is directly updated using $\mathbf{g}_B$, the update may change the outputs learned for task A . Therefore, OGD removes the component of $\mathbf{g}_B$ that lies in the previous-task subspace $\mathcal{S}_A$. The projected gradient is computed as

$$\mathbf{g}_{B,\text{proj}} = \mathbf{g}_B - \mathbf{U} \mathbf{U}^T \mathbf{g}_B. \quad (4.9)$$

Equivalently, this can be written as

$$\mathbf{g}_{B,\text{proj}} = \mathbf{g}_B - \sum_{k=1}^K (\mathbf{g}_B^T \mathbf{u}_k) \mathbf{u}_k. \quad (4.10)$$

The term $\mathbf{U} \mathbf{U}^T \mathbf{g}_B$ represents the component of the current gradient that lies in the previous-task gradient subspace. By subtracting this component, OGD obtains an update direction that is orthogonal to the important directions of task A .

The model parameters are then updated using the projected gradient:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \mathbf{g}_{B,\text{proj}}, \quad (4.11)$$

where η is the learning rate.

In this way, OGD allows the model to learn task B using update directions that have minimal effect on the outputs of task A . However, OGD requires storing and maintaining basis vectors for previous-task gradients. As the number of tasks increases, the gradient subspace may become large, leading to higher memory and computational costs.

(b) Orthogonal Weight Modification

Orthogonal Weight Modification (OWM) [82] is a projection-based CL method that constrains parameter updates using the input subspace of previous tasks. Different from OGD, which uses gradients from previous tasks, OWM focuses on preserving the model responses to previous training samples.

Consider two tasks, task A and task B . During training on task A , the model observes a set of input samples from task A . These samples form an input subspace that contains the main directions of the previous task data. When learning task B , OWM attempts to update the model parameters only in directions that are orthogonal to this previous input subspace. In this way, the model can learn new information from task B while reducing changes to the input-output mapping learned from task A .

OWM constructs a projection matrix $\mathbf{P}$ that projects new updates onto the space orthogonal to the previously input sample subspace. During training, this projection matrix is updated recursively as

$$\mathbf{P}_t = \mathbf{P}_{t-1} - \frac{\mathbf{P}_{t-1} \mathbf{x}_{t,A} \mathbf{x}_{t,A}^T \mathbf{P}_{t-1}}{1 + \mathbf{x}_{t,A}^T \mathbf{P}_{t-1} \mathbf{x}_{t,A}}, \quad (4.12)$$

where $\mathbf{x}_{t,A}$ is the input sample at training step t from task A . This recursive update gradually removes directions associated with previously observed inputs from the projection matrix.

After training on task A , the resulting projection matrix is denoted as $\mathbf{P}_A$. When learning task B , let $\mathbf{g}_B$ denote the gradient of the task B loss with respect to the model parameter $\boldsymbol{\theta}$ as $\mathbf{g}_B = \nabla_{\boldsymbol{\theta}} L_B(\boldsymbol{\theta})$, where $L_B(\boldsymbol{\theta})$ is the loss function for task B . Then, OWM modifies the gradient using the projection matrix $\mathbf{P}_A$ as $\Delta \mathbf{W} = -\alpha \mathbf{g}_B \mathbf{P}_A$, where α is the learning rate.

However, OWM relies on the accurate estimation of the previous input subspace. If the projection matrix does not sufficiently represent the important directions of task A , the model may still forget previous knowledge. In addition, maintaining and updating the projection matrix can introduce memory and computational overhead, especially for high-dimensional input features or large neural networks.

4.3 Conclusion

This chapter reviewed the fundamental concepts and representative methods of CL, with emphasis on catastrophic forgetting and the stability–plasticity trade-off. Representative methods were discussed across regularization-based, replay-based, and projection-based approaches, highlighting how different strategies preserve prior knowledge during sequential learning.

Although these methods have shown strong performance in standard machine learning settings, their application to satellite-based precipitation estimation remains relatively limited. In particular, most existing approaches do not explicitly address the challenges of multi-domain variability, evolving data distributions, and computational scalability in remote sensing applications.

These limitations motivate the development of CL frameworks that are better suited for satellite-based precipitation retrieval. In this dissertation, the objective is to develop methods that can improve knowledge retention while allowing the model to adapt to new precipitation domains.

The following chapter builds on this foundation by introducing new CL algorithms for multi-domain precipitation estimation. The proposed methods leverage the null space of input representations from previous tasks to constrain parameter updates when learning subsequent tasks. Incremental SVD is used to efficiently update the null-space representation as new tasks are introduced. This formulation aims to minimize interference with previously learned knowledge while maintaining adaptability to new domains.

Chapter 5

Continual Learning using Incremental SVD

5.1 Introduction

In CL, a model is trained sequentially on a series of tasks that may have different data distributions [40, 41]. The main objective is to enable the model to learn new tasks while preserving knowledge acquired from previous tasks. A major challenge in this setting is catastrophic forgetting, where learning a new task causes performance degradation on earlier tasks [37, 38, 41]. To address this issue, this chapter introduces a CL algorithm based on incremental SVD, referred to as CL-ISVD. The proposed method incrementally updates the subspace representation as new data become available [86, 87], avoiding the need to recompute the full SVD from scratch [88]. The proposed CL-ISVD method is evaluated on benchmark datasets and compared with representative CL methods, including regularization-based, replay-based, and projection-based approaches [39, 43, 44, 80, 82], to demonstrate its effectiveness.

The remainder of this chapter is organized as follows. Section 5.2 introduces the problem formulation and notation. Section 5.3 derives the null-space condition for reducing catastrophic forgetting. Section 5.4 presents the projected parameter update rule. Section 5.5 describes the incremental SVD procedure. Section 5.6 presents the complete CL-ISVD algorithm. Section 5.7 analyzes computational complexity. Sections 5.8–5.10 present the experimental setup, results, and efficiency analysis. Finally, Section 5.11 concludes this chapter.

5.2 Problem Formulation and Notation

Consider a sequence of tasks $\{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_t\}$. At task $\mathcal{T}_t$, the model is trained using dataset $\{\mathbf{X}_t, \mathbf{Y}_t\}$, where $\mathbf{X}_t \in \mathbb{R}^{M_1 \times P}$ and $\mathbf{Y}_t \in \mathbb{R}^{N_L \times P}$. Here, M_1 is the input dimension, N_L is the number of output neurons, and P is the number of samples.

For a feedforward neural network with L layers, let $\mathbf{W}^l \in \mathbb{R}^{M \times N}$ denote the weight matrix of layer l , defined as

$$\mathbf{W}^l = [\mathbf{w}_1^l, \mathbf{w}_2^l, \dots, \mathbf{w}_M^l]^\top, \quad (5.1)$$

where each row vector

$$\mathbf{w}_m^l = [w_{m,1}^l, w_{m,2}^l, \dots, w_{m,N}^l]^\top \quad (5.2)$$

contains the weights connecting input neuron m to all N output neurons in layer l . For notational simplicity, the dependence of M and N on layer index l is omitted.

The input to layer l for task $\mathcal{T}_t$ is

$$\mathbf{X}_t^l = [\mathbf{x}_1^l, \mathbf{x}_2^l, \dots, \mathbf{x}_P^l], \quad (5.3)$$

where each input sample is given by

$$\mathbf{x}_p^l = [x_{1,p}^l, x_{2,p}^l, \dots, x_{M,p}^l]^\top. \quad (5.4)$$

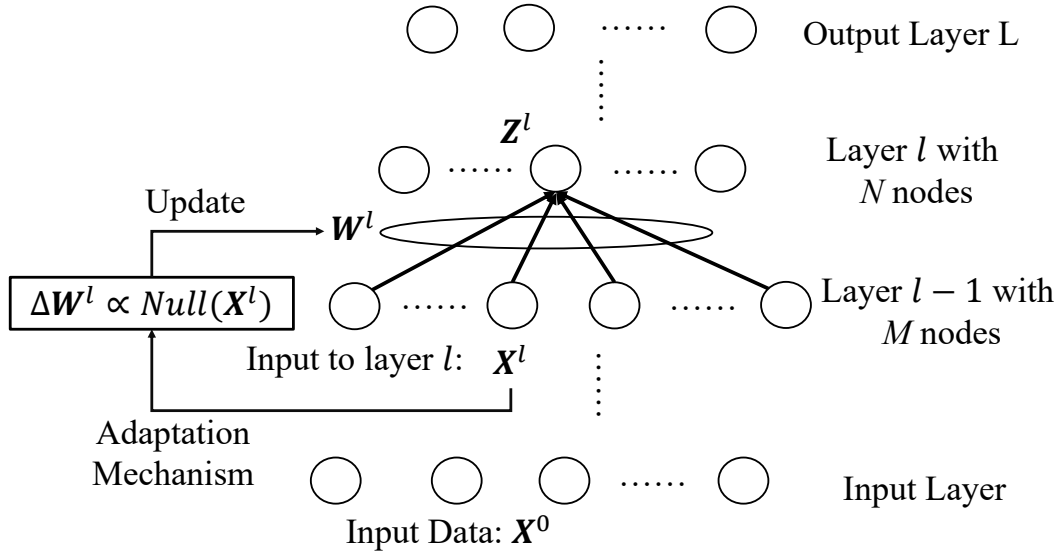


Figure 5.1: Weight updating of layer l for a new task $\mathcal{T}_t$.

Let $\mathbf{O}_t^l \in \mathbb{R}^{N \times P}$ denote the pre-activation outputs, defined as

$$\mathbf{O}_t^l = [\mathbf{o}_1^l, \mathbf{o}_2^l, \dots, \mathbf{o}_P^l]. \quad (5.5)$$

The forward propagation at layer l is given by

$$\mathbf{O}_t^l = \mathbf{W}^{l\top} \mathbf{X}_t^l, \quad (5.6)$$

followed by the activation

$$\mathbf{Z}_t^l = \sigma(\mathbf{O}_t^l), \quad (5.7)$$

where $\sigma(\cdot)$ denotes the activation function, such as ReLU. The output $\mathbf{Z}_t^l$ is then used as the input to the next layer.

5.3 Condition for Learning without Catastrophic Forgetting

A key challenge in CL is avoiding catastrophic forgetting without accessing data from previous tasks. To achieve this, we consider the weight update rule

$$\mathbf{W}^l(k) = \mathbf{W}^l(k-1) - \alpha \Delta \mathbf{W}^l(k), \quad (5.8)$$

where $\Delta \mathbf{W}^l(k)$ is the weight update at iteration k and α is the learning rate. If the update matrix satisfies

$$\mathbf{X}_\tau^{l\top} \Delta \mathbf{W}^l(k) = \mathbf{0}, \quad \forall \tau \in [1, t-1], \quad (5.9)$$

then the updates lie in the null space of the previous task data matrices.

From (5.6), it follows that

$$\mathbf{O}_\tau^l(k) = \mathbf{W}^l(k)^\top \mathbf{X}_\tau^l \quad (5.10)$$

$$= (\mathbf{W}^l(k-1) - \alpha \Delta \mathbf{W}^l(k))^\top \mathbf{X}_\tau^l \quad (5.11)$$

$$= \mathbf{W}^l(k-1)^\top \mathbf{X}_\tau^l - \alpha \Delta \mathbf{W}^l(k)^\top \mathbf{X}_\tau^l \quad (5.12)$$

$$= \mathbf{O}_\tau^l(k-1) - \alpha \Delta \mathbf{W}^l(k)^\top \mathbf{X}_\tau^l \quad (5.13)$$

$$= \mathbf{O}_\tau^l(k-1). \quad (5.14)$$

This implies that the layer outputs for previous tasks remain unchanged. Consequently, no forgetting occurs at this layer.

To represent all previous tasks jointly, let us define the aggregated matrix

$$\mathbf{A}_t^l = [\mathbf{X}_1^l, \mathbf{X}_2^l, \dots, \mathbf{X}_{t-1}^l] \in \mathbb{R}^{M \times Q}, \quad (5.15)$$

where $Q = (t-1)P$. Clearly, the null-space condition then becomes

$$\mathbf{A}_t^{l\top} \Delta \mathbf{W}^l(k) = \mathbf{0}. \quad (5.16)$$

5.4 Null-Space Projected Parameter Updates

As discussed above, the proposed parameter update requires the null space of the aggregated input representation matrix from previous tasks. For layer l , this matrix is denoted as $\mathbf{A}_t^l \in \mathbb{R}^{M \times Q}$, where M is the number of input neurons in layer l and Q is the total number of accumulated samples from previous tasks. To estimate the null space, we first consider the SVD of $\mathbf{A}_t^l$:

$$\mathbf{A}_t^l = \mathbf{U}_t^l \mathbf{\Sigma}_t^l \mathbf{V}_t^{l\top}, \quad (5.17)$$

where $\mathbf{U}_t^l$ contains the left singular vectors, Σ_t^l contains the singular values, and $\mathbf{V}_t^l$ contains the right singular vectors.

Since the projected update requires the null space of $\mathbf{A}_t^{l\top}$, this null space can be obtained from the left singular vectors of $\mathbf{A}_t^l$. If the effective rank of $\mathbf{A}_t^l$ is r , then the null space of $\mathbf{A}_t^{l\top}$ is spanned by the left singular vectors corresponding to the remaining small singular values:

$$\text{Null}(\mathbf{A}_t^{l\top}) = \text{span}\{\mathbf{u}_{r+1,t}^l, \mathbf{u}_{r+2,t}^l, \dots, \mathbf{u}_{M,t}^l\}. \quad (5.18)$$

In practice, the effective rank r is determined using the cumulative singular value ratio

$$\Gamma(r) = \frac{\sum_{m=1}^r \sigma_m^l}{\sum_{m=1}^M \sigma_m^l}, \quad (5.19)$$

where σ_m^l denotes the m th singular value of $\mathbf{A}_t^l$. The rank r is selected as the smallest integer satisfying

$$\Gamma(r) \geq \eta, \quad (5.20)$$

where η is a predefined threshold. The null-space basis is then formed as

$$\mathbf{U}_{\text{null},t}^l = [\mathbf{u}_{r+1,t}^l, \mathbf{u}_{r+2,t}^l, \dots, \mathbf{u}_{M,t}^l]. \quad (5.21)$$

Based on the above condition, parameter updates can be constrained by projecting gradients onto the null space of $\mathbf{A}_t^{l\top}$. Let

$$\mathbf{P}_{\text{null},t}^l = \mathbf{U}_{\text{null},t}^l \mathbf{U}_{\text{null},t}^{l\top} \quad (5.22)$$

denote the projection matrix onto the null space, where $\mathbf{U}_{\text{null},t}^l$ contains the basis vectors spanning the null space of $\mathbf{A}_t^{l\top}$.

The projected update rule becomes

$$\mathbf{W}^l(k) = \mathbf{W}^l(k-1) - \alpha \mathbf{P}_{\text{null},t}^l \mathbf{G}^l(k), \quad (5.23)$$

where $\mathbf{G}^l(k)$ is the gradient obtained via backpropagation at iteration k .

This formulation naturally balances stability and plasticity. The projection enforces stability by preventing interference with previous tasks, while the gradient term allows adaptation to new tasks. Therefore, the model can learn new information while reducing changes to the representations learned from previous tasks. However, directly recomputing the SVD of $\mathbf{A}_t^l$ becomes inefficient as the number of tasks increases, because the matrix grows whenever new task data are added. Therefore, an incremental SVD procedure is used to update the subspace representation efficiently without recomputing the full SVD from scratch.

5.5 Incremental SVD for Null-Space Estimation

Assume that the data matrix $\mathbf{X}_{t+1}^l \in \mathbb{R}^{M \times P}$ of the new task $\mathcal{T}_{t+1}$ for layer l has become available and we want to incrementally update the SVD of the augmented data matrix this layer $\mathbf{A}_{t+1}^l = [\mathbf{A}_t^l \mid \mathbf{X}_{t+1}^l] \in \mathbb{R}^{M \times (Q+P)}$ to find the null space $\mathbf{U}_{null,t+1}^l$.

Let us define the residual matrix $\mathbf{E}_t^l \in \mathbb{R}^{M \times P}$ which is the projection of the data matrix $\mathbf{X}_t^l$ onto the orthogonal complement of the space spanned by the columns of $\mathbf{U}_t^l$, i.e.,

$$\mathbf{E}_t^l = \mathbf{P}_t^{l\perp} \mathbf{X}_t^l, \quad (5.24)$$

where $\mathbf{P}_t^{l\perp} = \mathbf{I}_M - \mathbf{P}_t^l = \mathbf{I}_M - \mathbf{U}_t^l \mathbf{U}_t^{l\top}$, and $\mathbf{P}_t^l$ is the orthogonal projection matrix of the space spanned by the columns of $\mathbf{U}_t^l$ while $\mathbf{P}_t^{l\perp}$ is the orthogonal complement of it. Then, $\mathbf{E}_t^l$ is decomposed using the QR decomposition method [88] as $\mathbf{E}_t^l = \mathbf{Q}_t^l \mathbf{R}_t^l$ where $\mathbf{Q}_t^l \in \mathbb{R}^{M \times M}$ is an orthogonal matrix with $\mathbf{Q}_t^l \mathbf{Q}_t^{l\top} = \mathbf{I}_M$ and $\mathbf{R}_t^l \in \mathbb{R}^{M \times P}$ is an upper triangular matrix. Then, the

augmented matrix $\mathbf{A}_{t+1}^l$ can be expressed as

$$\mathbf{A}_{t+1}^l = [\mathbf{A}_t^l \mid \mathbf{X}_t^l] = [\mathbf{U}_t^l \Sigma_t^l \mathbf{V}_t^{l\top} \mid \mathbf{X}_t^l] = [\mathbf{U}_t^l \mid \mathbf{Q}_t^l] \underbrace{\begin{bmatrix} \Sigma_t^l & \mathbf{U}_t^{l\top} \mathbf{X}_t^l \\ \mathbf{0}_{M \times M} & \mathbf{R}_t^l \end{bmatrix}}_{\mathbf{S}_t^l} \begin{bmatrix} \mathbf{V}_t^l & \mathbf{0}_{Q \times P} \\ \mathbf{0}_{P \times M} & \mathbf{I}_P \end{bmatrix}^\top. \quad (5.25)$$

Now, using the reduced SVD of $\mathbf{S}_t^l = \mathbf{U}_S^l \Sigma_S^l \mathbf{V}_S^{l\top}$ where $\mathbf{U}_S^l \in \mathbb{R}^{2M \times M}$, $\Sigma_S^l \in \mathbb{R}^{M \times M}$, and $\mathbf{V}_S^l \in \mathbb{R}^{(M+P) \times M}$, Equation (5.25) can alternatively be expressed as

$$\mathbf{A}_{t+1}^l = [\mathbf{A}_t^l \mid \mathbf{X}_t^l] = ([\mathbf{U}_t^l \mid \mathbf{Q}_t^l] \mathbf{U}_S^l) \Sigma_S^l \cdot \left(\begin{bmatrix} \mathbf{V}_t^l & \mathbf{0}_{Q \times P} \\ \mathbf{0}_{P \times M} & \mathbf{I}_P \end{bmatrix} \mathbf{V}_S^l \right)^\top, \quad (5.26)$$

where it can be easily verified that $\mathbf{U}_{t+1}^l := [\mathbf{U}_t^l \mid \mathbf{Q}_t^l] \mathbf{U}_S^l$ is an orthogonal matrix whose columns are the left singular vectors of $\mathbf{A}_{t+1}^l$. Since the matrices $\mathbf{U}_t^l$, $\mathbf{Q}_t^l$, and $\mathbf{U}_S^l$ are individually orthogonal, it suffices to show that $\mathbf{U}_t^{l\top} \mathbf{Q}_t^l = \mathbf{0}$. Now, since $P \gg M$, $\mathbf{E}_t^l$ is full rank, making $\mathbf{R}_t^l$ also full rank. Consequently $(\mathbf{R}_t^l \mathbf{R}_t^{l\top})$ is non-singular and we can have

$$\mathbf{E}_t^l [\mathbf{R}_t^{l\top} (\mathbf{R}_t^l \mathbf{R}_t^{l\top})^{-1}] = \mathbf{Q}_t^l \mathbf{R}_t^l [\mathbf{R}_t^{l\top} (\mathbf{R}_t^l \mathbf{R}_t^{l\top})^{-1}] = \mathbf{Q}_t^l. \quad (5.27)$$

But because $\mathbf{U}_t^{l\top} \mathbf{E}_t^l = \mathbf{0}$, then $\mathbf{U}_t^{l\top} \mathbf{Q}_t^l = \mathbf{0}$. With this property, Σ_S^l is a diagonal matrix containing

the singular values of $\mathbf{A}_{t+1}^l$, and $\begin{bmatrix} \mathbf{V}_t^l & \mathbf{0}_{Q \times P} \\ \mathbf{0}_{P \times Q} & \mathbf{I}_P \end{bmatrix} \mathbf{V}_S^l$ is an orthogonal matrix whose columns are

the right singular vectors of $\mathbf{A}_{t+1}^l$. This result leads to the following incremental SVD process

$$\begin{aligned} \mathbf{U}_{t+1}^l &= [\mathbf{U}_t^l \mid \mathbf{Q}_t^l] \mathbf{U}_S^l \\ \Sigma_{t+1}^l &= \Sigma_S^l \\ \mathbf{V}_{t+1}^l &= \begin{bmatrix} \mathbf{V}_t^l & \mathbf{0}_{Q \times P} \\ \mathbf{0}_{P \times Q} & \mathbf{I}_P \end{bmatrix} \mathbf{V}_S^l. \end{aligned} \tag{5.28}$$

Algorithm 1 Incremental SVD for Continual Learning

Require: Previous SVD $\mathbf{A}_t^l = \mathbf{U}_t^l \Sigma_t^l \mathbf{V}_t^{l\top}$, new data $\mathbf{X}_t^l \in \mathbb{R}^{M \times P}$

1: Compute projection residual

$$\mathbf{E}_t^l = (\mathbf{I}_M - \mathbf{U}_t^l \mathbf{U}_t^{l\top}) \mathbf{X}_t^l$$

2: QR decomposition: $\mathbf{E}_t^l = \mathbf{Q}_t^l \mathbf{R}_t^l$

3: Form small matrix

$$\mathbf{S}_t^l = \begin{bmatrix} \Sigma_t^l & \mathbf{U}_t^{l\top} \mathbf{X}_t^l \\ \mathbf{0} & \mathbf{R}_t^l \end{bmatrix}$$

4: Compute reduced SVD: $\mathbf{S}_t^l = \mathbf{U}_S^l \Sigma_S^l \mathbf{V}_S^{l\top}$

5: Update left singular vectors

$$\mathbf{U}_{t+1}^l = [\mathbf{U}_t^l \mid \mathbf{Q}_t^l] \mathbf{U}_S^l$$

6: Update singular values: $\Sigma_{t+1}^l = \Sigma_S^l$

The above incremental SVD (ISVD) procedure is summarized in Algorithm 1 shown below.

5.6 CL-ISVD Algorithm

Building upon the CL formulation, this section presents the proposed CL algorithm based on incremental singular value decomposition (CL-ISVD). The key idea of the method is to maintain a compact representation of the input subspace associated with previously learned tasks using incremental SVD. This subspace representation enables efficient computation of the null space required for the projected gradient update, thereby preventing catastrophic forgetting when learning new tasks.

The algorithm begins by training the neural network on the first task using standard gradient descent. After training, the input activations for each layer l are extracted and denoted by $\mathbf{X}_1^l$. These activation matrices are used to construct the initial subspace representation through SVD. In practice, the number of samples P in a task can be large, making it inefficient to process the entire activation matrix at once. To improve scalability, the samples are partitioned into smaller disjoint mini-groups of size P_{group} . The incremental SVD procedure is then applied sequentially to each mini-group, allowing the decomposition to be updated efficiently without computing the SVD of the full dataset.

Once the initial subspace representation has been established, the algorithm proceeds to the CL phase. For each new task $\mathcal{T}_t$, the network is trained using the projected gradient update rule introduced in Section 5.4. Specifically, the gradient at each layer is projected onto the null space of the previously learned input subspace before updating the weights. This projection ensures that the weight updates do not modify the network outputs associated with earlier tasks, thereby preserving previously acquired knowledge.

After training on the current task, the activation matrices $\mathbf{X}_t^l$ are incorporated into the existing subspace representation using the incremental SVD update. Similar to the initialization stage, the activation samples are divided into mini-groups of size P_{group} , and the incremental SVD procedure is applied iteratively over $\lceil P/P_{\text{group}} \rceil$ groups. This incremental update allows the subspace representation to adapt to new data while maintaining computational efficiency.

In practice, the number of neurons M in layer l is typically moderate compared to the total number of samples accumulated across tasks. Since the residual matrix $\mathbf{E}_t^l$ has dimensions $M \times P$, the QR decomposition required in the update step remains computationally feasible. Furthermore, the SVD is performed on the matrix $\mathbf{S}_t^l \in \mathbb{R}^{2M \times (M+P)}$, whose row dimension remains bounded by $2M$. Consequently, the computational cost of the SVD step remains manageable even as additional tasks are introduced. Although the dimension of $\mathbf{V}_t^l$ grows as new samples are incorporated, the null-space computation required for CL depends only on $\mathbf{U}^l$ and $\mathbf{\Sigma}^l$, ensuring that the overall algorithm remains scalable.

The complete procedure of the proposed CL algorithm is summarized in Algorithm 2.

Algorithm 2 Continual Learning with Incremental SVD (CL-ISVD)

Input: Dataset $\{\mathbf{X}_t, \mathbf{Y}_t\}$ for tasks $\mathcal{T}_t, t \in [1, T]$, each with P samples; weight matrices $\mathbf{W}^l \in \mathbb{R}^{M \times N}$ for layers $l \in [1, L]$; threshold η ; mini-group size P_{group} ; number of epochs; learning rate α .

- 1: Train the network on the first task $\{\mathbf{X}_1, \mathbf{Y}_1\}$ using standard gradient descent [89]
- 2: Extract layer input activations $\mathbf{X}_1^l$ for all layers $l \in [1, L]$
- 3: **for** $l = 1$ to L **do**
- 4: Initialize $\mathbf{U}^l, \Sigma^l$
- 5: Partition $\mathbf{X}_1^l$ into mini-groups: $\mathbf{X}_1^l = \bigcup_i \mathbf{X}_{i,1}^l$ with $|\mathbf{X}_{i,1}^l| = P_{\text{group}}$
- 6: **for** $i = 1$ to $\lceil P/P_{\text{group}} \rceil$ **do**
- 7: $[\mathbf{U}^l, \Sigma^l, _] \leftarrow \text{ISVD}(\mathbf{U}^l, \Sigma^l, _, \mathbf{X}_{i,1}^l)$
- 8: **end for**
- 9: **end for**
- 10: **for** $t = 2$ to T **do** ▷ Continual learning phase
- 11: **for** epoch = 1 to epochs **do**
- 12: Forward propagate $\mathbf{X}_t$ through the network to obtain layer activations $\mathbf{X}_t^l$
- 13: Compute gradients $\mathbf{G}^l$ via backpropagation
- 14: **for** $l = 1$ to L **do**
- 15: Determine rank r using Eq. (5.20) with threshold η
- 16: $\mathbf{U}_{\text{null}}^l \leftarrow \text{last } (M - r) \text{ columns of } \mathbf{U}^l$
- 17: $\mathbf{P}_{\text{null}}^l = \mathbf{U}_{\text{null}}^l (\mathbf{U}_{\text{null}}^l)^T$
- 18: $\mathbf{W}^l \leftarrow \mathbf{W}^l - \alpha \mathbf{P}_{\text{null}}^l \mathbf{G}^l$
- 19: **end for**
- 20: **end for**
- 21: **for** $l = 1$ to L **do**
- 22: Partition $\mathbf{X}_t^l$ into mini-groups: $\mathbf{X}_t^l = \bigcup_i \mathbf{X}_{i,t}^l$ with $|\mathbf{X}_{i,t}^l| = P_{\text{group}}$
- 23: **for** $i = 1$ to $\lceil P/P_{\text{group}} \rceil$ **do**
- 24: $[\mathbf{U}^l, \Sigma^l, _] \leftarrow \text{ISVD}(\mathbf{U}^l, \Sigma^l, _, \mathbf{X}_{i,t}^l)$
- 25: **end for**
- 26: **end for**
- 27: **end for**

5.7 Computational Complexity Analysis

This section analyzes the computational complexity of the proposed CL-ISVD algorithm for maintaining the null space of the aggregated data matrix in a CL setting. Since network training

through backpropagation is required regardless of the CL strategy, the analysis focuses on the additional computational cost introduced by the incremental SVD updates.

Let M denote the feature dimension (i.e., the number of neurons in a layer) and P the number of samples associated with the current task. In practical DL settings, the number of samples is typically much larger than the feature dimension, and therefore we assume $P \gg M$.

For the ISVD update, the computational cost of each major step can be summarized as follows:

- **Step 1:** Computing the projection residual matrix $\mathbf{E}^l$ in (5.24) requires projecting the new data onto the existing subspace, which incurs a cost of $\mathcal{O}(M^2P)$.
- **Step 2:** Performing the QR decomposition of $\mathbf{E}^l \in \mathbb{R}^{M \times P}$ requires $\mathcal{O}(M^2P)$ operations when $M \ll P$.
- **Step 3–4:** Constructing the intermediate matrix $\mathbf{S}^l$ and computing its SVD involves a matrix of size approximately $2M \times (M + P)$. Since the row dimension is bounded by $2M$, the computational cost is approximately $\mathcal{O}(M(M + P)^2) = \mathcal{O}(MP^2)$.
- **Step 5:** Updating the left singular vectors $\mathbf{U}^l$ requires matrix multiplications of size $M \times M$, leading to a cost of $\mathcal{O}(M^3)$.

Under the practical assumption $P \gg M$, the dominant computational cost arises from the projection and QR decomposition steps. Consequently, the overall computational complexity of the incremental update can be approximated as $\mathcal{O}(MP^2)$.

In practice, the algorithm further improves scalability by partitioning the task data into mini-groups of size P_{group} and performing multiple ISVD updates sequentially. This strategy reduces memory requirements and allows the subspace representation to be updated efficiently without processing the entire dataset at once. Importantly, the row dimension of the intermediate matrix involved in the SVD step remains bounded by $2M$, ensuring that the computational cost does not grow significantly as additional tasks are incorporated.

5.8 Experimental Setup

In this section, we describe the experimental setup, including the benchmark datasets, network architectures, evaluation metrics, and comparison methods. The proposed CL-ISVD algorithm was evaluated on several benchmark CL datasets. After introducing the experimental setup, we present the quantitative results and analyze the performance and efficiency of the proposed method.

5.8.1 Benchmarking Datasets and Network Architecture

We evaluated the performance of the proposed CL-ISVD algorithm on several benchmark datasets, including permuted MNIST, split MNIST [55], split SVHN [90], and split CIFAR-10 [91]. All experiments were conducted using PyTorch [92] for model implementation and training. Both task-incremental and class-incremental experiments were performed.

(a) Permuted MNIST

The original MNIST dataset consists of 10 classes of handwritten digits from 0 to 9, with 60,000 training samples and 10,000 testing samples. In the permuted MNIST experiment, sequential tasks were generated by applying a different random pixel permutation to the MNIST images for each task. Each task contains the same 10 digit classes, but the input distribution changes due to the different pixel permutation.

This task-incremental setup simulates a CL scenario where the model must adapt to new input distributions over time. For this task, we trained a three-layer MLP with the architecture [784–256–10]. The model was trained using the mean squared error (MSE) loss function with ReLU activation in the hidden layer [93]. Stochastic gradient descent (SGD) was used for optimization with a learning rate of 1×10^{-2} . The model was trained for 30 epochs with a batch size of 100.

(b) Split MNIST

In the split MNIST experiment, the MNIST dataset was divided into 10 sequential tasks, where each task contains samples from one digit class. This creates a class-incremental learning setting in which the model learns the digit classes sequentially.

For this experiment, we used a four-layer MLP with the architecture [784–512–512–10]. Similar to the permuted MNIST experiment, the model was trained using MSE loss with ReLU activation. SGD was used for optimization with a learning rate of 1×10^{-2} and a momentum of 0.9. The model was trained for 50 epochs.

(c) Split SVHN

The SVHN dataset is more complex than MNIST and contains digit images captured from real-world house numbers. It includes 73,257 training samples and 26,032 testing samples. In the split SVHN experiment, the dataset was divided into 5 sequential tasks, with each task containing two digit classes, as shown in Table 5.1.

For this dataset, we used a CNN consisting of two convolutional layers followed by three fully connected layers. The two convolutional layers had 32 and 64 filters, respectively. Each convolutional layer used a 5×5 kernel, and a 2×2 max-pooling layer was applied after the second convolutional layer. Dropout with a probability of 0.1 was applied to each max-pooling layer to reduce overfitting.

The three fully connected layers had 512, 256, and 10 neurons, respectively. The model was trained using cross-entropy loss with SGD optimization. The learning rate was set to 1×10^{-2} , and the momentum was set to 0.9. The model was trained for 50 epochs.

(d) Split CIFAR-10

The CIFAR-10 dataset contains 60,000 color images with a spatial size of 32×32 pixels. It includes 50,000 training samples and 10,000 testing samples across 10 object categories. In the split CIFAR-10 experiment, the dataset was divided into 5 sequential tasks, with each task containing two categories, as shown in Table 5.1.

For this experiment, we used a CNN consisting of three convolutional layers and three fully connected layers. The three convolutional layers had 64, 128, and 256 filters, respectively, and each convolutional layer used a 2×2 kernel. A 2×2 max-pooling layer was applied after each

convolutional layer. Dropout with a probability of 0.1 was also applied after each max-pooling layer.

The extracted features were flattened and then passed to the fully connected layers with the architecture [1024–512–10]. ReLU was used as the activation function for all layers. The model was trained using cross-entropy loss with SGD optimization. The learning rate and momentum were set to 1×10^{-2} and 0.9, respectively. The model was trained for 50 epochs.

Table 5.1: Labels for split SVHN and split CIFAR-10 in each task.

Task	Split SVHN	Split CIFAR-10
Task 1	digits 0, 1	Airplane, car
Task 2	digits 2, 3	Bird, cat
Task 3	digits 4, 5	Deer, dog
Task 4	digits 6, 7	Frog, horse
Task 5	digits 8, 9	Ship, truck

5.8.2 Evaluation Metrics

To evaluate the effectiveness of CL and quantify the extent of catastrophic forgetting, we used three widely adopted metrics in the CL literature, namely Average Accuracy (ACC), Backward Transfer (BWT), and Forward Transfer (FWT) [80, 94, 95].

Let $a_{t,\tau}$ be the test classification accuracy of the model on task $\mathcal{T}_\tau$ after observing the last sample from task $\mathcal{T}_t$, and let b_τ be the accuracy of a reference model, which is trained from scratch with random initialization using only the data from task $\mathcal{T}_\tau$. Then ACC, BWT, and FWT are defined as

$$\text{ACC} = \frac{1}{T} \sum_{\tau=1}^T a_{T,\tau}, \quad (5.29)$$

$$\text{BWT} = \frac{1}{T-1} \sum_{\tau=1}^{T-1} (a_{T,\tau} - a_{\tau,\tau}), \quad (5.30)$$

and

$$\text{FWT} = \frac{1}{T-1} \sum_{\tau=2}^T (a_{\tau-1,\tau} - b_{\tau}), \quad (5.31)$$

respectively.

ACC measures the average test accuracy across all tasks after training on the final task $\mathcal{T}_T$. It captures the model’s ability to perform well on both previously seen and current tasks, serving as an overall indicator of CL performance.

BWT quantifies the influence of learning new tasks on the performance of earlier ones. Specifically, it measures the change in accuracy for each previous task $\mathcal{T}_{\tau}$ after the model has been trained on all tasks up to $\mathcal{T}_T$, compared to the accuracy it achieved immediately after learning $\mathcal{T}_{\tau}$. A negative BWT indicates catastrophic forgetting, where the model’s performance on earlier tasks has deteriorated. Conversely, a value closer to zero or positive suggests better retention of learned knowledge.

FWT quantifies how training on earlier tasks influences performance on tasks that have not yet been encountered. A positive FWT indicates that the model can leverage useful knowledge from previous tasks to improve performance on future tasks before directly training on them.

In practice, ACC is used to assess the model’s final performance across tasks, while BWT highlights the system’s ability to maintain and preserve previously acquired knowledge. FWT provides insight into the model’s ability to generalize or transfer beneficial representations forward in time. Together, these metrics provide a comprehensive view of the behavior of CL systems.

5.8.3 Baseline and Benchmark Methods

To assess the effectiveness of the proposed CL-ISVD method for CL on these datasets, we conducted a comprehensive comparison with several established benchmark algorithms. All methods used the same network architecture and training/testing datasets for a fair comparison. The implementation of the following benchmark methods was based on two open-source libraries: Avalanche [96] and LibContinual [97].

EWC: Elastic Weight Consolidation (EWC) [39] is a regularization-based approach that mitigates catastrophic forgetting by constraining the neural network weights using the diagonal elements of the Fisher information matrix. The regularization factor λ was set to 1, following the original implementation.

RWalk: RWalk [85] generalizes EWC with a theoretically grounded KL-divergence-based perspective. It introduces parameter importance scores based on the sensitivity of the loss over the movement on the Riemannian manifold.

SI: Synaptic Intelligence (SI) [77] assigns an importance score to each parameter to slow changes in critical weights. In SI, the damping parameter of the consolidation cost term ξ was set to 1×10^{-3} .

LwF: Learning without Forgetting (LwF) [98] applies knowledge distillation to preserve previously acquired knowledge while simultaneously training on new tasks. It uses the previous model outputs as soft targets to reduce forgetting.

GEM: Gradient Episodic Memory (GEM) [80] optimizes gradients to avoid interference between new and previously learned tasks. GEM stores data from previous tasks, and the memory size was configured to 5120.

PGMA: PGMA [66] adapts the model to diverse data by generating new parameters, functioning similarly to pseudo-replay methods.

GDumb: GDumb [81] greedily stores samples in memory as they arrive. At test time, it trains a model from scratch using only the samples stored in memory.

GSS: Gradient-based Sample Selection (GSS) [99] formulates sample selection as a constraint reduction problem based on the constrained optimization view of CL. The goal is to select a fixed subset of constraints that best approximates the feasible region defined by the original constraints.

OCM: OCM [100] is based on mutual information maximization. It reduces feature bias caused by cross-entropy and encourages the learning of each task to make use of holistic representations or the full features of the task training data. It also encourages preservation of previously learned knowledge when training on new incrementally arriving data.

OGD: Orthogonal Gradient Descent (OGD) [43] maintains a memory of past task gradients and projects new gradients orthogonally to avoid interference. This orthogonalization ensures that updates do not disturb previously learned knowledge.

OWM: Orthogonal Weight Modification (OWM) [82] introduces a projection matrix to ensure that weight updates do not disrupt previously learned tasks. It incrementally computes the projection matrix using an approximate matrix inversion. The stability factor was set to 1×10^{-4} .

GPM: Gradient Projection Memory (GPM) [44] performs gradient updates in directions orthogonal to the feature subspaces that are important for previous tasks. These subspaces are stored and used to reduce interference with old tasks.

The EWC, SI, and RWalk methods regularize the changes of parameters across tasks, where each parameter is associated with an importance score. LwF uses knowledge distillation to preserve previously learned knowledge and can also be considered a regularization-based approach. GEM, PGMA, GDumb, GSS, and OCM focus on storing and reusing past samples or synthetic data to overcome forgetting. OGD, OWM, GPM, and the proposed CL-ISVD explicitly restrict gradient updates to subspaces that minimize interference with previous tasks.

5.9 Quantitative Evaluation Results

In this section, we present a detailed evaluation of the proposed CL-ISVD method, comparing it against the aforementioned benchmark methods. For consistency, all models, including CL-ISVD and the benchmarks, used the same network architecture, with equal numbers of layers and neurons per layer. Each experiment was repeated 10 times to ensure statistical reliability, and the results are presented as the mean accuracy with standard deviation. The hyperparameter settings for CL-ISVD were $\eta = 0.8$ and $P_{\text{group}} = 1000$.

The results are presented in Tables 5.2, 5.3, 5.4, and 5.5 for permuted MNIST, split MNIST, split SVHN, and split CIFAR-10, respectively. The best-performing values are highlighted in bold. Figure 5.2 illustrates the ACC and BWT as a function of the number of tasks learned. The evaluation protocol consists of assessing classification accuracy after the completion of each task.

Table 5.2: Performance evaluation on permuted MNIST with 10 tasks.

Type	Method	ACC	BWT	FWT
Regularization	EWC	89.12 ± 0.72	-1.05 ± 0.10	-0.12 ± 0.03
	LwF	88.45 ± 0.81	-2.15 ± 0.18	-0.13 ± 0.03
	SI	90.02 ± 0.74	-2.08 ± 0.16	-0.14 ± 0.04
	RWalk	90.55 ± 0.63	-0.98 ± 0.14	-0.15 ± 0.04
Replay	GEM	91.22 ± 0.55	-0.84 ± 0.12	-0.06 ± 0.02
	PGMA	91.86 ± 0.58	-0.95 ± 0.20	-0.05 ± 0.02
	GDumb	92.75 ± 0.41	-0.65 ± 0.10	-0.05 ± 0.02
	GSS	91.40 ± 0.49	-0.88 ± 0.15	-0.07 ± 0.03
	OCM	91.95 ± 0.44	-0.78 ± 0.13	-0.06 ± 0.02
Projection	OGD	92.48 ± 0.38	-0.82 ± 0.10	-0.07 ± 0.02
	OWM	93.45 ± 0.31	-0.68 ± 0.08	-0.07 ± 0.02
	GPM	92.85 ± 0.33	-0.72 ± 0.09	-0.06 ± 0.02
	CL-ISVD	93.56 ± 0.31	-0.60 ± 0.07	-0.08 ± 0.03

These results point to several observations about the models considered here. As far as classification accuracy and catastrophic forgetting are concerned, CL-ISVD provides the best overall results on the first three datasets. It achieves the highest ACC and best BWT on permuted MNIST, split MNIST, and split SVHN. The OWM method is the closest competitor, also providing strong classification accuracy and limited forgetting.

On split CIFAR-10, CL-ISVD achieves the second-best ACC and the best BWT. Although OWM obtains the highest classification accuracy on this dataset, CL-ISVD shows the smallest backward performance degradation, indicating stronger retention of previously learned knowledge.

It must be pointed out that the primary difference between CL-ISVD and OWM lies in how the null space is determined. In OWM, the projection matrix is computed using the approximate generalized inverse of the feature matrix, which may accumulate errors during incremental updates. In contrast, in CL-ISVD, the null space is specified by the singular vectors associated with the small singular values of the data matrix. This difference contributes to the strong performance of CL-ISVD, as demonstrated in Tables 5.2, 5.3, 5.4, and 5.5.

Table 5.3: Performance evaluation on split MNIST with 10 tasks.

Type	Method	ACC	BWT	FWT
Regularization	EWC	57.84 ± 0.68	-1.38 ± 0.15	0.07 ± 0.02
	LwF	48.90 ± 0.75	-2.85 ± 0.22	0.09 ± 0.03
	SI	53.42 ± 0.83	-2.42 ± 0.20	0.10 ± 0.03
	RWalk	60.12 ± 0.70	-1.95 ± 0.18	0.08 ± 0.03
Replay	GEM	80.34 ± 0.60	-1.85 ± 0.16	0.09 ± 0.03
	PGMA	83.72 ± 0.78	-2.45 ± 0.25	0.08 ± 0.03
	GDumb	88.14 ± 0.52	-1.28 ± 0.14	0.09 ± 0.03
	GSS	78.95 ± 0.65	-1.98 ± 0.18	0.09 ± 0.03
	OCM	85.10 ± 0.60	-1.55 ± 0.17	0.09 ± 0.03
Projection	OGD	83.10 ± 0.55	-1.55 ± 0.14	0.09 ± 0.03
	OWM	90.60 ± 0.30	-1.18 ± 0.11	0.10 ± 0.03
	GPM	89.80 ± 0.42	-1.30 ± 0.12	0.09 ± 0.03
	CL-ISVD	90.95 ± 0.35	-1.12 ± 0.10	0.10 ± 0.03

Compared with memory-based approaches, the proposed CL-ISVD does not require storing large amounts of raw data from previous tasks. Instead, CL-ISVD only requires storing the left singular matrices and the singular values in the diagonal singular matrices of each layer, leading to more memory-efficient learning.

Finally, it is useful to compare the performance of projection-based approaches with regularization-based algorithms. The results show that projection-based approaches tend to outperform regularization-based approaches in CL. Regularization methods, such as EWC and SI, rely on estimating parameter importance using the Fisher information matrix or predefined importance scores, which may not fully capture complex parameter interactions in deep networks. In contrast, projection-based methods provide more direct control over the update direction, reducing interference with previous tasks while maintaining the ability to learn new tasks.

Table 5.4: Performance evaluation on split SVHN with 5 tasks.

Type	Method	ACC	BWT	FWT
Regularization	EWC	18.20 ± 0.50	-4.30 ± 0.80	0.18 ± 0.08
	LwF	15.80 ± 0.55	-3.90 ± 0.70	0.20 ± 0.08
	SI	20.50 ± 0.90	-4.40 ± 0.85	0.17 ± 0.07
	RWalk	30.10 ± 0.50	-3.85 ± 0.45	0.18 ± 0.08
Replay	GEM	50.20 ± 0.45	-2.50 ± 0.40	0.22 ± 0.10
	PGMA	55.10 ± 0.60	-3.00 ± 0.90	0.19 ± 0.08
	GDumb	62.90 ± 0.50	-2.20 ± 0.30	0.23 ± 0.09
	GSS	48.70 ± 0.55	-2.90 ± 0.38	0.21 ± 0.09
	OCM	58.10 ± 0.50	-2.70 ± 0.35	0.22 ± 0.09
Projection	OGD	60.50 ± 0.35	-2.05 ± 0.15	0.22 ± 0.08
	OWM	69.10 ± 0.40	-2.40 ± 0.20	0.24 ± 0.09
	GPM	67.80 ± 0.38	-2.25 ± 0.25	0.23 ± 0.09
	CL-ISVD	70.80 ± 0.32	-2.00 ± 0.22	0.22 ± 0.07

5.10 Performance and Efficiency Analysis

In this section, we investigate the impact of several parameters on the performance and computational efficiency of the proposed CL-ISVD method.

5.10.1 Empirical Rank Determination

In our implementation of CL-ISVD, the rank factor was set to $\eta = 0.8$. To evaluate the impact of this parameter on the performance, we conducted an experiment on the permuted MNIST dataset by monitoring the classification performance while varying η . The results, shown in Figure 5.3, indicate that the performance of the model remained stable and consistent when η was varied between 0.7 and 0.9.

Additionally, we analyzed the singular values of the permuted MNIST dataset by plotting $\Gamma(r)$ as a function of r , where $\Gamma(r)$ is defined in Eq. (5.19). As shown in Figure 5.4, setting $\eta = 0.8$ effectively selects the most significant singular values, with the corresponding singular vectors adequately capturing the dominant input subspace.

Table 5.5: Performance evaluation on split CIFAR-10 with 5 tasks.

Type	Method	ACC	BWT	FWT
Regularization	EWC	21.10 ± 0.65	-5.20 ± 1.20	0.12 ± 0.04
	LwF	18.60 ± 0.70	-4.80 ± 1.25	0.15 ± 0.05
	SI	17.50 ± 1.40	-6.10 ± 1.60	0.17 ± 0.08
	RWalk	27.80 ± 0.80	-5.04 ± 0.75	0.15 ± 0.07
Replay	GEM	38.10 ± 0.90	-4.20 ± 0.45	0.19 ± 0.08
	PGMA	41.90 ± 0.80	-5.10 ± 1.20	0.22 ± 0.10
	GDumb	49.80 ± 0.60	-3.90 ± 0.55	0.21 ± 0.09
	GSS	35.65 ± 0.70	-4.80 ± 0.68	0.18 ± 0.09
	OCM	42.80 ± 0.85	-4.65 ± 0.58	0.20 ± 0.09
Projection	OGD	44.20 ± 0.60	-4.50 ± 0.50	0.26 ± 0.11
	OWM	55.70 ± 0.35	-3.15 ± 0.27	0.20 ± 0.11
	GPM	52.20 ± 0.60	-3.70 ± 0.50	0.21 ± 0.10
	CL-ISVD	54.30 ± 0.50	-3.05 ± 0.43	0.19 ± 0.08

The remaining singular vectors are then used to approximate the null space for projected parameter updates. In situations where the data matrix $\mathbf{X}_t^l$ contains a large number of samples, it may be difficult to define a perfect null space that completely prevents forgetting. As shown in Figure 5.4, the empirical rank selection provides an approximation of the null space by selecting a subset of singular directions. This approximation may contribute to small amounts of forgetting, as reflected in the performance variations shown in Figure 5.3.

5.10.2 Impact of ISVD Update Group Size

The proposed CL-ISVD method can process multiple samples jointly through an ISVD update group size, denoted as P_{group} . To study the impact of this parameter, we conducted ablation experiments using different P_{group} values. The performance was evaluated in terms of ACC, and the results are summarized in Table 5.6.

It can be observed that CL-ISVD demonstrates stable and consistent performance across different ISVD update group sizes. This consistency suggests that the null space is calculated accurately

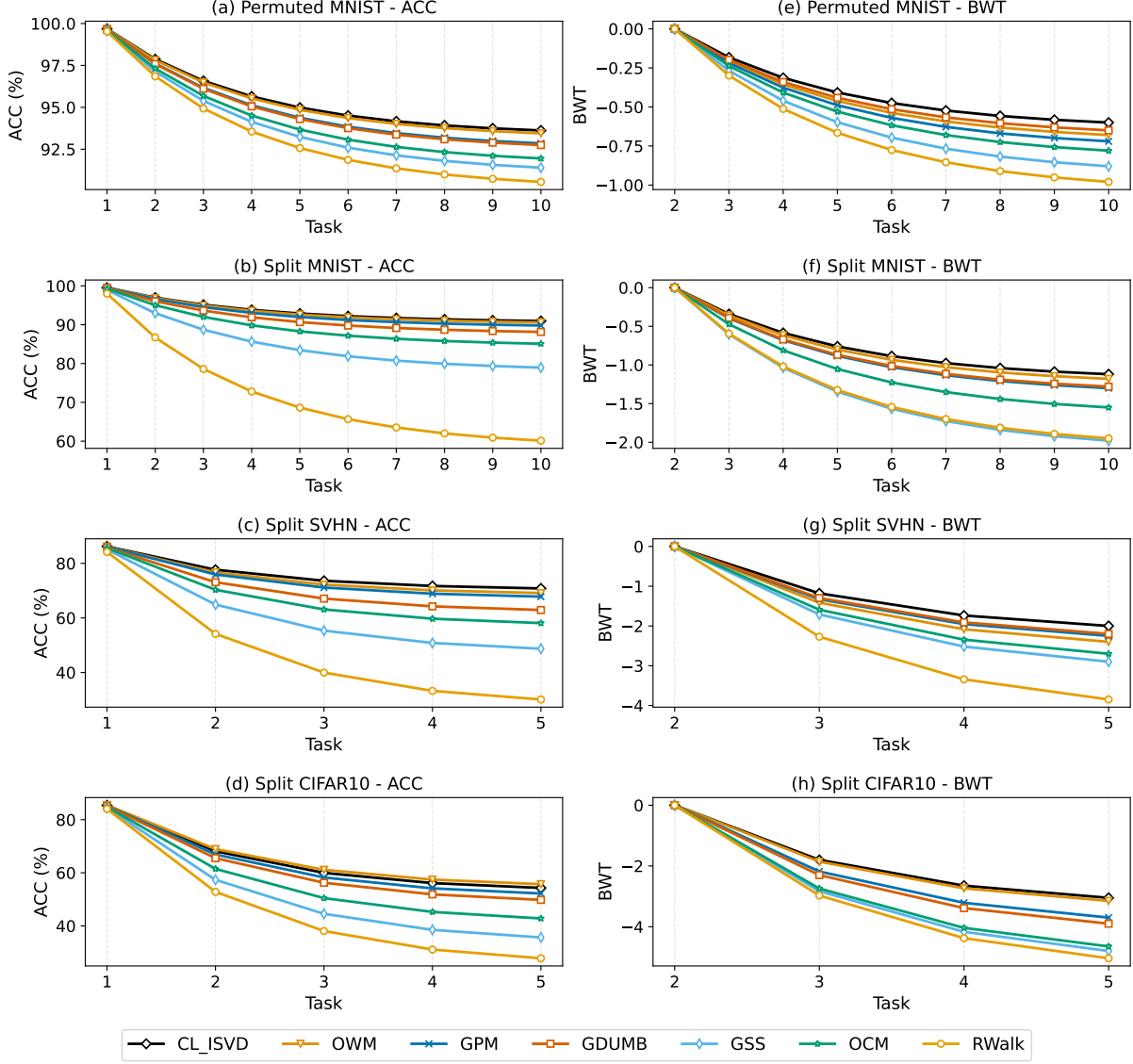


Figure 5.2: Average classification accuracy and BWT over all learned tasks from different benchmark datasets as a function of the number of encountered tasks. The results are shown for permuted MNIST, split MNIST, split SVHN, and split CIFAR-10.

regardless of the choice of P_{group} . This result is expected because splitting the data into multiple groups does not significantly alter the final subspace representation obtained through incremental SVD.

5.10.3 Computational Efficiency

We compared the computational performance of the proposed CL-ISVD method under different update group sizes. The experiment was performed on the MNIST training dataset, which

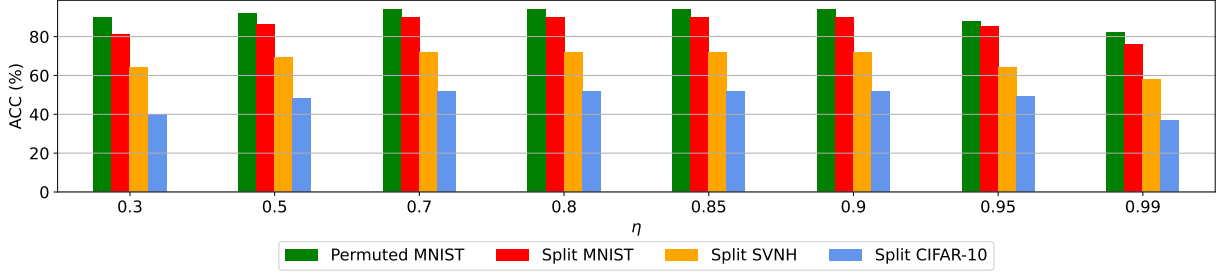


Figure 5.3: Study of different η values on the ACC of CL-ISVD on the permuted MNIST dataset.

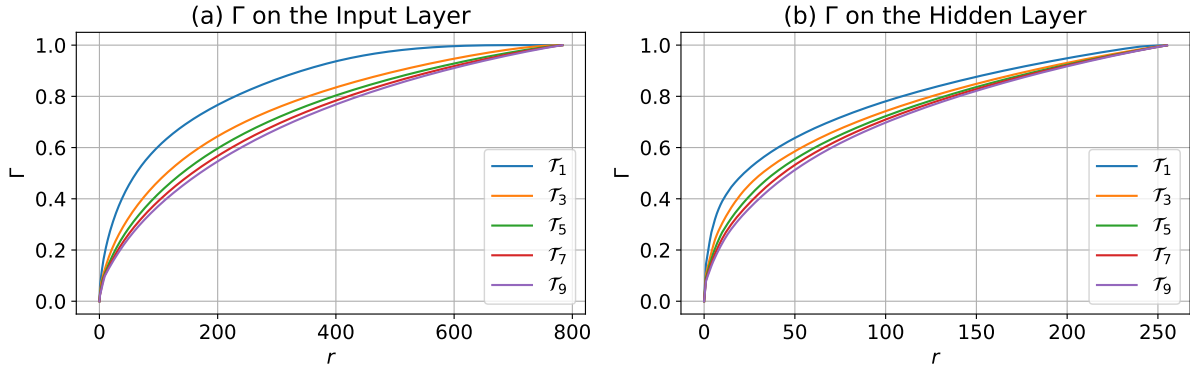


Figure 5.4: The value of $\Gamma(r)$ as a function of r for the input layer and hidden layer on the permuted MNIST dataset. The plots show the results for tasks $\mathcal{T}_1$, $\mathcal{T}_3$, $\mathcal{T}_5$, $\mathcal{T}_7$, and $\mathcal{T}_9$.

contains 60,000 samples. The experiment was executed on a CPU-based Linux platform with 12 GB of RAM. For each selected P_{group} , the experiment was repeated ten times, and the average computational speed in milliseconds per processed sample is summarized in Table 5.7. The case $P_{\text{group}} = 1$ corresponds to a per-sample incremental SVD update and is included for reference.

As shown in Table 5.7, CL-ISVD achieves a substantial improvement in computational efficiency when multiple samples are processed jointly. Across different group sizes, values in the range $P_{\text{group}} \in [625, 2000]$ exhibit similar performance and represent practical choices for efficient implementation. In contrast, the per-sample update case is significantly slower.

This demonstrates the computational advantage of updating the subspace representation using mini-groups rather than individual samples. Overall, the efficiency analysis confirms that CL-ISVD can maintain the required subspace representation with manageable computational cost.

Table 5.6: Performance under different update group sizes for ISVD.

P_{group}	Permuted	Split	Split	Split
	MNIST	MNIST	SVHN	CIFAR-10
	(10 tasks)	(10 tasks)	(5 tasks)	(5 tasks)
2000	93.56 ± 0.30	90.26 ± 0.34	72.16 ± 0.44	51.42 ± 0.53
1250	93.55 ± 0.31	90.28 ± 0.36	72.18 ± 0.41	51.17 ± 0.52
1000	93.56 ± 0.31	90.28 ± 0.36	72.18 ± 0.41	51.17 ± 0.52
625	93.55 ± 0.31	90.28 ± 0.36	72.18 ± 0.41	51.17 ± 0.52
500	93.54 ± 0.29	90.27 ± 0.35	72.18 ± 0.45	51.15 ± 0.51
400	93.51 ± 0.32	90.28 ± 0.36	72.17 ± 0.41	51.16 ± 0.52

Table 5.7: Computational speed comparison in milliseconds per sample under different P_{group} settings.

Method	P_{group}							
	2000	1250	1000	625	500	400	100	1
ISVD	1.4	1.5	1.6	1.7	1.9	1.9	2.8	207.3

Since the practical implementation only requires storing and updating the left singular vectors and singular values, the method remains scalable as new tasks are introduced.

5.11 Conclusion

This chapter presented a null-space-based CL framework using incremental SVD. First, a theoretical condition for learning without catastrophic forgetting was derived. The analysis showed that if parameter updates lie in the null space of the input representations from previous tasks, then the corresponding layer outputs for those tasks remain unchanged. Based on this condition, a projected gradient update rule was formulated to reduce interference with previously learned knowledge while allowing the model to adapt to new tasks.

To make the proposed framework computationally practical, this chapter introduced CL-ISVD, which incrementally updates the SVD of the accumulated input representation matrix. Instead of

recomputing the full SVD whenever new task data arrive, CL-ISVD updates the subspace representation using mini-groups of samples. This makes the method scalable for CL problems with large training datasets.

The proposed CL-ISVD method was evaluated on several benchmark CL datasets, including permuted MNIST, split MNIST, split SVHN, and split CIFAR-10. The results showed that CL-ISVD achieves strong performance compared with representative regularization-based, replay-based, and projection-based methods. In particular, CL-ISVD maintains high average accuracy while reducing backward transfer degradation, indicating improved retention of previously learned knowledge.

Additional experiments investigated the effect of the empirical rank threshold and the ISVD update group size. The results showed that the method is relatively stable across a range of parameter settings, demonstrating the robustness of the proposed null-space approximation. The computational efficiency analysis further showed that processing multiple samples through the ISVD update significantly improves efficiency compared with per-sample updates.

Overall, the results demonstrate that CL-ISVD provides an effective and scalable approach for CL. By combining null-space-constrained parameter updates with incremental SVD, the method reduces catastrophic forgetting without requiring large memory buffers of previous task data. However, the SVD-based subspace update can still introduce computational overhead as the number of tasks and activation samples increases. This motivates the next chapter, which develops more efficient subspace tracking methods based on incremental PCA and recursive least squares.

Chapter 6

Continual Learning using Subspace Tracking

6.1 Introduction

The previous chapter introduced ISVD, a null-space-based CL algorithm that maintains the left singular subspace of the aggregated activation matrix using incremental SVD. In this chapter, we review incremental PCA-based subspace tracking methods [101–103] for null-space CL. The main idea is based on the relationship between SVD and PCA. Specifically, the left singular vectors of the aggregated activation matrix span the same subspace as the eigenvectors of its covariance matrix. Therefore, the problem of estimating the null space of the aggregated activation matrix can be reformulated as a PCA tracking problem.

Based on this observation, this chapter first presents CL via incremental PCA proposed in [101], referred to as CL-IPCA. Instead of updating the full SVD of the aggregated activation matrix, CL-IPCA tracks the principal subspace using sample-wise incremental PCA updates [101]. In addition to CL-IPCA, this chapter also reviews a recursive least squares (RLS)-based subspace tracking method, referred to as CL-RLS, which was originally introduced in [102, 103].

The remainder of this chapter is organized as follows. Section 6.2 reformulates SVD-based null-space estimation as a PCA problem. Section 6.3 presents the incremental PCA formulation for null-space estimation in [101]. Section 6.4 introduces the recursive least squares formulation for subspace tracking [102, 103]. Section 6.5 presents the CL-IPCA and CL-RLS algorithms. Section 6.6 compares the computational complexity of CL-ISVD, CL-IPCA, and CL-RLS. Section 6.7 presents the experimental comparison of the three methods. Finally, Section 6.8 concludes this chapter.

6.2 PCA Formulation of Null-Space Estimation

In the ISVD method presented in the previous chapter, CL is achieved by incrementally updating the left singular subspace of the aggregated activation matrix $\mathbf{A}_t^l$. This subspace is used to identify the null space required for constraining gradient updates and preserving knowledge from previously learned tasks. Although the SVD-based formulation provides a principled solution, maintaining and updating the SVD can still introduce computational overhead when the dataset grows with new tasks.

Recall that the aggregated activation matrix is $\mathbf{A}_t^{l\top}$, and the null space of $\mathbf{A}_t^{l\top}$ corresponds to the subspace orthogonal to the column space of $\mathbf{A}_t^l$. Let the SVD of $\mathbf{A}_t^l$ be

$$\mathbf{A}_t^l = \mathbf{U}_t^l \Sigma_t^l \mathbf{V}_t^{l\top}. \quad (6.1)$$

The column space of $\mathbf{A}_t^l$ is spanned by the left singular vectors associated with the largest singular values, while the null space of $\mathbf{A}_t^{l\top}$ is spanned by the remaining left singular vectors. Since the left singular vectors of $\mathbf{A}_t^l$ are equivalent to the eigenvectors of the covariance matrix

$$\mathbf{C}_t^l = \mathbf{A}_t^l \mathbf{A}_t^{l\top}, \quad (6.2)$$

identifying the null space of $\mathbf{A}_t^{l\top}$ can be reformulated as a PCA problem [104].

In particular, the dominant eigenvectors of $\mathbf{C}_t^l$ span the column space of $\mathbf{A}_t^l$, while the remaining eigenvectors correspond to directions with small variance and form the null space used in the CL framework. Therefore, by estimating the principal subspace of $\mathbf{A}_t^l$, we can obtain the task-preserving subspace required for null-space projection.

This observation motivates the use of incremental PCA (IPCA) proposed in [101–103], which updates the principal subspace directly as new data arrive. Compared with incremental SVD, the PCA formulation allows the subspace to be updated without explicitly maintaining the full SVD of the aggregated activation matrix. As a result, incremental PCA provides a computationally attractive alternative for large-scale CL problems.

Based on this idea, the following section reviews the method in [101] that leverages incremental PCA to maintain the input subspace representation while preserving the knowledge of previously learned tasks. In addition, an RLS-based subspace tracking method [102, 103] is reviewed as another efficient approach for updating the task-preserving subspace.

6.3 Incremental PCA for Null-Space Estimation

Let the centered input data matrix at layer l of task $\mathcal{T}_t$ be

$$\overline{\mathbf{X}}_t^l = [\mathbf{x}_1^l, \mathbf{x}_2^l, \dots, \mathbf{x}_P^l] \in \mathbb{R}^{M \times P}, \quad (6.3)$$

where $\mathbf{x}_k^l \in \mathbb{R}^M$, $k \in [1, P]$, denotes a zero-mean activation sample arriving sequentially at layer l , potentially originating from different tasks. Then, the associated covariance matrix is

$$\mathbf{C}_t^l = \overline{\mathbf{X}}_t^l \overline{\mathbf{X}}_t^{l\top}. \quad (6.4)$$

The objective of principal component analysis (PCA) is to determine the eigen-pairs

$$\mathbf{C}_t^l \mathbf{u}_m^l = \lambda_m^l \mathbf{u}_m^l, \quad (6.5)$$

where λ_m^l and $\mathbf{u}_m^l$ denote the m -th eigenvalue and eigenvector, respectively.

In CL settings, explicitly forming $\mathbf{C}_t^l$ and computing its eigen-decomposition is computationally prohibitive because the number of samples P can be large.

To address this challenge, we adopt *Candid Covariance-Free Incremental PCA* (CCIPCA) [101], which estimates the principal subspace directly from streaming samples without explicitly constructing the covariance matrix.

Let $\mathbf{C}_t^l(k)$ be the sample covariance matrix after observing sample k , which can be updated as

$$\mathbf{C}_t^l(k) = \frac{k-1}{k} \mathbf{C}_t^l(k-1) + \frac{1}{k} \mathbf{x}_k^l \mathbf{x}_k^{l\top}. \quad (6.6)$$

Let $\mathbf{u}_1^l(k)$ denote the first principal eigenvector with associated largest eigenvalue $\lambda_1^l(k)$. Multiplying $\mathbf{u}_1^l(k-1)$ on both sides, we have

$$\mathbf{C}_t^l(k)\mathbf{u}_1^l(k-1) = \frac{k-1}{k}\mathbf{C}_t^l(k-1)\mathbf{u}_1^l(k-1) + \frac{1}{k}\mathbf{x}_k^l\mathbf{x}_k^{l\top}\mathbf{u}_1^l(k-1). \quad (6.7)$$

CCIPCA [101] defines the *target vector* of the first principal eigenvector as

$$\tilde{\mathbf{u}}_1^l(k) := \mathbf{C}_t^l(k)\mathbf{u}_1^l(k-1), \quad (6.8)$$

and approximates it as

$$\tilde{\mathbf{u}}_1^l(k) \approx \mathbf{C}_t^l(k)\mathbf{u}_1^l(k). \quad (6.9)$$

The target vector $\tilde{\mathbf{u}}_1^l(k)$ has the same direction as the updated principal eigenvector after normalization, while its magnitude provides an estimate of the corresponding eigenvalue. This approximation is reasonable when the first principal eigenvector is close to convergence, such that the change between two consecutive estimates is small, i.e., $\mathbf{u}_1^l(k) \approx \mathbf{u}_1^l(k-1)$. Then, (6.7) can be rewritten as

$$\tilde{\mathbf{u}}_1^l(k) = \frac{k-1}{k}\tilde{\mathbf{u}}_1^l(k-1) + \frac{1}{k}\mathbf{x}_k^l \left[\mathbf{x}_k^{l\top} \mathbf{u}_1^l(k-1) \right], \quad (6.10)$$

which is the updating equation for the target first principal eigenvector.

It should be noted that the direction of $\tilde{\mathbf{u}}_1^l(k)$ is the same as $\mathbf{u}_1^l(k)$, while the magnitude of $\tilde{\mathbf{u}}_1^l(k)$ approximates the associated eigenvalue $\lambda_1^l(k)$. Thus, we obtain $\mathbf{u}_1^l(k)$ by normalizing $\tilde{\mathbf{u}}_1^l(k)$ as

$$\mathbf{u}_1^l(k) = \frac{\tilde{\mathbf{u}}_1^l(k)}{\|\tilde{\mathbf{u}}_1^l(k)\|}, \quad (6.11)$$

and the corresponding eigenvalue is estimated as

$$\lambda_1^l(k) = \|\tilde{\mathbf{u}}_1^l(k)\|. \quad (6.12)$$

For the second principal eigenvector, we define the residual by removing the contribution of the first principal eigenvector from the sample:

$$\mathbf{r}_1^l(k) = \mathbf{x}_k^l - \mathbf{u}_1^l(k) \mathbf{u}_1^l(k)^\top \mathbf{x}_k^l. \quad (6.13)$$

Then, we perform the same procedure to find the target vector of the second principal eigenvector:

$$\tilde{\mathbf{u}}_2^l(k) = \frac{k-1}{k} \tilde{\mathbf{u}}_2^l(k-1) + \frac{1}{k} \mathbf{r}_1^l(k) \left[\mathbf{r}_1^l(k)^\top \mathbf{u}_2^l(k-1) \right]. \quad (6.14)$$

The vector $\tilde{\mathbf{u}}_2^l(k)$ can then be normalized to obtain the eigenvector $\mathbf{u}_2^l(k)$ and the eigenvalue $\lambda_2^l(k)$. This procedure can be repeated until all M principal eigenvectors are estimated.

To summarize, the recursive equations for calculating the target vector and residual for the m -th eigenvector are given by

$$\tilde{\mathbf{u}}_m^l(k) = \frac{k-1}{k} \tilde{\mathbf{u}}_m^l(k-1) + \frac{1}{k} \mathbf{r}_{m-1}^l(k) \left[\mathbf{r}_{m-1}^l(k)^\top \mathbf{u}_m^l(k-1) \right], \quad (6.15)$$

and

$$\mathbf{r}_m^l(k) = \mathbf{r}_{m-1}^l(k) - \mathbf{u}_m^l(k) \mathbf{u}_m^l(k)^\top \mathbf{r}_{m-1}^l(k), \quad (6.16)$$

respectively, where $\mathbf{r}_0^l(k) = \mathbf{x}_k^l$.

After each update, the eigenvector is estimated by normalizing the target vector:

$$\mathbf{u}_m^l(k) = \frac{\tilde{\mathbf{u}}_m^l(k)}{\|\tilde{\mathbf{u}}_m^l(k)\|}, \quad (6.17)$$

with the associated eigenvalue

$$\lambda_m^l(k) = \|\tilde{\mathbf{u}}_m^l(k)\|. \quad (6.18)$$

By incrementally estimating the principal eigenvectors of $\mathbf{X}_t^l$, CCIPCA [101] provides direct access to the least-variance directions that span the null space of $\mathbf{A}_t^{l\top}$. This null space is pre-

cisely the subspace used in the proposed CL framework to constrain parameter updates and reduce catastrophic forgetting.

Algorithm 3 CCIPCA Subspace Update

Require: Current eigenvector matrix $U^l = [\mathbf{u}_1^l, \dots, \mathbf{u}_m^l, \dots, \mathbf{u}_M^l]$, centered input data matrix

$\bar{\mathbf{X}}_t^l = [\mathbf{x}_1^l, \mathbf{x}_2^l, \dots, \mathbf{x}_P^l]$ of layer l

1: **for** $k = 1, \dots, P$ **do**

2: Initialize residual $\mathbf{r}^l \leftarrow \mathbf{x}_k^l$

3: **for** $m = 1, \dots, M$ **do**

4: Project

$$y_m^l \leftarrow \mathbf{r}^{l\top} \mathbf{u}_m^l$$

5: Update target vector

$$\tilde{\mathbf{u}}_m^l \leftarrow \frac{k-1}{k} \tilde{\mathbf{u}}_m^l + \frac{1}{k} y_m^l \mathbf{r}^l$$

6: Estimate eigenvalue

$$\lambda_m^l \leftarrow \|\tilde{\mathbf{u}}_m^l\|$$

7: Normalize eigenvector

$$\mathbf{u}_m^l \leftarrow \frac{\tilde{\mathbf{u}}_m^l}{\lambda_m^l}$$

8: Residual deflation

$$\mathbf{r}^l \leftarrow \mathbf{r}^l - y_m^l \mathbf{u}_m^l$$

9: **end for**

10: **end for**

11: **return** updated U^l and $\{\lambda_m^l\}_{m=1}^M$

It is important to note that CCIPCA [101] assumes zero-mean samples. In practice, we therefore maintain a running estimate of the activation mean $\boldsymbol{\mu}^l$ and subtract it from each incoming activation sample before applying the CCIPCA update.

Algorithm 3 summarizes the CCIPCA subspace update procedure used in this work. Compared with full SVD-based updating, the CCIPCA procedure updates the eigenvector subspace sequentially and avoids explicitly storing the entire activation matrix.

6.4 RLS for Null-Space Estimation

Recall that the input data matrix at layer l of task $\mathcal{T}_t$ is $\mathbf{X}_t^l$, and the centered input data matrix is

$$\overline{\mathbf{X}}_t^l = [\mathbf{x}_1^l, \mathbf{x}_2^l, \dots, \mathbf{x}_P^l] \in \mathbb{R}^{M \times P}, \quad (6.19)$$

where $\mathbf{x}_k^l \in \mathbb{R}^M$, $k \in [1, P]$, denotes a zero-mean activation sample arriving sequentially at layer l , potentially originating from different tasks.

Our objective is to estimate the full principal subspace of the aggregated activation matrix using RLS proposed in [102, 103], where

$$\mathbf{A}_t^l = [\mathbf{X}_1^l, \mathbf{X}_2^l, \dots, \mathbf{X}_{t-1}^l]. \quad (6.20)$$

Let $\mathbf{u}_1^l(k) \in \mathbb{R}^M$ denote the estimated first subspace basis vector after observing k samples. The basis vector can be obtained by minimizing the projection error

$$\mathcal{J}(\mathbf{u}_1^l(k)) = \sum_{i=1}^k \left\| \mathbf{x}_i^l - \mathbf{u}_1^l(k) \mathbf{u}_1^l(k)^\top \mathbf{x}_i^l \right\|_2^2. \quad (6.21)$$

Here, $\mathbf{u}_1^l(k)$ is constrained to be a unit vector, i.e., $\|\mathbf{u}_1^l(k)\|_2 = 1$. After the first basis vector is estimated, the contribution of this component is removed from the input sample through deflation. The same procedure is then applied to the residual to estimate the second basis vector. By repeating this process, the complete principal subspace can be recursively estimated. The detailed derivation of the RLS-based principal component update can be found in [102, 103].

To summarize, the recursive equations for calculating the m -th eigenvector are given by

$$\mathbf{u}_m^l(k) = \mathbf{u}_m^l(k-1) + \frac{y_m^l(k)}{d_m^l(k)} \mathbf{e}_m^l(k), \quad (6.22)$$

where

$$\mathbf{e}_m^l(k) := \mathbf{r}_{m-1}^l(k) - y_m^l(k) \mathbf{u}_m^l(k-1), \quad (6.23)$$

$$d_m^l(k) = d_m^l(k-1) + y_m^l(k)^2, \quad (6.24)$$

$$y_m^l(k) := \mathbf{u}_m^l(k-1)^\top \mathbf{r}_{m-1}^l(k), \quad (6.25)$$

and

$$\mathbf{r}_m^l(k) := \mathbf{r}_{m-1}^l(k) - y_m^l(k) \mathbf{u}_m^l(k), \quad (6.26)$$

with initialization $\mathbf{r}_0^l(k) = \mathbf{x}_k^l$.

The corresponding eigenvalue is estimated as

$$\lambda_m^l(k) = \frac{d_m^l(k)}{k}. \quad (6.27)$$

Algorithm 4 summarizes the RLS-PCA procedure for incremental subspace tracking. The algorithm implicitly estimates the eigenvectors of the autocorrelation matrix $\mathbf{A}_t^l \mathbf{A}_t^{l\top}$ without requiring explicit eigen-decomposition or sorting [102, 103]. Since the full subspace basis is tracked, the null space of $\mathbf{A}_t^{l\top}$ can be identified from the components associated with the smallest eigenvalues.

These components are particularly useful in CL because they represent directions orthogonal to previously learned tasks, allowing new knowledge to be incorporated with reduced interference with prior information.

Unlike CL-ISVD, which performs updates at task boundaries, both IPCA and RLS update the subspace sequentially as new samples arrive. This property allows the RLS method to naturally accommodate streaming data across tasks. However, recursive updates may not strictly preserve orthogonality of the estimated basis. Therefore, periodic re-orthogonalization, such as Gram–Schmidt orthogonalization, may be applied to maintain numerical stability during long CL sequences.

Algorithm 4 RLS Subspace Update

Require: Current subspace matrix $U^l = [\mathbf{u}_1^l, \dots, \mathbf{u}_m^l, \dots, \mathbf{u}_M^l]$, accumulated energy terms $\{d_m^l\}_{m=1}^M$, centered input data matrix $\bar{\mathbf{X}}_t^l = [\mathbf{x}_1^l, \mathbf{x}_2^l, \dots, \mathbf{x}_P^l]$ of layer l

- 1: **for** $k = 1, \dots, P$ **do**
 - 2: Initialize residual $\mathbf{r}_0^l \leftarrow \mathbf{x}_k^l$
 - 3: **for** $m = 1, \dots, M$ **do**
 - 4: Project
$$y_m^l \leftarrow \mathbf{u}_m^{l\top} \mathbf{r}_{m-1}^l$$
 - 5: Update energy term
$$d_m^l \leftarrow d_m^l + (y_m^l)^2$$
 - 6: Compute prediction error
$$\mathbf{e}_m^l \leftarrow \mathbf{r}_{m-1}^l - y_m^l \mathbf{u}_m^l$$
 - 7: Update subspace vector
$$\mathbf{u}_m^l \leftarrow \mathbf{u}_m^l + \frac{y_m^l}{d_m^l} \mathbf{e}_m^l$$
 - 8: Normalize
$$\mathbf{u}_m^l \leftarrow \frac{\mathbf{u}_m^l}{\|\mathbf{u}_m^l\|}$$
 - 9: Estimate eigenvalue
$$\lambda_m^l \leftarrow \frac{d_m^l}{k}$$
 - 10: Residual deflation
$$\mathbf{r}_m^l \leftarrow \mathbf{r}_{m-1}^l - y_m^l \mathbf{u}_m^l$$
 - 11: **end for**
 - 12: **end for**
 - 13: **return** updated U^l and $\{\lambda_m^l\}_{m=1}^M$
-

6.5 Continual Learning using Subspace Tracking

Building upon the incremental PCA and RLS subspace tracking procedures described in the previous sections, we now present the CL frameworks based on CCIPCA and RLS, referred to as CL-IPCA and CL-RLS, respectively.

Similar to CL-ISVD, the objective is to maintain a compact representation of the activation subspace associated with previously learned tasks in order to compute the null-space projection used

in the constrained gradient update. However, instead of maintaining the SVD of the aggregated activation matrix, CL-IPCA and CL-RLS tracks the principal subspace using sample-wise.

Let $\mathbf{A}_t^l$ denote the aggregated activation matrix at layer l up to task t . The principal eigenvectors of $\mathbf{A}_t^l$ span the same subspace as the left singular vectors obtained from the SVD of $\mathbf{A}_t^l$. Therefore, by maintaining the principal component basis $\mathbf{U}^l = [\mathbf{u}_1^l, \dots, \mathbf{u}_M^l]$, we can recover the same subspace information required for computing the null space used in the projected gradient update.

The training procedure follows the same overall framework as CL-ISVD. The network is first trained on the initial task using standard gradient descent. The resulting activation samples are then used to initialize the subspace estimate. During CL, the gradients associated with new tasks are projected onto the null space of the previously learned activation subspace before updating the network weights.

After each new task is learned, the activation samples from the current task are used to update the subspace basis. The complete training procedure for CL-IPCA and CL-RLS is summarized in Algorithm 5.

6.6 Computational Complexity Analysis

This section compares the computational complexity of the three CL methods, namely CL-ISVD, CL-IPCA, and CL-RLS. Although all three methods share the same null-space constrained parameter update during network training, they differ in how the activation subspace is maintained after each task.

CL-ISVD updates the subspace using incremental SVD. The method computes the residual matrix, performs QR decomposition, and updates the singular vectors through the SVD of a small auxiliary matrix. Since the update is performed after accumulating a group of activation samples, CL-ISVD is computationally efficient while preserving the exact singular subspace.

CL-IPCA updates the subspace sequentially using the CCIPCA algorithm. Each incoming activation sample is processed once, and the principal eigenvectors are updated recursively without

Algorithm 5 Continual Learning via Incremental Subspace Tracking

Input: Task datasets $\{\mathbf{X}_t, \mathbf{Y}_t\}_{t=1}^T$; weight matrices $\mathbf{W}^l$ for layers $l \in [1, L]$; threshold η ; epochs; learning rate α ; subspace update method $\text{Method} \in \{\text{CCIPCA}, \text{RLS}\}$.

- 1: Train network on the first task $\{\mathbf{X}_1, \mathbf{Y}_1\}$ using gradient descent
- 2: Extract the layer input activations for all layers, denoted as $\mathbf{X}_1^l$
- 3: **for** $l = 1$ to L **do**
- 4: Center data matrix $\mathbf{X}_1^l$
- 5: Initialize $\mathbf{U}^l \leftarrow \mathbf{I}$
- 6: **if** $\text{Method} = \text{RLS}$ **then**
- 7: Initialize accumulated energy terms $\{d_m^l\}_{m=1}^M$
- 8: $\mathbf{U}^l \leftarrow \text{RLS_Update}(\mathbf{U}^l, \mathbf{X}_1^l)$
- 9: **else**
- 10: $\mathbf{U}^l \leftarrow \text{CCIPCA_Update}(\mathbf{U}^l, \mathbf{X}_1^l)$
- 11: **end if**
- 12: **end for**
- 13: **for** $t = 2$ to T **do**
- 14: **for** epoch = 1 to epochs **do**
- 15: Forward propagate $\mathbf{X}_t$ through the network to obtain layer activations $\mathbf{X}_t^l$
- 16: Compute gradients $\mathbf{G}^l$ via backpropagation
- 17: **for** $l = 1$ to L **do**
- 18: Determine rank r using Eq. (5.19) with threshold η
- 19: Construct $\mathbf{U}_{\text{null}}^l$ from $\mathbf{U}^l$ using the last $(M - r)$ columns
- 20: $\mathbf{P}_{\text{null}}^l \leftarrow \mathbf{U}_{\text{null}}^l \mathbf{U}_{\text{null}}^{l\top}$
- 21: Update weights:
$$\mathbf{W}^l \leftarrow \mathbf{W}^l - \alpha \mathbf{P}_{\text{null}}^l \mathbf{G}^l$$
- 22: **end for**
- 23: **end for**
- 24: **for** $l = 1$ to L **do**
- 25: Forward propagate $\mathbf{X}_t$ to obtain layer input activations $\mathbf{X}_t^l$
- 26: Center data matrix $\mathbf{X}_t^l$
- 27: **if** $\text{Method} = \text{RLS}$ **then**
- 28: $\mathbf{U}^l \leftarrow \text{RLS_Update}(\mathbf{U}^l, \mathbf{X}_t^l)$
- 29: **else**
- 30: $\mathbf{U}^l \leftarrow \text{CCIPCA_Update}(\mathbf{U}^l, \mathbf{X}_t^l)$
- 31: **end if**
- 32: **end for**
- 33: **end for**

explicitly forming the covariance matrix or performing matrix decomposition. Consequently, CL-IPCA has very low computational cost and memory requirements, making it attractive for large-scale CL problems. However, the accuracy of CCIPCA can be affected by the approximation

used in the target vector update. Since the updated eigenvector is estimated recursively using the previous eigenvector, small approximation errors may accumulate during sequential updates, leading to a less accurate subspace estimate compared with SVD-based methods.

CL-RLS also updates the subspace sequentially, but employs RLS to estimate the subspace. Compared with CL-IPCA, CL-RLS incorporates second-order information through RLS updates, which generally improves convergence and numerical stability. However, this comes at the expense of additional computations for each incoming sample. In addition, the subspace estimated by CL-RLS is through an optimization problem. As a result, CL-RLS may produce a less accurate null-space projection than CL-ISVD, which can lead to lower CL performance.

Table 6.1 summarizes the computational complexity of the three methods. Here, M denotes the activation dimension, P denotes the number of activation samples in one task, and r denotes the retained subspace rank. The sample-wise complexity of CL-ISVD is obtained by dividing the group-wise update cost by the mini-group size P_{group} .

Table 6.1: Comparison of computational complexity.

Method	Update Type	Time Complexity	Memory Complexity
CL-ISVD	Batch (group-wise)	$\mathcal{O}(MP_{\text{group}} + M^3)$ per group	$\mathcal{O}(M^2)$
CL-ISVD	Amortized sample-wise	$\mathcal{O}\left(M + \frac{M^3}{P_{\text{group}}}\right)$ per sample	$\mathcal{O}(M^2)$
CL-IPCA	Sample-wise	$\mathcal{O}(Mr)$ per sample	$\mathcal{O}(Mr)$
CL-RLS	Sample-wise	$\mathcal{O}(Mr)$ per sample	$\mathcal{O}(Mr)$

Although CL-IPCA and CL-RLS exhibit similar asymptotic computational complexity, their update mechanisms are fundamentally different. CL-IPCA uses stochastic recursive updates to estimate the eigenvector subspace, whereas CL-RLS performs RLS estimation using second-order information. As a result, CL-RLS generally provides more stable subspace estimates but requires more computations per update.

Compared with CL-IPCA and CL-RLS, CL-ISVD performs updates less frequently using groups of activation samples rather than individual samples. This batch update strategy reduces the

total number of subspace updates while preserving the exact singular subspace through incremental SVD. Since SVD directly decomposes the activation matrix and produces an orthogonal basis ordered by singular values, it provides a more accurate estimate of the dominant and null subspaces than recursive sample-wise approximations. Consequently, CL-ISVD provides a favorable balance between computational efficiency and subspace estimation accuracy.

6.7 Experimental Results and Comparison

In this section, we compare the performance of the three null-space-based CL methods: CL-ISVD, CL-IPCA, and CL-RLS. The experiments use the same benchmark datasets, network architectures, training settings, and evaluation metrics described in the previous chapter. Therefore, the dataset descriptions, model architectures, and definitions of ACC, BWT, and FWT are not repeated here. All three methods use the same null-space projected gradient update rule during sequential learning. The quantitative results are summarized in Table 6.2. The results are reported as mean $\pm$ standard deviation over repeated experiments.

Figure 6.1 further compares the learning behavior of CL-ISVD, CL-IPCA, and CL-RLS on the same datasets in Chapter 5 with representative benchmark methods as additional tasks are introduced. The ACC curves show the overall classification performance after each task, while the BWT curves show the degree of forgetting accumulated during CL.

The results show that CL-ISVD generally maintains the highest ACC and the least negative BWT across the benchmark datasets, indicating stronger knowledge retention. CL-IPCA and CL-RLS also reduce forgetting compared with many existing methods, but their performance is generally lower than CL-ISVD because their sample-wise subspace updates provide approximate estimates of the task-preserving subspace.

Among the two sample-wise methods, CL-RLS usually provides more stable performance than CL-IPCA, especially on the more challenging datasets. This suggests that RLS-based subspace tracking can improve stability compared with the CCIPCA update, while still avoiding the block-wise matrix decomposition required by CL-ISVD.

Table 6.2: Compact comparison of benchmark methods and the proposed CL-ISVD, CL-IPCA, and CL-RLS methods. ACC and BWT are reported for each dataset.

Type	Method	Permuted MNIST		Split MNIST		Split SVHN		Split CIFAR-10	
		ACC	BWT	ACC	BWT	ACC	BWT	ACC	BWT
Regularization	EWC	89.12	-1.05	57.84	-1.38	18.20	-4.30	21.10	-5.20
	LwF	88.45	-2.15	48.90	-2.85	15.80	-3.90	18.60	-4.80
	SI	90.02	-2.08	53.42	-2.42	20.50	-4.40	17.50	-6.10
	RWalk	90.55	-0.98	60.12	-1.95	30.10	-3.85	27.80	-5.04
Replay	GEM	91.22	-0.84	80.34	-1.85	50.20	-2.50	38.10	-4.20
	PGMA	91.86	-0.95	83.72	-2.45	55.10	-3.00	41.90	-5.10
	GDumb	92.75	-0.65	88.14	-1.28	62.90	-2.20	49.80	-3.90
	GSS	91.40	-0.88	78.95	-1.98	48.70	-2.90	35.65	-4.80
	OCM	91.95	-0.78	85.10	-1.55	58.10	-2.70	42.80	-4.65
Projection	OGD	92.48	-0.82	83.10	-1.55	60.50	-2.05	44.20	-4.50
	OWM	93.45	-0.68	90.60	-1.18	69.10	-2.40	55.70	-3.15
	GPM	92.85	-0.72	89.80	-1.30	67.80	-2.25	52.20	-3.70
	CL-IPCA	90.86	-0.87	86.52	-1.35	61.50	-2.30	44.80	-4.14
	CL-RLS	92.40	-0.83	88.16	-1.26	60.80	-2.40	49.50	-3.85
	CL-ISVD	93.56	-0.60	90.95	-1.12	70.80	-2.00	54.30	-3.05

For compactness, Table 6.2 reports ACC and BWT, which are the two most relevant metrics for evaluating final performance and catastrophic forgetting. The full ACC, BWT, and FWT results for the benchmark methods were reported in the previous chapter. Here, the table is extended to include the proposed CL-IPCA and CL-RLS methods for direct comparison with CL-ISVD and other CL baselines. As shown in Table 6.2, CL-ISVD achieves the best overall performance among the three proposed methods. It obtains the highest ACC and the best BWT on all four benchmark datasets. This indicates that the SVD-based subspace update provides the most accurate approximation of the task-preserving subspace, leading to better retention of previously learned knowledge. Since SVD directly factorizes the accumulated activation matrix, the resulting singular vectors form an orthogonal basis ordered by singular values. This allows CL-ISVD to more accurately separate the dominant task-related subspace from the null space, thereby reducing interference when learning new tasks.

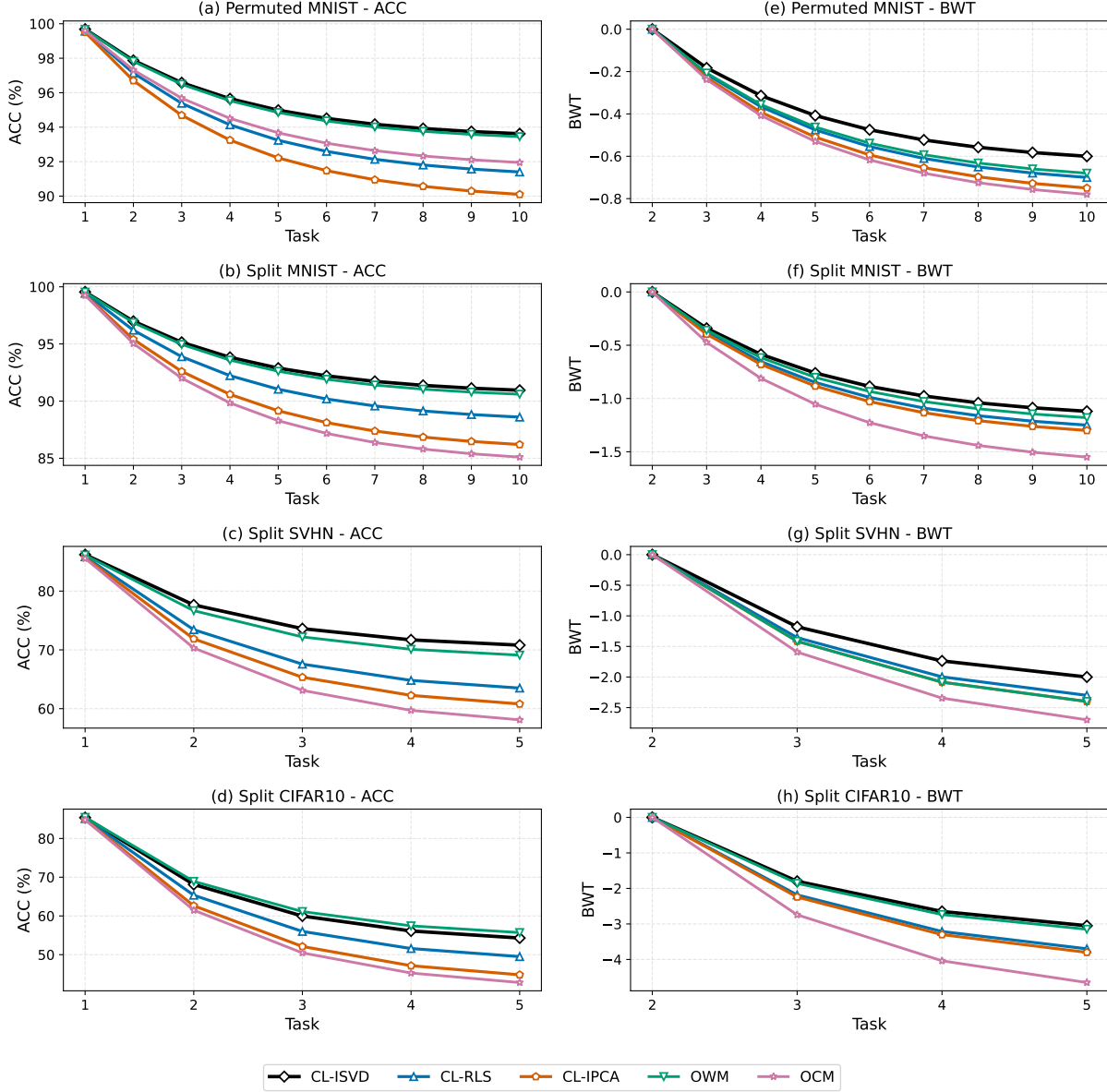


Figure 6.1: Average classification accuracy and BWT over all learned tasks as a function of the number of encountered tasks.

CL-IPCA and CL-RLS also reduce catastrophic forgetting, but their performance is generally lower than CL-ISVD. This is expected because both methods rely on sample-wise recursive updates to estimate the principal subspace. These updates are more computationally efficient, but the resulting subspace estimates may be less accurate than those obtained from the block-wise SVD update.

In addition to the overall performance comparison, we evaluate the sensitivity of the three proposed methods to the rank threshold η . A larger η preserves more task-related directions and provides stronger protection against forgetting, while a smaller η leaves a larger null space for learning new tasks. As shown in Figure 6.2, the performance of all three methods remains relatively stable across a range of η values. CL-ISVD generally achieves the highest ACC, suggesting that the SVD-based update provides a more accurate estimate of the task-preserving subspace. CL-IPCA and CL-RLS also show stable behavior, indicating that the sample-wise subspace tracking methods are not highly sensitive to the exact choice of η .

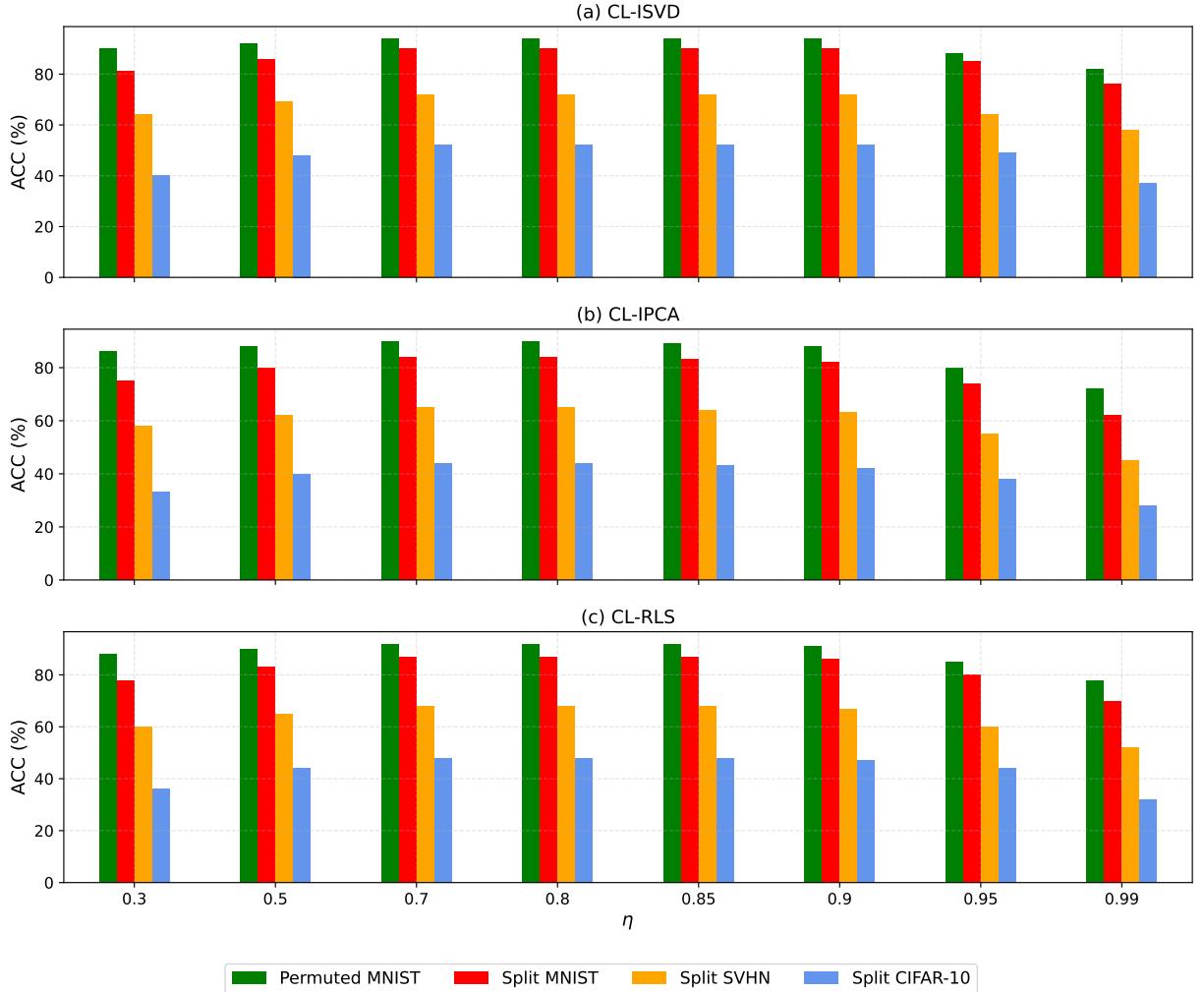


Figure 6.2: Study of different η values on the ACC of CL-ISVD, CL-IPCA, and CL-RLS on the permuted MNIST dataset.

Compared with other benchmark methods, the proposed CL-ISVD achieves strong and consistent performance across the four datasets. Among all methods, CL-ISVD obtains the highest ACC on permuted MNIST, split MNIST, and split SVHN, and achieves the best BWT on all four datasets, indicating the strongest retention of previously learned knowledge. On split CIFAR-10, OWM obtains the highest ACC, while CL-ISVD still achieves the best BWT, suggesting that CL-ISVD is particularly effective at reducing catastrophic forgetting. Compared with regularization-based methods, the proposed projection-based methods show clear advantages, especially on the more challenging split SVHN and split CIFAR-10 datasets. Replay-based methods such as GDumb also achieve competitive results, but they require storing previous samples, whereas the proposed methods preserve prior knowledge through subspace projection without relying on a memory buffer.

Overall, these results demonstrate a clear trade-off among the three methods. CL-ISVD provides the best CL performance and the strongest resistance to forgetting, while CL-IPCA and CL-RLS provide more lightweight sample-wise alternatives. Therefore, CL-ISVD is preferred when accuracy and knowledge retention are the primary goals, whereas CL-IPCA and CL-RLS may be useful when computational efficiency and online updating are more important.

6.8 Conclusion

This chapter presented incremental PCA-based subspace tracking methods for null-space CL. Building on the CL-ISVD framework introduced in the previous chapter, the null-space estimation problem was reformulated as a principal subspace tracking problem by using the relationship between SVD and PCA.

Based on this reformulation, this chapter introduced CL-IPCA, which uses CCIPCA to update the principal subspace sequentially as new activation samples arrive. This avoids repeatedly computing the full SVD of the aggregated activation matrix and provides a lightweight approach for maintaining the task-preserving subspace.

This chapter also introduced CL-RLS, which uses RLS to update the activation subspace. Similar to CL-IPCA, CL-RLS performs sample-wise subspace updates and does not require storing historical activation data. Compared with CL-IPCA, the RLS-based formulation incorporates recursive least-squares information, which can improve the stability of subspace tracking.

The computational analysis and experimental comparison showed that CL-ISVD, CL-IPCA, and CL-RLS offer different trade-offs. CL-ISVD provides the strongest CL performance and best retention of previously learned knowledge, while CL-IPCA and CL-RLS provide more computationally efficient alternatives for online or large-scale learning scenarios.

Overall, the three proposed methods provide a unified null-space-based CL framework with different subspace tracking strategies. CL-ISVD is preferred when accuracy and knowledge retention are the primary goals, while CL-IPCA and CL-RLS are useful when computational efficiency, memory usage, or sample-wise updating is more important.

Chapter 7

Continual Learning for Multi-Domain Precipitation Estimation

7.1 Introduction

Building upon the theoretical framework developed in the previous chapter, this chapter presents its application to multi-domain precipitation estimation. In particular, the null-space-based parameter update strategy [42, 44] is incorporated into a DL model to enable sequential learning across multiple geographical domains while mitigating catastrophic forgetting [37, 38]. The focus of this chapter is on formulating the multi-domain precipitation estimation problem, defining the study domains, describing the overall CL framework, and evaluating its performance.

In Chapter 3, a DL framework for satellite-based precipitation estimation was developed using GOES satellite observations [23, 32, 33] as inputs and MRMS precipitation data [11] as the ground reference. The results demonstrated that the model architecture can effectively learn the nonlinear relationship between satellite measurements and precipitation within a single domain [105]. However, in practical applications, precipitation retrieval models must operate across multiple domains, such as different geographical regions, seasons, or meteorological regimes.

Each domain may exhibit distinct statistical characteristics due to variations in atmospheric conditions, surface properties, and precipitation dynamics. As a result, models trained on data from one domain may not generalize well when applied to another [106, 107]. A straightforward solution is to retrain the model using data from each new domain. However, this approach becomes impractical when domains arrive sequentially or when access to previously observed data is not practical because of the volume of the data. Furthermore, conventional retraining can lead to *catastrophic forgetting*, where the model adapts to new domains at the cost of degrading performance on previously learned domains [37, 38].

To address these challenges, this study considers the problem of **multi-domain precipitation estimation using a CL framework**. Let the dataset consist of a sequence of domains

$$\mathcal{D} = \{D_1, D_2, \dots, D_T\},$$

where each domain D_t contains input–output pairs $(\mathbf{x}, y)$. The input $\mathbf{x}$ represents satellite-derived features such as brightness temperatures and lightning observations, while the target y corresponds to precipitation estimates derived from the MRMS. The domains are introduced sequentially during training, and the model only has access to data from the current domain at each stage. Specifically, after learning domain D_t , the model should achieve strong predictive performance not only on the current domain but also on all previously observed domains $D_1, D_2, \dots, D_{t-1}$. At the same time, the learning process should remain computationally efficient and scalable for large satellite datasets.

The experiments in this chapter evaluate the model’s ability to adapt to new precipitation regimes while preserving performance in previously learned regions. In particular, we analyze cross-domain generalization, performance retention during domain transitions, and the effectiveness of null-space-constrained updates in mitigating catastrophic forgetting.

The remainder of this chapter is organized as follows. Section 7.2 introduces the study domains, data preparation procedure, DL model architecture, and sequential CL training setup. Section 7.3 presents the experimental results and evaluates the proposed CL methods across the five domains. Section 7.4 analyzes domain transitions, catastrophic forgetting, and the performance differences among the null-space-based CL methods. Finally, Section 7.5 concludes this chapter.

7.2 Study Domains and Model Training

To evaluate the proposed multi-domain CL framework for geostationary satellite precipitation estimation (MDCL-GSPE), five geographically distinct domains across the United States are se-

lected, as summarized in Table 7.1 and illustrated in Figure 7.1. These domains exhibit diverse climatological and surface characteristics, providing a challenging testbed for CL.

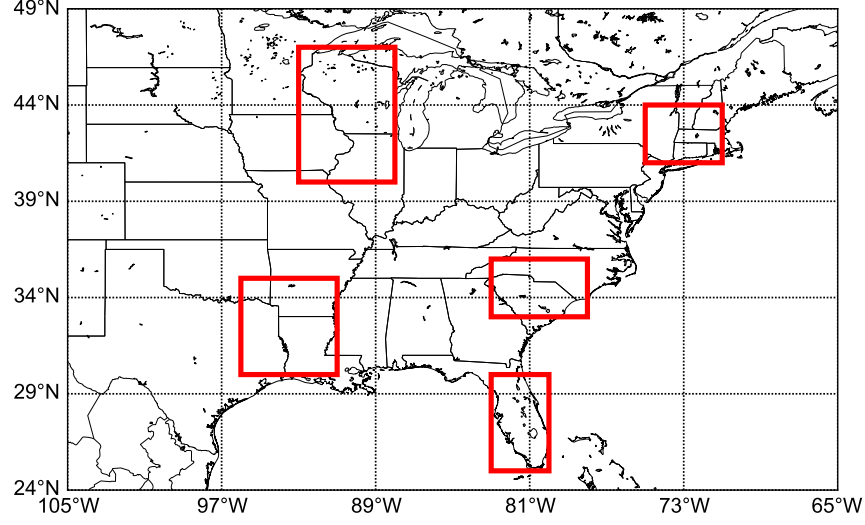


Figure 7.1: Five study domains utilized for MDCL-GSPE.

Table 7.1: Geolocation of the selected five study domains for MDCL-GSPE.

Domain	Geolocation
A	96°W to 91°W, 30°N to 35°N
B	93°W to 88°W, 40°N to 46°N
C	83°W to 80°W, 25°N to 30°N
D	82°W to 78°W, 33°N to 37°N
E	76°W to 71°W, 41°N to 44°N

The data preparation procedure of the GOES-16 ABI and GLM measurements for the DL model follows the same scheme used in Chapter 3. The overall architecture and loss functions of MDCL-GSPE also follow the framework in [105], as illustrated in Figure 3.1. The proposed DL model adopts a dual-CNN architecture consisting of a precipitation detection module (CNN-1) and a precipitation estimation module (CNN-2). CNN-1 performs binary precipitation detection, while CNN-2 estimates rainfall rates for detected precipitation voxels. For CNN-1 training, voxels are

labeled as precipitation if the corresponding MRMS rainfall rate exceeds 0.1 mm h^{-1} . For CNN-2, the MRMS rainfall rate at the voxel center serves as the regression target.

After preparing the satellite observations into voxels, we randomly selected 2.5 million voxels for each study domain. Each dataset contained an equal number of precipitation and non-precipitation voxels. This balanced representation prevents bias and inflated evaluation results, thereby enhancing the reliability and robustness of the DL model's performance in precipitation estimation. Including a disproportionate number of non-precipitation samples could lead to imbalanced testing sets, skewing metrics such as MSE and MAE and potentially inflating apparent model performance. Prior to input into the DL model, all the voxels were normalized between 0 and 1. The DL model was sequentially trained across the five domains (A to E). For CL-IPCA and CL-RLS, we randomly sampled 1 million samples to obtain the sample mean. The batch size was set to 500, and each domain's training involved 50 epochs.

For all study domains, data from May to September 2023 are used for training, while data from May to September 2024 are reserved for evaluation. The DL model is trained sequentially from domain A to domain E, where each domain is viewed as one task in the CL process. The training loss is calculated based on the output of the proposed DL model and the ground-based MRMS reference. The error is then back-propagated to update the network parameters for CNN-1 and CNN-2 separately. The model is trained using the Adam optimizer [108] with a learning rate of 2×10^{-5} for both CNN-1 and CNN-2. The batch size is set to 500, and each domain is trained for 50 epochs. For CL-IPCA and CL-RLS, 1 million voxels are randomly sampled to obtain the sample mean.

In the testing phase, the test data undergo similar pre-processing steps as the training data before being fed into the DL model. The output of the proposed DL model, i.e., the precipitation rainfall rate estimate, is derived by combining the outcomes from the dual CNNs. Specifically, if the predicted outcome from the CNN-1 module is "1" for a particular voxel, the precipitation rainfall rate estimate at the voxel center is determined using the predicted result from the CNN-2 module. Vice versa, if the outcome from the CNN-1 module is "0" for a voxel, this means no

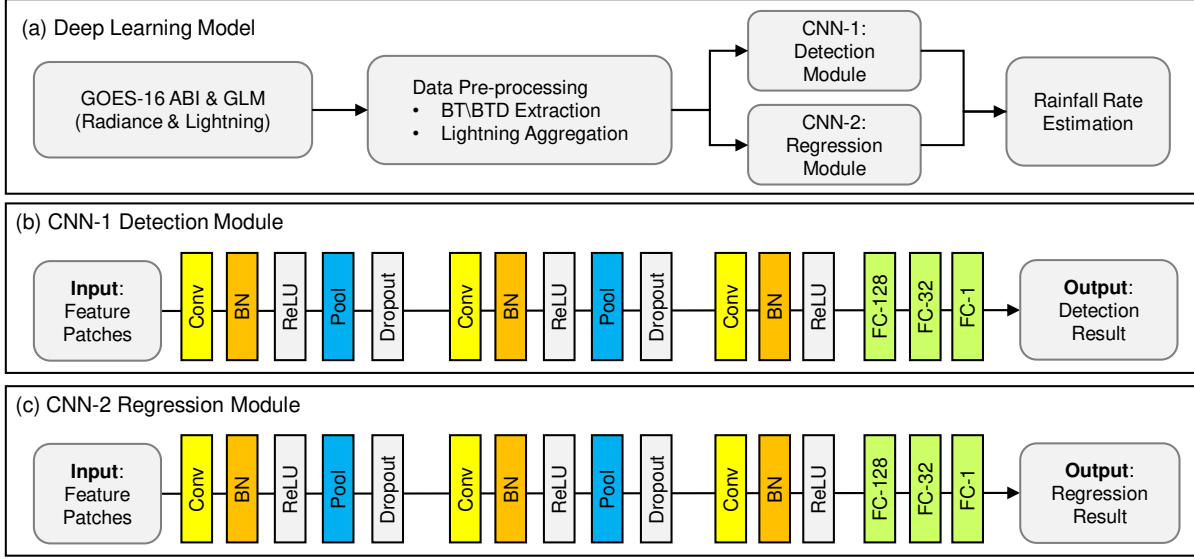


Figure 7.2: (a) The DL-based framework for geostationary satellite precipitation estimation using GOES-16 ABI and GLM measurements, which consists of (b) CNN-1 detection module and (c) CNN-2 regression module. The kernel size in the convolutional (Conv) are 4-by-4 everywhere with 64, 128, and 128 feature maps on each module in both CNN-1 and CNN-2. The mask size and stride size of pooling layers are 2 and 2 everywhere in both CNN-1 and CNN-2. The fully connected (FC) layers have 128, 32 neurons, and 2 or 1 neurons for CNN-1 or CNN-2, respectively. The rainfall rate estimates are determined by combining the detection results from CNN-1 and the regression results from CNN-2.

precipitation pattern was detected in this voxel and the corresponding precipitation estimate at the voxel center location is assumed to be 0 mm/hr.

7.3 Experimental Results and Evaluation

We trained the models using the proposed CL algorithms based on null-space methods: CL-ISVD, CL-IPCA, and CL-RLS, and compared them against established baselines, Elastic Weight Consolidation (EWC) [67] and Gradient Episodic Memory (GEM) [80].

Figure 7.3 shows sample precipitation estimation outputs from the CL-trained model alongside the MRMS ground truth, the operational GOES Rapid Refresh Quantitative Precipitation Estimation (RRQPE) product, and the IMERG product [109] for each of the five domains. When using MRMS as ground reference, the DL-based estimates clearly outperform the operational GOES and IMERG products by successfully detecting strong precipitation events that the operational prod-

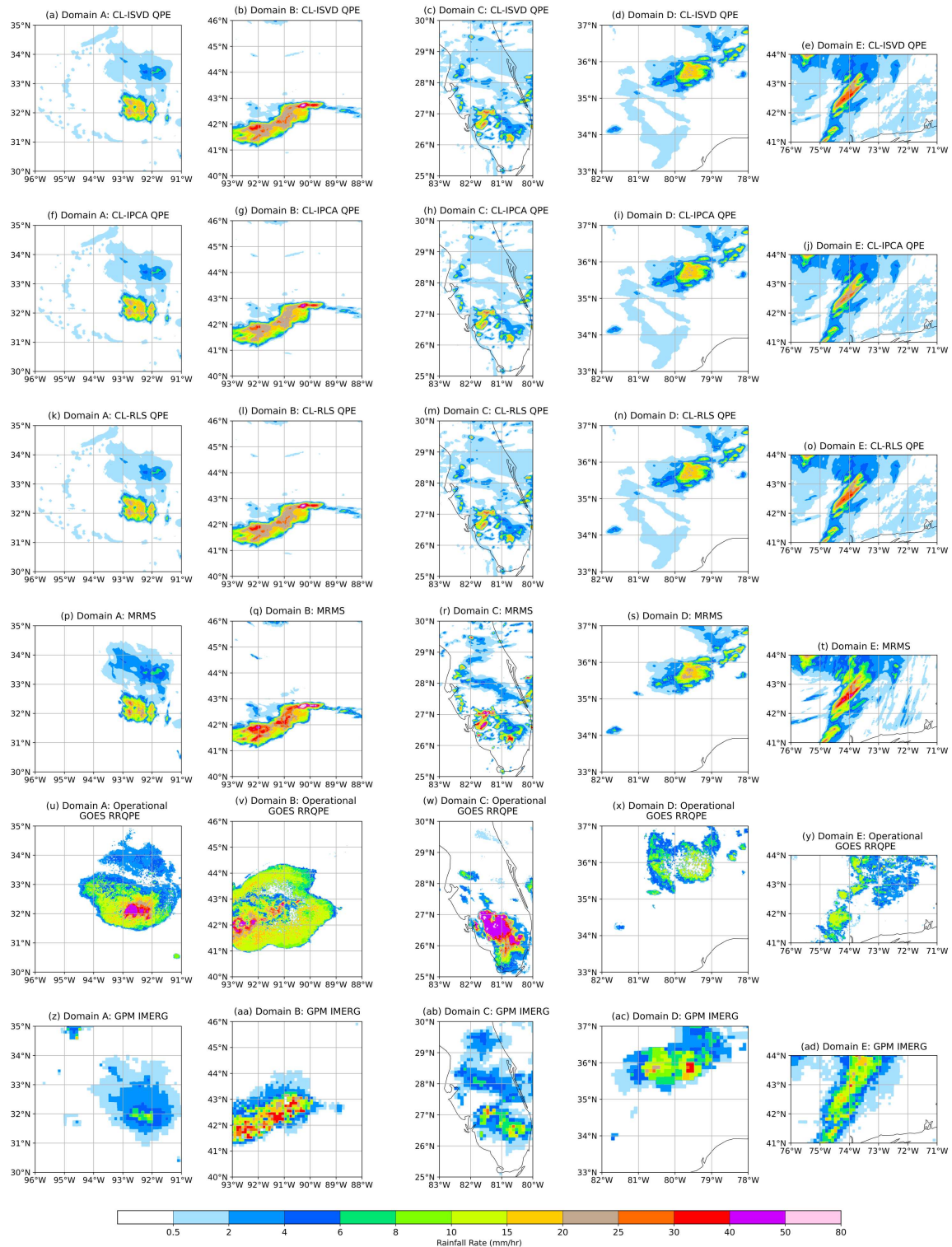


Figure 7.3: Examples of precipitation estimates over domains A, B, C, D, and E (left column to right column). Dates from left to right: August 19, 07:00 UTC; July 16, 00:00 UTC; September 10, 00:00 UTC; May 27, 03:00 UTC; August 9, 20:00 UTC, 2024.

uct missed. Moreover, all three null-space-based models accurately depict precipitation regions, despite some false alarms, highlighting the robustness of the proposed CL approach.

Table 7.2 summarizes the quantitative testing performance averaged across the five domains, comparing the three null-space methods, GEM, EWC, and the operational GOES RRQPE product. Among the proposed methods, CL-ISVD and CL-RLS achieve nearly identical performance, both outperforming CL-IPCA, which in turn outperforms GEM and EWC. The operational product lags behind all DL-based methods across all metrics.

Table 7.2: Average testing performance of precipitation retrievals from the designed DL model over all five domains.

Method	HSS	CSI	POD	FAR	RMSE (mm)	MAE (mm)	NME	NMAE
CL-ISVD	0.58	0.51	0.55	0.09	7.21	4.30	-0.41	0.61
CL-IPCA	0.57	0.48	0.53	0.09	7.95	5.04	-0.43	0.66
CL-RLS	0.58	0.50	0.55	0.09	7.38	4.77	-0.41	0.63
GEM	0.53	0.46	0.51	0.08	8.14	5.11	-0.46	0.67
EWC	0.51	0.45	0.50	0.07	8.26	5.29	-0.48	0.69
Operational	0.41	0.33	0.35	0.02	8.50	5.84	-0.60	0.86

To further investigate performance across varying precipitation intensities, we evaluated detection and estimation metrics at rainfall rate thresholds of 1, 5, 10, and 20 mm/hr, focusing only on pixels exceeding these thresholds in the MRMS ground reference. Figures 7.4 and 7.5 show detection and estimation metrics, respectively, averaged over all domains and thresholds.

The results reveal a general decline in detection performance for all models as the rainfall threshold exceeds 10 mm/hr. In particular, CL-ISVD shows the best detection skill, though its performance, like others, deteriorates for intense precipitation events. CL-RLS achieves comparable results, while CL-IPCA exhibits notable deficiencies at higher precipitation intensities. This trend highlights the inherent difficulty in accurately estimating heavy precipitation, which is often associated with complex, transient convective phenomena. The rise in RMSE and MAE at higher

thresholds underscores the challenges in modeling extreme rainfall events and indicates potential avenues for future improvement of DL-based precipitation retrieval. Interestingly, the normalized error metrics NME and NMAE decrease with increasing rainfall thresholds (Figure 7.5), primarily due to the normalization process, which scales errors relative to observed values.

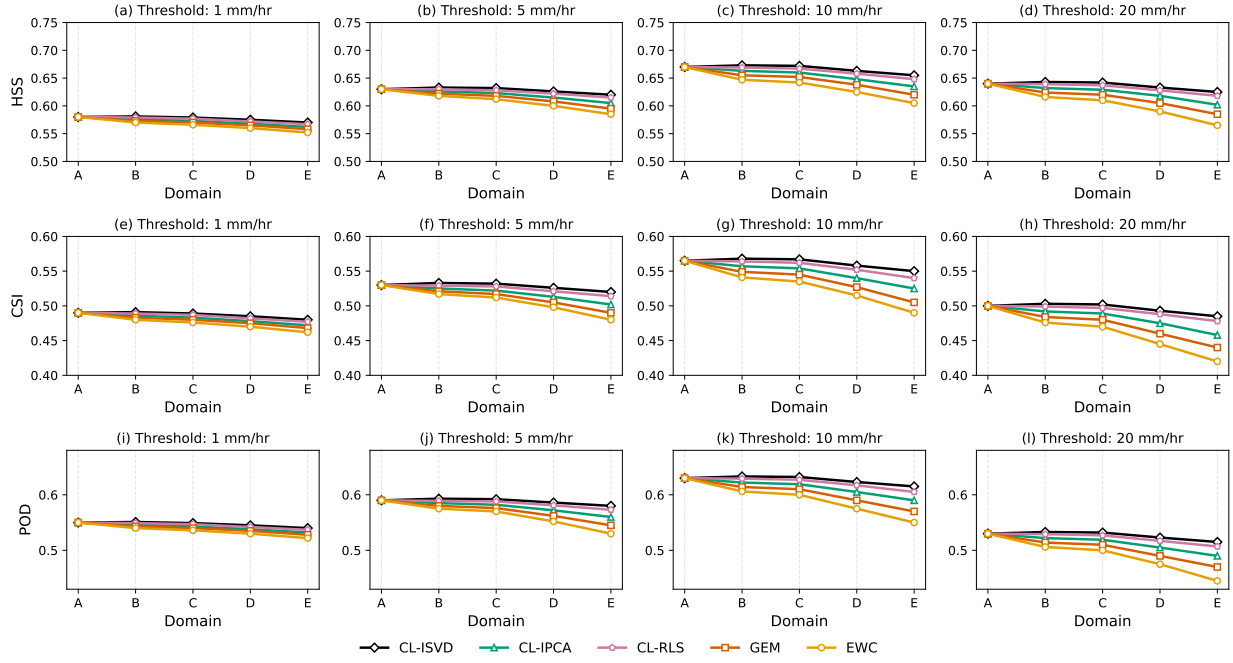


Figure 7.4: Detection metrics (HSS, CSI, POD) averaged across domains and rainfall thresholds.

7.4 Domain Transition and Forgetting Analysis

In this section, we analyze how the precipitation estimation model behaves when trained sequentially across multiple domains. In particular, we focus on the impact of domain transitions on model performance and investigate the extent of catastrophic forgetting that occurs during sequential training.

7.4.1 Performance of Baseline Model without CL

In the previous sections, we presented results based on well-trained CL models across all five domains. To further investigate the effects of catastrophic forgetting, we also trained a baseline

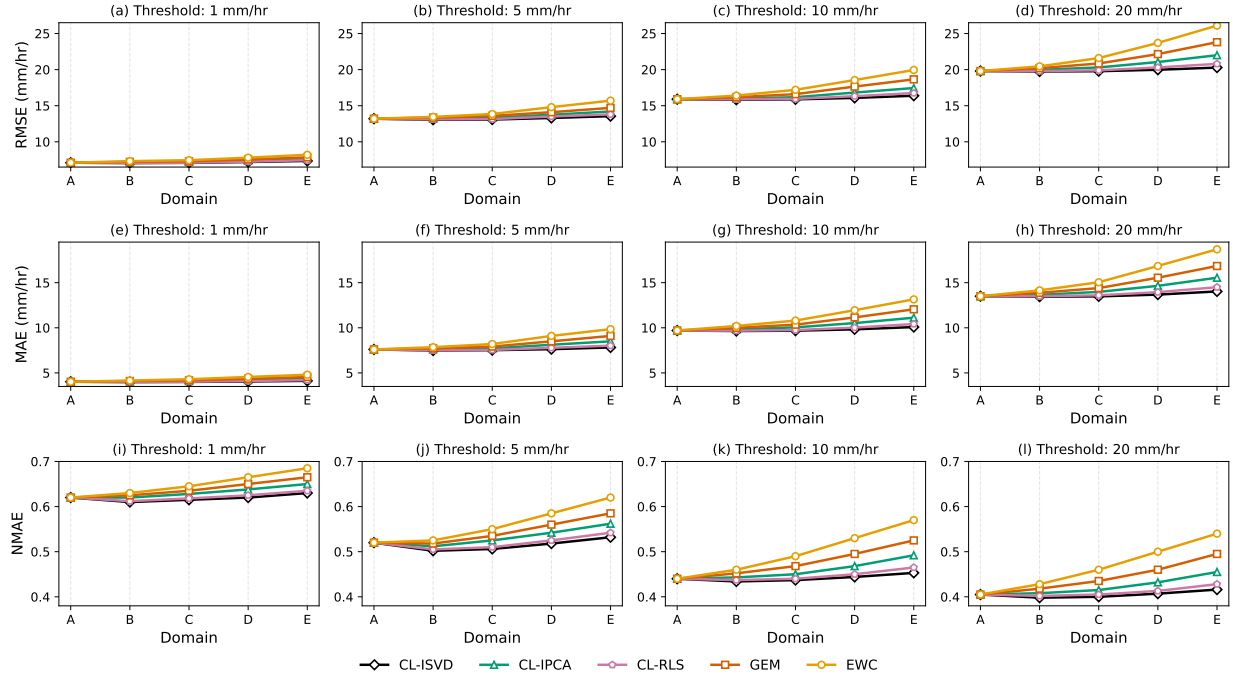


Figure 7.5: Estimation error metrics (RMSE, MAE, NMAE) averaged across domains and rainfall thresholds.

model sequentially from domain A to domain E without employing any CL strategy. After training on each domain, the baseline model was tested across all five domains using the evaluation metrics described in Section 3.3. The results are summarized in Figure 7.6.

Figure 7.6 clearly demonstrates that without CL, the model exhibits significant catastrophic forgetting as it learns from multiple regions sequentially. This effect is particularly pronounced for stronger precipitation events, as indicated by the observed decline in skill scores and increased errors over earlier domains. The degradation highlights the model’s inability to retain knowledge from previously learned domains when naively trained in sequence.

To illustrate the benefit of CL further, Figure 7.7 shows testing examples of two models: the top row presents results from a model trained only on domain A, while the bottom row shows results from a model trained up to domain C using the CL-ISVD method. As expected, the model trained solely on domain A performs poorly on domains B through E, which it has never seen during training. Conversely, the model trained sequentially with CL-ISVD shows marked performance improvements on domains B and C, demonstrating better generalization and retention. Testing

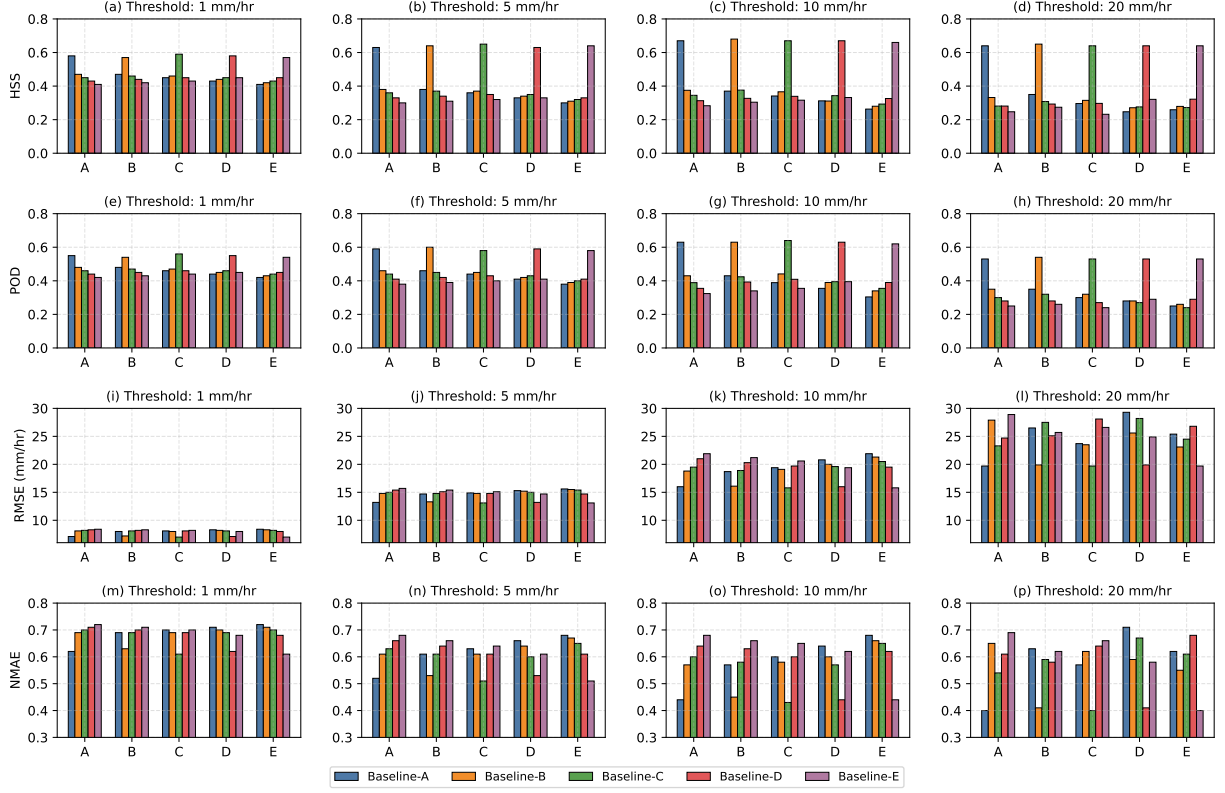


Figure 7.6: Metrics for the baseline model trained sequentially on domains A to E without CL. The degradation in performance across previously learned domains indicates catastrophic forgetting.

results for domains A to B: top row shows the model trained only on domain A; bottom row shows the model trained up to domain C using CL-ISVD. The improved detection and estimation in the bottom row highlight the importance of CL for multi-domain precipitation retrieval.

7.4.2 Comparative Analysis of Null-Space-Based CL Methods

From both the benchmark MNIST experiments and the precipitation estimation tasks, it is evident that CL-ISVD consistently outperforms CL-IPCA and CL-RLS. This suggests that CL-ISVD approximates the null space more accurately, leading to better retention of learned knowledge and reduced catastrophic forgetting. However, this improved accuracy comes at the cost of higher computational complexity.

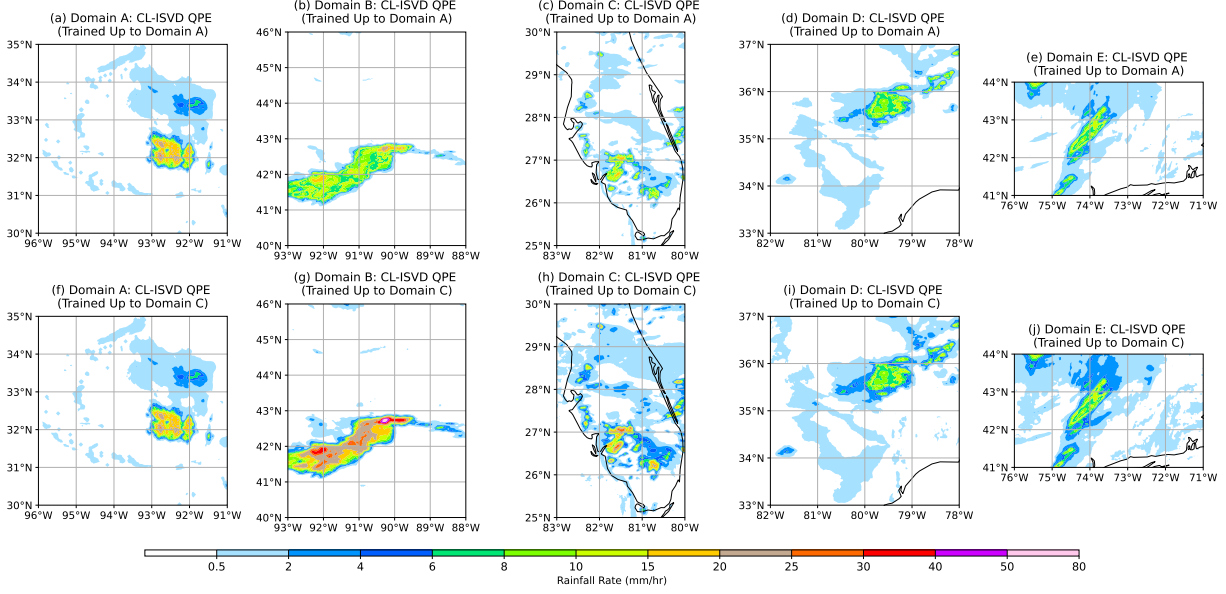


Figure 7.7: Testing results for domains A to E: top row shows the model trained only on domain A; bottom row shows the model trained up to domain C using CL-ISVD. The improved detection and estimation in the bottom row highlight the importance of CL for multi-domain precipitation retrieval.

On the other hand, CL-RLS offers a favorable trade-off by requiring significantly less computation while still delivering promising performance. CL-IPCA generally falls between the two in terms of accuracy and efficiency.

In practice, the choice among these methods should consider the computational resources available and the specific requirements of the application. When accuracy is paramount and computational resources permit, CL-ISVD is the preferred approach. In scenarios demanding efficiency and scalability, CL-RLS presents an effective alternative.

7.5 Conclusion

This chapter addressed the problem of multi-domain precipitation estimation under a CL setting and presented the overall MDCL-GSPE framework. The proposed framework enables sequential training across multiple geographic domains using a unified DL model while mitigating catastrophic forgetting caused by domain shifts.

The study domains and data processing procedures were defined, and the dual-CNN architecture for precipitation detection and estimation was described. Using GOES-16 ABI and GLM

observations as inputs and MRMS precipitation data as the ground reference, the model was sequentially adapted across five geographically distinct domains. The proposed null-space-based CL strategies, including CL-ISVD, CL-IPCA, and CL-RLS, were integrated into the training process to preserve knowledge from previously learned domains while adapting to newly introduced regions.

Experimental results demonstrated that the CL-based methods effectively reduce catastrophic forgetting during domain transitions while maintaining competitive performance on new domains. All CL-based models outperform the operational GOES-16 precipitation product, confirming the practical value of the proposed framework for satellite-based precipitation estimation.

Among the evaluated methods, CL-ISVD achieved the most consistent performance across domains, indicating that a more accurate null-space approximation leads to improved knowledge retention. Overall, the proposed MDCL-GSPE framework provides a stable and robust approach for multi-domain precipitation estimation within a single sequentially trained DL model.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

This dissertation addresses the challenge of building scalable and generalizable DL models for satellite-based quantitative precipitation estimation (QPE). While satellite observations provide broad spatial coverage compared to ground-based radars and rain gauges, conventional DL-based precipitation retrieval models are typically trained within fixed geographic domains. As a result, their performance often degrades when applied to new regions with different terrain characteristics, climatic conditions, and precipitation regimes.

To overcome this limitation, this research makes three major contributions.

First, a DL-based geostationary satellite precipitation retrieval framework was developed using GOES-16 ABI and GLM observations, with MRMS serving as ground-based reference data. The proposed architecture, consisting of dual CNNs, effectively captures nonlinear relationships between satellite measurements and surface rainfall rates.

Second, to address the plasticity–stability dilemma in sequential learning, novel null-space-based CL methods were proposed. These methods constrain parameter updates by projecting them onto the approximate null space of previously learned data, thereby preserving prior knowledge while enabling adaptation to new tasks. Three practical implementations, namely, CL-ISVD, CL-IPCA, and CL-RLS, were developed and systematically analyzed in terms of theoretical properties, accuracy, and computational complexity.

Third, the proposed CL strategies were integrated into a multi-domain geostationary satellite precipitation estimation framework. The DL model was sequentially adapted across five geographically distinct domains. Experimental results demonstrated that:

- The proposed CL-based QPE models significantly outperform the operational GOES-16 precipitation product.

- Among the three methods, CL-ISVD consistently achieved the best performance across domains due to its more accurate null-space approximation.
- The multi-domain geostationary satellite precipitation estimation framework effectively mitigates catastrophic forgetting and maintains stable performance during domain transitions.
- A single continually trained model can achieve competitive accuracy across heterogeneous precipitation regimes without retraining from scratch.

Overall, this work demonstrated that CL is a practical and effective solution for enabling DL-based precipitation retrieval systems to evolve across multiple geographic domains. The proposed framework substantially enhances cross-domain generalization and provides a scalable pathway toward hemispheric and potentially global geostationary satellite precipitation estimation.

8.2 Future Work

Although the proposed framework shows strong performance, several directions remain for further improvement that are listed below.

1. The current architecture can be extended through adaptive model expansion mechanisms, such as incremental node [110] or branch creation [111], to incorporate additional environmental predictors. Integrating terrain information [64, 112] and atmospheric variables (e.g., water vapor, temperature, and wind fields) [65] may further improve precipitation retrieval accuracy, particularly for complex storm systems.
2. Improving the computational efficiency of null-space approximation methods [102, 103, 113] will be essential for large-scale operational deployment. Developing adaptive rank-selection strategies [114] or hybrid approximation schemes could enhance scalability without sacrificing stability.
3. The CL framework provides opportunities for model transplantation to data-sparse regions, such as oceanic areas where ground reference data are limited [12]. Leveraging knowledge

learned from data-rich continental domains may enable reliable precipitation estimation in previously under-observed regions [106, 107, 115].

4. Extending the framework to hemispheric or global scales [14, 109], potentially incorporating multi-sensor observations from additional geostationary platforms [116], represents a promising direction toward next-generation satellite-based precipitation monitoring systems [21, 117].

Bibliography

- [1] Elfatih AB Eltahir and Rafael L Bras. Precipitation recycling. *Reviews of geophysics*, 34(3):367–378, 1996.
- [2] Kevin E Trenberth, Aiguo Dai, Roy M Rasmussen, and David B Parsons. The changing character of precipitation. *Bulletin of the American Meteorological Society*, 84(9):1205–1218, 2003.
- [3] Tim Barnett, Robert Malone, William Pennell, Detlet Stammer, Bert Semtner, and Warren Washington. The effects of climate change on water resources in the west: introduction and overview. *Climatic Change*, 62(1):1–11, 2004.
- [4] Ashley D Ficke, Christopher A Myrick, and Lara J Hansen. Potential impacts of global climate change on freshwater fisheries. *Reviews in Fish Biology and Fisheries*, 17(4):581–613, 2007.
- [5] Kevin E Trenberth, Aiguo Dai, Gerard Van Der Schrier, Philip D Jones, Jonathan Barichivich, Keith R Briffa, and Justin Sheffield. Global warming and changes in drought. *Nature Climate Change*, 4(1):17–22, 2014.
- [6] Gregg Greenough, Michael McGeehin, Susan M Bernard, Juli Trtanj, Jasmin Riad, and David Engelberg. The potential impacts of climate variability and change on health impacts of extreme weather events in the united states. *Environmental health perspectives*, 109(suppl 2):191–198, 2001.
- [7] Kevin E Trenberth. Changes in precipitation with climate change. *Climate research*, 47(1-2):123–138, 2011.
- [8] Caroline C Ummenhofer and Gerald A Meehl. Extreme weather and climate events with ecological relevance: a review. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 372(1723):20160135, 2017.

- [9] Julius H Kotir. Climate change and variability in sub-saharan africa: a review of current and future trends and impacts on agriculture and food security. *Environment, Development and Sustainability*, 13(3):587–605, 2011.
- [10] Peng Zhang, Junjie Zhang, and Minpeng Chen. Economic impacts of climate change on agriculture: The importance of additional climatic variables other than temperature and precipitation. *Journal of Environmental Economics and Management*, 83:8–31, 2017.
- [11] Jian Zhang, Kenneth Howard, Carrie Langston, Brian Kaney, Youcun Qi, Lin Tang, Heather Grams, Yadong Wang, Stephen Cocks, Steven Martinaitis, et al. Multi-radar multi-sensor (MRMS) quantitative precipitation estimation: Initial operating capabilities. *Bulletin of the American Meteorological Society*, 97(4):621–638, 2016.
- [12] Chris Kidd, Andreas Becker, George J Huffman, Catherine L Muller, Paul Joe, Gail Skofronick-Jackson, and Dalia B Kirschbaum. So, how much of the earth’s surface is covered by rain gauges? *Bulletin of the American Meteorological Society*, 98(1):69–78, 2017.
- [13] Christian Kummerow, William Barnes, Toshiaki Kozu, James Shiue, and Joanne Simpson. The tropical rainfall measuring mission (TRMM) sensor package. *Journal of atmospheric and oceanic technology*, 15(3):809–817, 1998.
- [14] Arthur Y Hou, Ramesh K Kakar, Steven Neeck, Ardeshir A Azarbarzin, Christian D Kummerow, Masahiro Kojima, Riko Oki, Kenji Nakamura, and Toshio Iguchi. The global precipitation measurement mission. *Bulletin of the American Meteorological Society*, 95(5):701–722, 2014.
- [15] Robert J. Kuligowski. Chapter 11 - rainfall rates from the GOES-R series. In Steven J. Goodman, Timothy J. Schmit, Jaime Daniels, and Robert J. Redmon, editors, *The GOES-R Series*, pages 125–132. Elsevier, 2020.

- [16] Gabriele Villarini, Pradeep V Mandapaka, Witold F Krajewski, and Robert J Moore. Rainfall and sampling uncertainties: A rain gauge perspective. *Journal of Geophysical Research: Atmospheres*, 113(D11), 2008.
- [17] Christian Kummerow, Y Hong, WS Olson, S Yang, RF Adler, J McCollum, R Ferraro, G Petty, Dong-Bin Shin, and TT Wilheit. The evolution of the goddard profiling algorithm (GPROF) for rainfall estimation from passive microwave sensors. *Journal of Applied Meteorology*, 40(11):1801–1820, 2001.
- [18] Sid-Ahmed Boukabara, Kevin Garrett, Wanchun Chen, Flavio Iturbide-Sanchez, Christopher Grassotti, Cezar Kongoli, Ruiyue Chen, Quanhua Liu, Banghua Yan, Fuzhong Weng, Ralph Ferraro, Thomas J. Kleespies, and Huan Meng. MiRS: An all-weather 1DVAR satellite data assimilation and retrieval system. *IEEE Transactions on Geoscience and Remote Sensing*, 49(9):3249–3272, 2011.
- [19] Robert J Kuligowski. A self-calibrating real-time GOES rainfall algorithm for short-term rainfall estimates. *Journal of Hydrometeorology*, 3(2):112–130, 2002.
- [20] George J Huffman, Robert F Adler, David T Bolvin, and Eric J Nelkin. The TRMM multi-satellite precipitation analysis (TMPA). *Satellite rainfall applications for surface hydrology*, pages 3–22, 2010.
- [21] Yang Hong, Kuo-Lin Hsu, Soroosh Sorooshian, and Xiaogang Gao. Precipitation estimation from remotely sensed imagery using an artificial neural network cloud classification system. *Journal of Applied Meteorology*, 43(12):1834–1853, 2004.
- [22] Pingping Xie, Robert Joyce, Shaorong Wu, Soo-Hyun Yoo, Yelena Yarosh, Fengying Sun, and Roger Lin. Reprocessed, bias-corrected CMORPH global high-resolution precipitation estimates from 1998. *Journal of Hydrometeorology*, 18(6):1617–1641, 2017.
- [23] Steven J Goodman, Timothy J Schmit, Jaime Daniels, and Robert J Redmon. *The GOES-R series: a new generation of geostationary environmental satellites*. Elsevier, 2019.

- [24] Timothy J. Schmit, Paul Griffith, Mathew M. Gunshor, Jaime M. Daniels, Steven J. Goodman, and William J. Lebar. A closer look at the abi on the goes-r series. *Bulletin of the American Meteorological Society*, 98(4):681 – 698, 2017.
- [25] Luyao Sun, Haonan Chen, Zhe Li, and Lei Han. Cross validation of GOES-16 and NOAA multi-radar multi-sensor (MRMS) QPE over the continental united states. *Remote Sensing*, 13(20), 2021.
- [26] H. Chen, R. Cifelli, P. Xie, and E. J. Thompson. Improving GOES-R/ABI rainfall estimates with ground-based radar observations using machine learning. In *AGU Fall Meeting Abstracts*, volume 2020, pages A142–0012, December 2020.
- [27] Haonan Chen, Robert Cifelli, and Pingping Xie. Precipitation identification and quantification using ABI and GLM observations aboard the GOES-R series: A machine learning perspective. In *AGU Fall Meeting Abstracts*, volume 2021, pages A35F–1700, December 2021.
- [28] Haonan Chen, Yifan Yang, Kyle Hilburn, and Mahmood Azimi-Sadjadi. Impacts of the GLM measurements aboard the GOES-R series satellites on severe convective precipitation detection and retrievals. In *AGU Fall Meeting Abstracts*, volume 2022, pages GC32G–0688, December 2022.
- [29] Haonan Chen, Luyao Sun, Robert Cifelli, and Pingping Xie. Deep learning for bias correction of satellite retrievals of orographic precipitation. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–11, 2022.
- [30] Russ S Schumacher and Kristen L Rasmussen. The formation, character and changing nature of mesoscale convective systems. *Nature Reviews Earth & Environment*, 1(6):300–314, 2020.
- [31] Shruti A Upadhyaya, Pierre-Emmanuel Kirstetter, Robert J Kuligowski, and Maresa Searls. Towards improved precipitation estimation with the GOES-16 advanced baseline im-

- ager: Algorithm and evaluation. *Quarterly Journal of the Royal Meteorological Society*, 148(748):3406–3427, 2022.
- [32] Timothy J Schmit, Scott S Lindstrom, Jordan J Gerth, and Mathew M Gunshor. Applications of the 16 spectral bands on the advanced baseline imager (ABI). *Journal of Operational Meteorology*, 6(4):33–46, 2018.
 - [33] Steven J Goodman, Richard J Blakeslee, William J Koshak, Douglas Mach, Jeffrey Bailey, Dennis Buechler, Larry Carey, Chris Schultz, Monte Bateman, Eugene McCaul Jr, et al. The GOES-R geostationary lightning mapper (GLM). *Atmospheric research*, 125:34–49, 2013.
 - [34] Haonan Chen, V Chandrasekar, Robert Cifelli, and Pingping Xie. A machine learning system for precipitation estimation using satellite and ground radar network observations. *IEEE Transactions on Geoscience and Remote Sensing*, 58(2):982–994, 2019.
 - [35] Yang Liu, Haonan Chen, Lei Han, and Jieying He. A two-stage learning technique for precipitation retrieval using ABI and GLM measurements on the GOES-R series. In *2021 Photonics & Electromagnetics Research Symposium (PIERS)*, pages 2851–2858. IEEE, 2021.
 - [36] Yang Liu, Haonan Chen, Lei Han, and Jieying He. A deep learning system for precipitation estimation using measurements from the advanced baseline imager (ABI) on the GOES-R series. In *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, pages 7662–7665. IEEE, 2021.
 - [37] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.
 - [38] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.
 - [39] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska,

- et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.
- [40] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.
 - [41] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
 - [42] Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 184–193, 2021.
 - [43] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR, 2020.
 - [44] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. *arXiv preprint arXiv:2103.09762*, 2021.
 - [45] Timothy J. Schmit and Mathew M. Gunshor. Chapter 4 - ABI imagery from the GOES-R series. In Steven J. Goodman, Timothy J. Schmit, Jaime Daniels, and Robert J. Redmon, editors, *The GOES-R Series*, pages 23–34. Elsevier, 2020.
 - [46] AK Heidinger, MJ Pavolonis, RE Holz, Bryan A Baum, and S Berthier. Using CALIPSO to explore the sensitivity to cirrus height in the infrared observations from NPOESS/VIIRS and GOES-R/ABI. *Journal of Geophysical Research: Atmospheres*, 115(D4), 2010.
 - [47] Yunyue Yu, Dan Tarpley, Jeffrey L Privette, Mitchell D Goldberg, MK Rama Varma Raja, Konstantin Y Vinnikov, and Hui Xu. Developing algorithm for operational GOES-R land

- surface temperature product. *IEEE Transactions on Geoscience and Remote Sensing*, 47(3):936–951, 2008.
- [48] Anastasios Papadopoulos, Themis G Chronis, and Emmanouil N Anagnostou. Improving convective precipitation forecasting through assimilation of regional lightning measurements in a mesoscale model. *Monthly Weather Review*, 133(7):1961–1977, 2005.
 - [49] Kyle A Hilburn, Imme Ebert-Uphoff, and Steven D Miller. Development and interpretation of a neural-network-based synthetic radar reflectivity estimator using GOES-R satellite observations. *Journal of Applied Meteorology and Climatology*, 60(1):3–21, 2021.
 - [50] Haonan Chen, Rob Cifelli, V. Chandrasekar, and Yingzhao Ma. A flexible bayesian approach to bias correction of radar-derived precipitation estimates over complex terrain: Model design and initial verification. *Journal of Hydrometeorology*, 20(12):2367 – 2382, 2019.
 - [51] Geoffrey J McLachlan. *Discriminant analysis and statistical pattern recognition*. John Wiley & Sons, 2005.
 - [52] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
 - [53] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.
 - [54] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. TensorFlow: a system for large-scale machine learning. In *Osdi*, volume 16, pages 265–283. Savannah, GA, USA, 2016.
 - [55] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

- [56] Joseph T. Schaefer. The critical success index as an indicator of warning skill. *Weather and Forecasting*, 5(4):570 – 575, 1990.
- [57] Yifan Yang, Haonan Chen, and Kwo-Sen Kuo. Deep learning for precipitation retrievals using combined measurements from GOES-16 and GOES-18 satellites. In *IGARSS 2024-2024 IEEE International Geoscience and Remote Sensing Symposium*, pages 1292–1295. IEEE, 2024.
- [58] Wensheng Yang, Haonan Chen, Lei Han, and Yurong Ge. Multitask learning for precipitation estimation using satellite observations from the GOES-R series. *IEEE Geoscience and Remote Sensing Letters*, 20:1–5, 2023.
- [59] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [60] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016.
- [61] ANTHONY C BERNAL AYALA, Jordan J Gerth, Timothy J Schmit, Scott S Lindstrom, and JAMES P NELSON III. Parallax shift in GOES ABI data. *Journal of Operational Meteorology*, 11(2), 2023.
- [62] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.
- [63] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coord-conv solution. *Advances in neural information processing systems*, 31, 2018.
- [64] Brian Henn, Andrew J Newman, Ben Livneh, Christopher Daly, and Jessica D Lundquist. An assessment of differences in gridded precipitation datasets in complex terrain. *Journal of hydrology*, 556:1205–1219, 2018.

- [65] Christian Barthlott and Norbert Kalthoff. A numerical sensitivity study on the impact of soil moisture on convection-related parameters and convective precipitation over complex terrain. *Journal of the atmospheric sciences*, 68(12):2971–2987, 2011.
- [66] Wenpeng Hu, Zhou Lin, Bing Liu, Chongyang Tao, Zhengwei Tao Tao, Dongyan Zhao, Jinwen Ma, and Rui Yan. Overcoming catastrophic forgetting for continual learning via model adaptation. In *International conference on learning representations*, 2019.
- [67] Xu He and Herbert Jaeger. Overcoming catastrophic interference using conceptor-aided backpropagation. In *International Conference on Learning Representations*, 2018.
- [68] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information fusion*, 58:52–68, 2020.
- [69] Nikhil Churamani, Sinan Kalkan, and Hatice Gunes. Continual learning for affective robotics: Why, what and how? In *2020 29th IEEE international conference on robot and human interactive communication (RO-MAN)*, pages 425–431. IEEE, 2020.
- [70] Khadija Shaheen, Muhammad Abdullah Hanif, Osman Hasan, and Muhammad Shafique. Continual learning for real-world autonomous systems: Algorithms, challenges and frameworks. *Journal of Intelligent & Robotic Systems*, 105(1):9, 2022.
- [71] Andri Ashfahani and Mahardhika Pratama. Autonomous deep learning: Continual learning approach for dynamic environments. In *Proceedings of the 2019 SIAM international conference on data mining*, pages 666–674. SIAM, 2019.
- [72] Fatemeh Amrollahi, Supreeth P Shashikumar, Andre L Holder, and Shamim Nemati. Leveraging clinical data across healthcare institutions for continual learning of predictive risk models. *Scientific reports*, 12(1):8380, 2022.

- [73] Amritpal Singh, Mustafa Burak Gurbuz, Shiva Souhith Gantha, and Prahlad Jasti. Class-incremental continual learning for general purpose healthcare models. *arXiv preprint arXiv:2311.04301*, 2023.
- [74] Haifeng Li, Hao Jiang, Xin Gu, Jian Peng, Wenbo Li, Liang Hong, and Chao Tao. CLRS: Continual learning benchmark for remote sensing image scene classification. *Sensors*, 20(4):1226, 2020.
- [75] John J. Hall, Mahmood R. Azimi-Sadjadi, Steven G. Kargl, Yinghui Zhao, and Kevin L. Williams. Underwater unexploded ordnance (UXO) classification using a matched subspace classifier with adaptive dictionaries. *IEEE Journal of Oceanic Engineering*, 44(3):739–752, 2019.
- [76] Martial Mermillod, Aurélia Bugaïska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects, 2013.
- [77] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR, 2017.
- [78] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154, 2018.
- [79] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.
- [80] David Lopez-Paz and Marc’Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

- [81] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. GDumb: A simple approach that questions our progress in continual learning. In *European conference on computer vision*, pages 524–540. Springer, 2020.
- [82] Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019.
- [83] R. A. Fisher. Theory of statistical estimation. *Mathematical Proceedings of the Cambridge Philosophical Society*, 22(5):700–725, 1925.
- [84] Yoshinari Takeishi, Masazumi Iida, and Jun’ichi Takeuchi. Approximate spectral decomposition of fisher information matrix for simple ReLU networks. *Neural Networks*, 164:691–706, 2023.
- [85] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pages 532–547, 2018.
- [86] Matthew Brand. Incremental singular value decomposition of uncertain data with missing values. In *Computer Vision—ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part I 7*, pages 707–720. Springer, 2002.
- [87] Hiba Fareed, John R Singler, Yangwen Zhang, and Jiguang Shen. Incremental proper orthogonal decomposition for PDE simulation data. *Computers & Mathematics with Applications*, 75(6):1942–1960, 2018.
- [88] G. H. Golub and C.F. Van Loan. *Matrix computation*. John Hopkins University Press, 1996.
- [89] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

- [90] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 4. Granada, 2011.
- [91] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [92] A Paszke. PyTorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.
- [93] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.
- [94] Fan Feng, Rosa HM Chan, Xuesong Shi, Yimin Zhang, and Qi She. Challenges in task incremental learning for assistive robotics. *IEEE Access*, 8:3434–3441, 2019.
- [95] Xiangli Nie, Zhiguang Deng, Mingdong He, Mingyu Fan, and Zheng Tang. Online active continual learning for robotic lifelong object recognition. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.
- [96] Vincenzo Lomonaco, Lorenzo Pellegrini, Andrea Cossu, Antonio Carta, Gabriele Graffieti, Tyler L Hayes, Matthias De Lange, Marc Masana, Jary Pomponi, Gido M Van de Ven, et al. Avalanche: an end-to-end library for continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3600–3610, 2021.
- [97] Wenbin Li, Shangge Liu, Borui Kang, Yiyang Chen, KaXuan Lew, Yang Chen, Yinghuan Shi, Lei Wang, Yang Gao, and Jiebo Luo. LibContinual: A comprehensive library towards realistic continual learning. *arXiv preprint arXiv:2512.22029*, 2025.
- [98] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

- [99] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019.
- [100] Yiduo Guo, Bing Liu, and Dongyan Zhao. Online continual learning through mutual information maximization. In *International conference on machine learning*, pages 8109–8126. PMLR, 2022.
- [101] Juyang Weng, Yilu Zhang, and Wey-Shiuan Hwang. Candid covariance-free incremental principal component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):1034–1040, 2003.
- [102] Bin Yang. Projection approximation subspace tracking. *IEEE Transactions on Signal processing*, 43(1):95–107, 1995.
- [103] Sami Bannour and Mahmood R Azimi-Sadjadi. Principal component extraction using recursive least squares learning. *IEEE Transactions on Neural Networks*, 6(2):457–469, 1995.
- [104] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.
- [105] Yifan Yang, Haonan Chen, Kyle A. Hilburn, Robert J. Kuligowski, and Robert Cifelli. Deep learning for precipitation retrievals using ABI and GLM measurements on the GOES-R series. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–14, 2023.
- [106] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.
- [107] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, volume 27, 2014.

- [108] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [109] George J Huffman, David T Bolvin, Dan Braithwaite, Kuolin Hsu, Robert Joyce, Pingping Xie, and Soo-Hyun Yoo. NASA global precipitation measurement (GPM) integrated multi-satellite retrievals for GPM (IMERG). *Algorithm theoretical basis document (ATBD) version*, 4(26):30, 2015.
- [110] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.
- [111] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.
- [112] Gerard H. Roe. Orographic precipitation. *Annual Review of Earth and Planetary Sciences*, 33(Volume 33, 2005):645–671, 2005.
- [113] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.
- [114] Matan Gavish and David L. Donoho. The optimal hard threshold for singular values is $4/\sqrt{3}$. *IEEE Transactions on Information Theory*, 60(8):5040–5053, 2014.
- [115] Zhaoyu Liu, Qinli Yang, Junming Shao, Guoqing Wang, Hongyuan Liu, Xiongpeng Tang, Yunhong Xue, and Linlong Bai. Improving daily precipitation estimation in the data scarce area by merging rain gauge and TRMM data with a transfer learning framework. *Journal of Hydrology*, 613:128455, 2022.
- [116] Chris Kidd and Vincenzo Levizzani. The status of satellite precipitation retrievals. *Hydrology and Earth System Sciences*, 15(4):1109–1116, 2011.

- [117] Phu Nguyen, Mohammed Ombadi, Soroosh Sorooshian, Kuo-Lin Hsu, Amir AghaKouchak, Dan Braithwaite, Hamed Ashouri, and Andrea R. Thorstensen. PERSIANN dynamic infrared–rain rate model (PDIR) for high-resolution, real-time satellite precipitation estimation. *Bulletin of the American Meteorological Society*, 101(3):E286–E302, 2020.