CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Deep Learning-Based Multimodal Satellite Precipitation Retrieval with HPC-Optimized Inference

Master's thesis in Computer Science and Engineering

Guanhua Huang

Department of Environmental and Energy Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2026

Master's thesis 2026

Deep Learning-Based Multimodal Satellite Precipitation Retrieval with HPC-Optimized Inference

Guanhua Huang

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Environmental and Energy Sciences
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2026

Deep Learning-Based Multimodal Satellite Precipitation Retrieval with HPC-Optimized Inference

Guanhua Huang

Supervisor: Ahmed Ali-Eldin Hassan, Computer Science and Engineering
Company Supervisor: Adam Dybbroe, Swedish Meteorological and Hydrological Institute
Company Supervisor: Bengt Rydberg, Swedish Meteorological and Hydrological Institute
Company Supervisor: Pouria Khalaj, Swedish Meteorological and Hydrological Institute
Examiner: Patrick Eriksson, Environmental and Energy Sciences

Master's Thesis 2026
Department of Environmental and Energy Sciences
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2026

Deep Learning-Based Multimodal Satellite Precipitation Retrieval with HPC-Optimized Inference

Guanhua Huang
Department of Environmental and Energy Sciences
Chalmers University of Technology and University of Gothenburg

Abstract

Accurate precipitation retrieval is important for weather monitoring and hydrological applications, especially where radar coverage is limited or affected by terrain. This thesis adapts a deep-learning precipitation retrieval framework to geostationary Meteosat Third Generation Flexible Combined Imager (FCI) observations and extends it with passive microwave (PMW) observations from the Advanced Technology Microwave Sounder (ATMS) and Arctic Weather Satellite (AWS). The main aim is to evaluate whether sparse PMW observations improve FCI-only precipitation retrieval over the MetCoOp Ensemble Prediction System (MEPS) domain, and to improve the efficiency of inference on a GPU-based system.

The retrieval model estimates near-surface precipitation from recent satellite observations using a 3D encoder-decoder convolutional neural network. FCI provides dense and temporally continuous geostationary input, while ATMS and AWS provide sparse PMW input, available only when polar-orbiting swaths pass over the domain. Missing PMW observations are handled with masks, allowing full-domain retrievals even when microwave coverage is absent. The models are evaluated against BALTRAD radar-derived precipitation for selected periods in 2025.

The FCI-only model captures the general precipitation structure, but its precipitation amount is not stably calibrated across the evaluation periods. The main result is that adding PMW improves several metrics: mean absolute error decreases from 2.70 to 2.14 mm/day, R^2 increases from -0.19 to 0.17, and the Matthews correlation coefficient at the 0.1 mm/day threshold increases from 0.22 to 0.35. The benefit is strongest when PMW observations are available and weaker when microwave coverage is missing. A separate zenith-angle experiment shows that viewing geometry can strongly affect the retrieved precipitation distribution, but did not give a balanced improvement.

Profiling shows that the original inference pipeline was limited mainly by data loading, small tile processing, CPU-side operations, and NetCDF writing, rather than by the 3D convolutional network alone. Batched tile inference, asynchronous I/O, and multi-GPU execution improved throughput. The optimized two-GPU pipeline achieved a 1.58× speedup for near-real-time processing, while temporal sharding achieved a 2.27× speedup for offline processing. Overall, the thesis shows that dense FCI input can be combined with sparse PMW observations, and that system-level optimizations can substantially speed up inference without changing the trained model.

Keywords: precipitation retrieval, deep learning, MTG-FCI, PMW, CHIMP, GPU.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, examiner, and everyone who supported me during this thesis project. I would like to thank Patrick Eriksson for examining the thesis and for his constructive feedback during the project. His suggestions and patience helped me greatly throughout this thesis work. I am also very grateful to my supervisors at SMHI, Adam Dybbroe, Bengt Rydberg, and Pouria Khalaj, for giving me the opportunity to work on this project and for providing valuable advice throughout the process.

I am also thankful to SMHI for providing the data, technical support, and project context that made this work possible. This project gave me the opportunity to work on a real-world application where deep learning, satellite observations, and high-performance computing meet.

I would also like to thank my friends for their support and encouragement during my master's studies. Their company and encouragement made this journey much easier, especially during the most difficult moments.

Finally, I would like to thank my family for their continuous love, understanding, and support. Even though they were far away, their encouragement always gave me strength and helped me keep going.

Guanhua Huang, Gothenburg, 2026

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background and Motivation

Accurate precipitation estimation is an important part of weather monitoring, hydrological forecasting, and warning services. Reliable precipitation information is needed for ordinary weather forecasts, but it becomes even more important during high-impact events such as heavy rainfall, snowfall, and rapidly developing convective systems. In Sweden, precipitation is also a difficult variable to monitor continuously because weather conditions vary strongly between seasons and regions. Places such as coastal areas, mountainous areas, and northern Sweden all create different observational challenges.

Weather radar is often used as a primary source of precipitation information because it provides high spatial and temporal resolution. Radar observations are especially useful for tracking precipitation systems in near real time. However, there are issues with radar-based precipitation analysis. Radar quality can be affected by factors such as distance from radar stations, beam blockage, terrain, the vertical structure of precipitation, and uncertainties in the conversion from reflectivity to precipitation rate. These problems are especially relevant in northern Sweden and mountainous regions, where radar coverage is more difficult and where operational precipitation analyses can still contain gaps or systematic errors.

Satellite observations provide a useful complement to radar. They cover large areas and can observe cloud systems even where ground-based observations are sparse. Satellite retrievals do not measure surface precipitation directly, but they provide information about the atmospheric and cloud structures that are related to precipitation. This is why satellite data are widely used in precipitation monitoring, especially in regions where radar or gauge information is limited (Kidd and Levizzani, 2011; Prigent, 2010).

The Swedish Meteorological and Hydrological Institute (SMHI) has recently started operating the CHIMP precipitation retrieval framework, a deep-learning-based system developed at Chalmers University of Technology for estimating near-surface precipitation from geostationary satellite observations. In this thesis, retrieval means estimating the precipitation field that corresponds to available observations; it is not a weather-forecasting task. The current operational direction is not to treat radar and satellite as separate products, but to combine the advantages of different observing

systems. In that setting, a neural-network retrieval model can be useful because it can learn statistical relationships between satellite observations and precipitation fields by using large collocated datasets (Reichstein et al., 2019).

The original CHIMP work was developed around geostationary observations from the Spinning Enhanced Visible and Infrared Imager (SEVIRI) on Meteosat Second Generation satellites. The introduction of Meteosat Third Generation (MTG) changes the available satellite input. The Flexible Combined Imager (FCI) can provide more spectral channels and better overall imaging capabilities. This creates an opportunity to adapt CHIMP to the MTG era and evaluate how well FCI can support precipitation retrieval over the MetCoOp Ensemble Prediction System (MEPS) domain used at SMHI (ESA, n.d.; EUMETSAT, n.d.).

At the same time, passive microwave (PMW) observations from polar-orbiting satellites provide a different and physically valuable source of information. Geostationary visible and infrared imagers mainly observe cloud-top temperature, reflected solar radiation, water vapour, and other radiative properties. Cloud-top observations give valuable but indirect clues, because a cold cloud top does not guarantee a specific rainfall rate on the surface. On the other hand, PMW observations measure the emission and scattering from actual cloud water and ice particles. This inside view provides a more direct physical link to the precipitation process. It is especially helpful during intense rainfall when ice scattering signatures give critical details hidden from standard visible or infrared views (Wilheit et al., 1994; Kidd et al., 2021).

PMW observations also introduce a practical difficulty. FCI data are dense, regularly gridded, and available at high temporal frequency. Data from PMW instruments, such as the Advanced Technology Microwave Sounder (ATMS) and the microwave sounder on the Arctic Weather Satellite (AWS), are not continuous. They are only available during satellite overpasses. At many times and locations, there is no PMW information at all. A retrieval system that uses both FCI and PMW must therefore learn two behaviours at the same time: it should use PMW when it is available, but it must still produce a full precipitation field when PMW is missing.

This thesis mainly addresses two types of challenges: scientific issues and computational problems. The scientific objective of this study is to determine whether incorporating PMW observation data can improve FCI-only precipitation retrieval, particularly in terms of the retrieval accuracy of precipitation intensity and total precipitation amount. The computational objective is to verify whether the resulting inference process can run efficiently and thus be applicable in practical applications. Therefore, this study combines satellite precipitation retrieval, multi-sensor input processing, and high-performance inference optimization.

1.2 Problem Statement

Currently, the retrieval settings for FCI have an important limitation. FCI provides frequent and spatially continuous observations, making it a strong pillar for precipitation retrieval. It can show the movement and development of cloud systems over time.

But the information it provides is mainly about cloud-top and radiative properties. In many cases these are related to surface precipitation, but the relationship is not one-to-one. Cold cloud tops do not necessarily lead to heavy precipitation, and some precipitation processes cannot be clearly displayed through infrared or visible light signals. This can result in conservative inversions, especially for local heavy rainfall.

Incorporating PMW observations helps address this issue because these sensors can capture the hydrometeors directly involved in cloud precipitation. Theoretically, adding PMW should give the model a stronger physical constraint on precipitation intensity. But in practice, the problem is harder. PMW data are very intermittent and incomplete over the target domain. The model cannot simply assume that PMW is always present. It must learn to operate under realistic conditions when the data are frequently missing.

This creates a multi-sensor fusion problem. The model must combine dense FCI sequences with sparse PMW observations and output precipitation over the full MEPS domain. The simplest way to evaluate this is input-level fusion; we provided FCI, ATMS, and AWS as input sources to the same CHIMP model. This keeps the architecture close to the existing framework and it gave a direct comparison between FCI-only and FCI+PMW retrievals. At the same time, it raises a robustness question: if the model adapts to strongly rely on PMW data during overpasses, does it degrade when PMW is missing?

There is also a system-level problem. A precipitation retrieval model intended for operational use must process large amounts of satellite data. Multi-sensor input increases the complexity of the pipeline. The system must load multiple input streams, construct temporal sequences, handle missing PMW values, run tiled inference over a large domain, assemble the output, and write NetCDF files. Even if the neural network itself is fast on a GPU, if tasks like file-system I/O or CPU-bound array reconstruction take too much time, it will ultimately affect the final execution speed (Chien et al., 2018; Pumma et al., 2019).

The main problem of this thesis is in two directions. First, it investigates whether PMW observations improve CHIMP-based precipitation retrieval compared with an FCI-only baseline. Second, it examines the computational behaviour of the inference pipeline and evaluates practical optimizations that reduce runtime on GPU hardware.

1.3 Aims and Research Questions

The overall aim of this thesis is to adapt the CHIMP retrieval framework to FCI and FCI+PMW precipitation retrieval over the MEPS domain, and to evaluate both retrieval performance and inference efficiency. The work starts from an FCI-only model, then extends the input configuration with ATMS and AWS passive microwave observations. Several model configurations are trained in order to study the effects of spatial scene size, temporal sequence length, PMW availability, and zenith-angle information. In parallel, the inference pipeline is profiled and optimized to identify the main performance bottlenecks.

The thesis is guided by the following research questions:

RQ1. How can CHIMP be adapted from SEVIRI-based retrieval to FCI and FCI+PMW precipitation retrieval?

RQ2. Does adding PMW observations improve precipitation retrieval compared with an FCI-only baseline?

RQ3. What are the main computational bottlenecks of the current inference pipeline, and what methods can improve inference performance?

1.4 Scope and Limitations

This thesis does not aim to develop a new precipitation retrieval system from scratch. Instead, it builds on the existing CHIMP code base, SMHI data archives, and preprocessing tools. The research focuses on adapting CHIMP to the selected satellite input data, training and evaluating a limited set of model configurations, and improving the inference pipeline around the model.

The data processing is also limited in scope. The satellite and reference data are assumed to be available on the MEPS/arome4km target grid. Additional preprocessing in this thesis is restricted to what is needed to inspect the data, prepare model inputs, align FCI and BALTRAD samples, add PMW sources, and handle missing PMW observations. Building a full operational production chain is outside the scope of the project.

The scientific evaluation focuses on comparing FCI-only retrieval with FCI+PMW retrieval. The three final evaluation periods are 1-14 January 2025, 14-27 July 2025, and 1-14 September 2025. These three periods are excluded from training. During training, the last three days of each month are used only as internal validation periods for monitoring model development.

The computational part focuses on inference and evaluation throughput. The main profiling and optimization work are in the retrieval phase. The project does not attempt a complete rewrite of CHIMP or a fully custom CUDA implementation. Custom kernels are considered as future work if additional speedup is needed.

Parts of the thesis language were edited with assistance from a large language model (LLM). The research design, implementation, experimental setup, analysis, results, and conclusions remain my own responsibility.

2

Background and Related Work

2.1 Satellite-Based Precipitation Retrieval

Precipitation is one of the most important variables in weather monitoring, hydrology and climate applications, but it is also one of the most difficult variables to accurately observe on a large scale. Rain gauges can provide direct surface measurement data, but their spatial coverage is limited and unevenly distributed. Weather radars can provide high-resolution precipitation fields, but the quality of radar estimations depends on the radar coverage, terrain, distance from radar stations, beam geometry, attenuation, and assumptions made when converting radar measurements into precipitation rate.

Satellite precipitation retrievals are an important supplement to ground-based observations. According to Kidd and Levizzani (2011), satellite retrievals are especially valuable because they provide precipitation information in regions where gauges and radars are sparse or unable to cover. Prigent (2010) described spaceborne precipitation retrieval as an indirect but necessary observational method, satellites do not directly measure surface precipitation, but infer precipitation through radiation brightness or brightness temperature related to clouds, precipitation particles, and atmospheric conditions.

Different satellite sensors provide different types of information. The visible and infrared observations from geostationary satellites are relatively frequent and can be used to monitor the evolution and movement of cloud systems. These observational data are helpful for monitoring weather changes, but they mainly describe cloud tops and radiation properties. The relationship between cloud top temperature, cloud texture, and near-surface precipitation is indirect. This makes infrared-based retrieval methods suitable for continuous monitoring, but their results are uncertain when the cloud top signal does not represent the precipitation below.

Kuehnlein et al. (2014) provide an example of this type of geostationary retrieval. They showed that MSG SEVIRI daytime, nighttime, and twilight observations can be used to estimate precipitation with random forests. Their work supports the idea that geostationary visible and infrared observations contain useful precipitation-related information, while also showing that statistical or machine-learning methods are needed because the relation between cloud-top radiances and surface rainfall is indirect.

Passive microwave observations provide a more direct source of precipitation in-

formation. Microwave radiation is affected by emission and scattering from liquid and frozen hydrometeors, which makes PMW observations more closely related to precipitation processes inside clouds. Many satellite precipitation products rely heavily on microwave observations when they are available. The drawback is that most PMW sensors are carried by low-Earth-orbiting or polar-orbiting satellites. They observe the Earth in swaths and they do not provide continuous temporal coverage over a fixed region.

Because of these complementary strengths and weaknesses, many operational and research precipitation products combine geostationary infrared and PMW information. Joyce et al. (2004) introduced CMORPH, which uses motion vectors derived from geostationary infrared imagery to propagate precipitation estimates obtained from passive microwave observations forward in time. Huffman et al. (2020) describe IMERG as a multi-satellite system that combines microwave precipitation estimates with infrared precipitation estimates calibrated against microwave observations to provide precipitation fields at high temporal resolution. These systems show that multi-sensor precipitation retrieval is not a new idea. What changes in this thesis is the use of a deep-learning retrieval framework to learn the relationship between FCI, PMW observations, and a gridded precipitation reference.

Huffman et al. (2007) also showed a combined-sensor idea clearly in Tropical Rainfall Measuring Mission Multi-satellite Precipitation Analysis (TMPA), where microwave estimates, microwave-calibrated infrared estimates, and gauge information were brought together to produce quasi-global precipitation fields at fine spatial and temporal scales. It shows that the main difficulty is not only detecting rain from one image, but also deciding how to merge observations that differ in sampling, physical sensitivity, and error structure.

Nguyen et al. (2018) reviewed the PERSIANN family and described it as an important early example of neural-network-based satellite precipitation estimation, with geostationary infrared imagery used as the main input and passive microwave information used to adjust the retrieval. Nguyen et al. (2020) later presented PDIR-Now as a near-real-time, quasi-global infrared precipitation dataset, which further illustrates why IR observations remain valuable even though their link to surface rainfall is indirect.

Ushio et al. (2009) also studied the combination of PMW and IR information in GSMaP, using a Kalman-filter approach to update precipitation fields propagated with infrared-derived motion information. In a later product evaluation, Gebregiorgis et al. (2018) compared Day-1 IMERG with TMPA-RT and reported improvements related to reduced missed-rain and false-rain errors. It shows how newer multi-satellite systems try to reduce the limitations of earlier retrieval products.

In this thesis, satellite precipitation retrieval is treated as a supervised learning problem. The model receives satellite observations as input and learns to estimate near-surface precipitation on a common target grid using BALTRAD as the reference product. FCI provides dense temporal context, while PMW provides intermittent but physically useful information about precipitation intensity. The rest of this chapter introduces the observation types and modelling ideas needed for this setup.

2.2 Geostationary and Passive Microwave Observations

Geostationary and passive microwave observations provide complementary views of precipitation. Geostationary imagers such as FCI observe the same region frequently and provide the temporal continuity needed to follow cloud evolution. Passive microwave sounders observe precipitation-related hydrometeor signals more directly, but only during intermittent overpasses. This complementarity is the reason for the FCI+PMW experiments in this thesis.

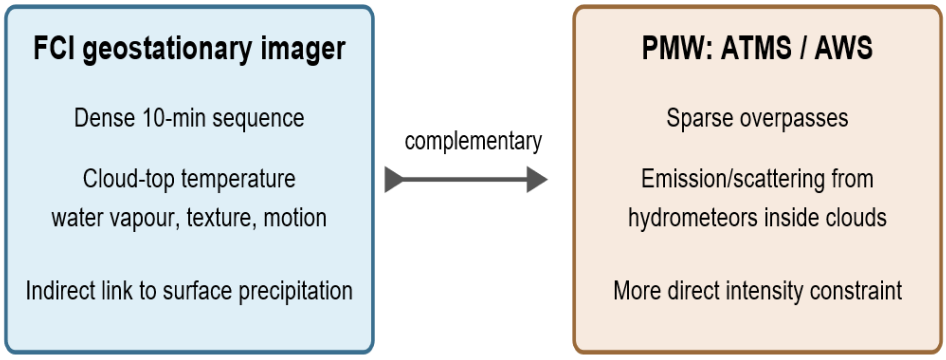

Figure 2.1: Complementary roles of FCI and PMW observations in the retrieval problem.

Figure 2.1 summarizes this role separation: FCI provides the continuous cloud-evolution context, and PMW provides intermittent but more physically direct precipitation information.

2.2.1 Geostationary Observations and FCI

Geostationary satellites are widely used in weather monitoring because they repeatedly observe the same region. Unlike polar-orbiting satellites, which pass over a location only at certain times, geostationary satellites remain fixed relative to the Earth and provide frequent observations over a large field of view. This makes them especially useful for monitoring cloud development, storm motion, and rapidly changing precipitation systems.

The Flexible Combined Imager(FCI) is the main imaging instrument on Meteosat Third Generation Imager satellites. It is the successor to SEVIRI on Meteosat Second Generation. This transition is important for this thesis because CHIMP was originally developed around SEVIRI-based geostationary input. The thesis work here adapts the retrieval setup to FCI. According to ESA (n.d.) and EUMETSAT (n.d.), FCI provides improved imaging capability for MTG, including frequent full-disk observations and rapid scanning over Europe.

Schmetz et al. (2002) introduced Meteosat Second Generation as the European geostationary system that carried SEVIRI, which explains why SEVIRI became a natural input source for earlier geostationary precipitation retrieval work. Aminou (2002)

described SEVIRI as a 12-channel imaging radiometer with a repeat cycle of about 15 minutes, providing the kind of frequent cloud and water-vapour observations that are useful for monitoring cloud evolution.

Martin et al. (2021) presented the design and development status of the FCI instrument on Meteosat Third Generation. Compared with the SEVIRI-based MSG period, FCI represents the next step in the same geostationary observing line. Therefore, adapting CHIMP from SEVIRI to FCI is not a change of retrieval philosophy, but an update to a newer European imager with richer spectral and temporal information.

FCI provides 16 spectral channels covering visible, near-infrared, short-wave infrared, water-vapour, and thermal infrared wavelengths. The WMO OSCAR entry (n.d.) lists these FCI channels and their spectral characteristics. The additional channels and improved observational capabilities are relevant for precipitation retrieval because they give the model a richer description of cloud and atmospheric conditions than older geostationary inputs.

The main strength of FCI in this project is its temporal continuity. FCI observations are available every 10 minutes in the prepared dataset, which makes it possible to construct input sequences over several hours. A temporal sequence provides information about cloud motion and evolution, rather than only the cloud state at a single time. For example, the development of convection, the advection of a rain band, or the persistence of a frontal cloud system can all provide useful retrieval information.

The main weakness of FCI is that it does not observe precipitation directly. Visible and infrared channels can indicate cloud type, cloud-top temperature, water vapour structure, and cloud evolution, but the model must learn how these signals relate to precipitation at the surface. This relationship can be ambiguous. Some cold cloud tops produce little precipitation, while some precipitation processes can be difficult to infer from cloud-top properties alone. This limitation motivates the use of PMW observations as an additional input source.

2.2.2 Passive Microwave Observations

Microwave sensors measure radiation emitted or scattered by the atmosphere, clouds, precipitation particles, and the surface. Wilheit et al. (1994) describe the physical basis of PMW rainfall retrieval: liquid hydrometeors can increase microwave emission, while frozen hydrometeors, especially ice particles in deeper clouds, can scatter radiation at higher frequencies and reduce observed brightness temperatures. This physical connection makes PMW observations valuable for precipitation retrieval. They can provide information about cloud liquid water, ice scattering, and precipitation-related structures that are not directly visible from FCI.

Kidd et al. (2021) show that cross-track microwave sounders can be used for precipitation retrieval and profiling, which is relevant here because ATMS and AWS belong to this type of observing system. In this thesis, PMW observations are provided by microwave sounders on polar-orbiting satellites. The Advanced Technology Microwave Sounder (ATMS) is flown on platforms such as Suomi-NPP, NOAA-20,

and NOAA-21. The Arctic Weather Satellite (AWS) is a newer microwave-sounding mission designed to improve observations at high latitudes. Both sensors provide brightness-temperature measurements in microwave channels that are sensitive to atmospheric temperature, humidity, cloud ice, and precipitation-related scattering.

Ferraro et al. (2000) studied precipitation characteristics over land using the NOAA-15 AMSU sensor, providing an early example of precipitation retrieval from microwave sounder observations over land. This is relevant to ATMS and AWS because these instruments also provide cross-track microwave sounding measurements rather than dense geostationary images. Kummerow et al. (2015) described the evolution of the Goddard Profiling Algorithm into a fully parametric scheme for GPM-era passive microwave retrievals, showing how sensor observations, a priori databases, and ancillary information are combined to estimate precipitation.

The limitation is coverage. ATMS and AWS observations are not dense geostationary images. They are intermittent overpasses, and at a given retrieval time only part of the target domain may be covered. In many time steps, PMW may be completely missing. This creates a different modelling problem from ordinary multi-channel image input. The model must not only learn how to use PMW, but also how to behave when PMW is absent.

PMW retrieval over land is especially difficult because the surface background is bright and variable. The land surface has a stronger and more variable microwave background than the ocean, which can make precipitation signals harder to separate from surface emission. Snow-covered surfaces and high-latitude conditions can add further ambiguity. You et al. (2015) showed that stratifying retrieval databases by surface condition and precipitation vertical structure can improve precipitation retrieval over land, while Tang et al. (2014) found that land-based PMW retrievals can show systematic biases that depend on season and precipitation intensity.

Ringerud et al. (2021) further showed that dynamic land-surface information can improve passive microwave precipitation retrieval and reduce false detections. More recent work also connects this retrieval tradition to neural networks: Pfreundschuh et al. (2022) introduced GPROF-NN as a neural-network implementation of GPROF, and Pfreundschuh et al. (2024) evaluated GPROF V7 and potential future GPROF versions against radar measurements. These studies are important background for this thesis because they show that PMW retrieval is both physically valuable and sensitive to land surface conditions, retrieval databases, and algorithm design.

2.3 Deep Learning for Precipitation Retrieval

Deep learning is useful for precipitation retrieval because the relationship between satellite observations and surface precipitation is nonlinear and context dependent. A cloud-top temperature, for example, does not map to one unique rainfall rate. The meaning of that signal depends on the surrounding cloud structure, recent cloud evolution, surface type, atmospheric conditions, and whether the precipitation is convective or stratiform. This is one reason why purely threshold-based infrared

retrievals often struggle: they can identify cloud patterns related to precipitation, but they cannot always infer the amount of precipitation reaching the ground.

Reichstein et al. (2019) argued that deep learning can be valuable in Earth system science when it is used together with physical understanding, because environmental data often contain nonlinear relationships and strong spatial-temporal dependencies. This also support that this model is not intended to replace the physical interpretation of FCI and PMW, but to learn how these observations relate to BALTRAD precipitation under realistic regional conditions.

Earlier neural-network retrieval systems already showed that data-driven retrieval could be useful. Hsu et al. (1997) introduced PERSIANN, which used neural networks to estimate precipitation from remotely sensed information. More recent systems use convolutional architectures, larger training datasets, and probabilistic outputs. Pfreundschuh et al. (2022) developed Hydronn as a neural-network-based near-real-time precipitation retrieval system using visible and infrared satellite observations. CHIMP follows the same broad direction, but in this thesis it is adapted to FCI and then extended with PMW input.

Sadeghi et al. (2019) later developed PERSIANN-CNN, showing that convolutional neural networks can use geostationary infrared and water-vapour imagery to estimate precipitation rate more directly than earlier fully connected neural-network designs. Afzali Gorooh et al. (2022) extended this direction with Deep-STEP, which combines passive microwave brightness temperatures with infrared observations in a deep neural network for high spatiotemporal precipitation estimation. These studies are close to the motivation of this thesis because they treat satellite precipitation retrieval as a learned mapping from physically meaningful radiances to surface precipitation.

2.3.1 CNNs and Encoder-Decoder Networks

Convolutional neural networks are a natural fit for satellite precipitation retrieval due to the inherently spatial structure of satellite observations. A CNN can learn local features such as cloud edges, convective cores, frontal bands, and texture patterns that are difficult to describe with hand-written rules. Deeper layers can combine these local features into larger weather structures, while early layers preserve fine spatial detail.

Encoder-decoder networks are useful when the model needs to map input images or image sequences to a full output field. The encoder gradually reduces the spatial resolution and builds more abstract feature representations. The decoder then reconstructs the precipitation field at the target resolution. This structure is common in dense prediction tasks, where the output is not a single class label but a complete gridded field.

Ronneberger et al. (2015) introduced U-Net, an encoder-decoder architecture with skip connections that became a standard reference for dense image-to-image prediction. Although U-Net was originally proposed for biomedical image segmentation, the same architectural idea is useful for precipitation retrieval because the model must preserve both large-scale context and local spatial detail.

Lin et al. (2017) introduced feature pyramid networks to build multi-scale feature representations with top-down and lateral connections. This is useful background for precipitation retrieval because rainfall systems occur over multiple spatial scales, from narrow convective cells to broad frontal bands, and a model needs features that can represent both.

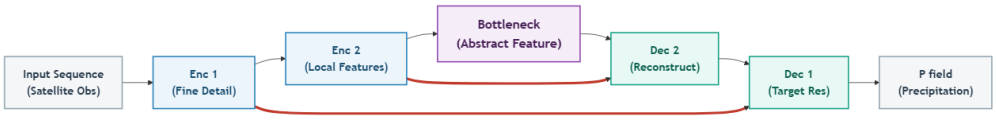

Figure 2.2: Simplified encoder-decoder structure with skip connections.

Skip connections are important in this setting. Without skip connection, the decoder has to reconstruct fine-scale spatial information only from the compressed bottleneck representation. For precipitation retrieval this can lead to overly smooth output. Skip connections pass higher-resolution encoder features directly to the decoder, helping the model keep sharper spatial structure and better location information. This matters because even a small displacement in a rain band can strongly affect grid-point verification metrics. This observation also matched the early experiments in this thesis, where disabling skip connections produced smoother and less useful retrieval fields.

2.3.2 3D CNNs and Probabilistic Outputs

For geostationary satellite retrieval, time is as important as space. A single FCI image gives the current cloud state, but a sequence shows whether clouds are growing, moving, decaying, or organizing into larger precipitation systems. Three-dimensional convolutional networks process both spatial and temporal dimensions, allowing the model to learn features related to motion, persistence, and development. Tran et al. (2015) showed that 3D CNNs can learn spatiotemporal features effectively in video-like data, and the same idea applies to satellite image sequences.

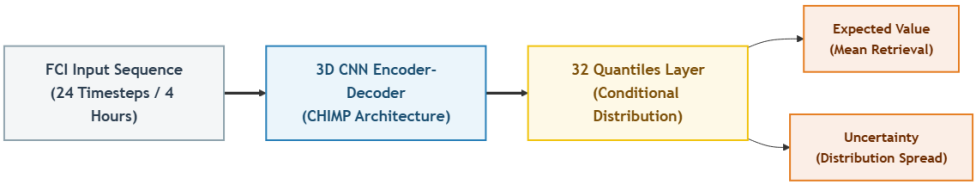

Figure 2.3: A 3D CNN uses a sequence of satellite images to learn spatial and temporal features.

CHIMP uses this type of spatiotemporal modelling for precipitation retrieval. In this thesis, the architecture is configured as an encoder-decoder 3D CNN. The model takes a sequence of satellite inputs and estimates near-surface precipitation on the target grid.

The output is probabilistic rather than purely deterministic. This is useful in precipitation retrieval because precipitation is intermittent, skewed, and uncertain.

Instead of predicting only one deterministic rain-rate value, a probabilistic model can represent a conditional distribution of possible precipitation rates. Pfreundschuh et al. (2018) showed that quantile regression neural networks can estimate posterior distributions for Bayesian remote-sensing retrieval problems and provide case-specific uncertainty information.

Moraux et al. (2019) used an encoder-decoder deep-learning model to combine geostationary satellite observations and rain-gauge measurements for precipitation estimation, showing the value of modern dense-prediction architectures in quantitative precipitation estimation. Moraux et al. (2021) further developed a multimodal deep-learning method using rain gauges, radar, and satellite infrared observations, which supports the broader idea that precipitation estimation benefits from combining complementary observation sources rather than relying on a single sensor type.

2.4 Multi-Sensor Fusion

Multi-sensor fusion is widely applied in satellite precipitation estimation because no single observation system can provide all the necessary information. Fusion can be implemented at different levels. In early fusion, the observation data from different sensors are provided together as input channels or sources to the same model. In mid-term fusion, the data from each sensor is first processed by separate feature extractors, and then the extracted feature representations are combined. In late fusion, different retrievals results are first generated separately, and then combined. Each approach has advantages and disadvantages. Mid-term or late fusion can handle the differences between sensors more clearly, but they also increase the complexity of the architecture and make it more difficult to isolate the impact of adding a single data source.

Kidd et al. (2003) provide an early example of combining passive microwave and infrared information for satellite rainfall estimation. Wang et al. (2021) later demonstrated how to use deep generative models to fuse incomplete microwave information with more complete infrared information. These studies are not exactly the same as the setup of CHIMP, but they all share the same basic motivation that passive microwave and infrared/geo-stationary orbit observations have complementary strengths, and the retrieval system must exploit their complementarity.

Sano et al. (2015) developed the Passive Microwave Neural Network Precipitation Retrieval algorithm for AMSU/MHS observations, showing that neural networks have also been used directly for cross-track microwave-sounder precipitation retrieval. Bagaglini et al. (2021) later extended this family toward climate applications with PNPR-CLIM, using long-term AMSU-B and MHS brightness-temperature records to support a global precipitation climate data record.

Turk et al. (2021) studied how passive-microwave precipitation algorithms can adapt to variable microwave land-surface emissivity within the GPM constellation. It shows that the usefulness of PMW information depends not only on whether a microwave observation is present, but also on sensor characteristics, surface background, and environmental conditions at the overpass time.

This thesis uses early fusion. FCI, ATMS, and AWS are provided as input sources to the CHIMP model, and the network learns how to fuse them. This is a practical and feasible first step, as it keeps the model closely connected to the existing CHIMP architecture and makes it straightforward to compare with the retrieval results using only FCI. If early fusion has shown that passive microwaves provide useful information, then subsequent work can explore more complex fusion architectures.

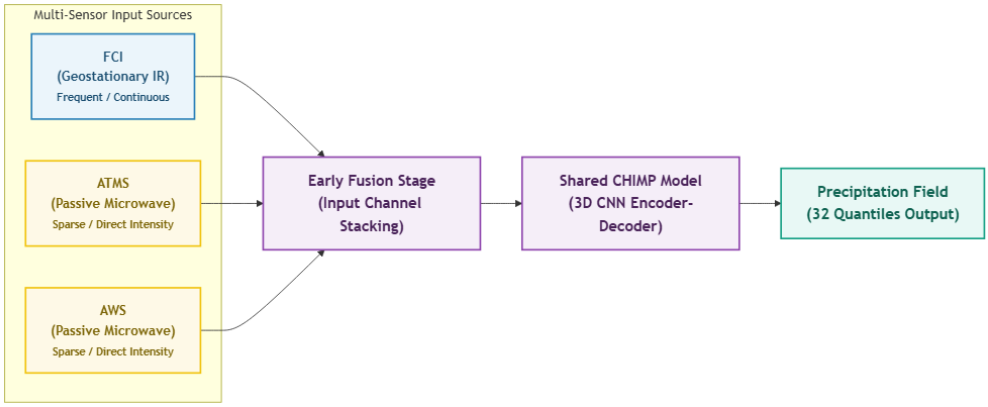

Figure 2.4: Early fusion setup used to combine FCI, ATMS, and AWS input sources in the shared CHIMP model.

The main risk of this setup lies in the lack of modal robustness. Due to the sparsity of PMW data, models trained with PMW may become dependent on it during satellite transits. If PMW is subsequently removed or becomes unavailable, the performance of the model may be inferior to that of a model that only uses FCI data and fully relies on geostationary orbit information.

Early fusion also raises a scale and resolution issue. FCI and PMW do not represent the atmosphere in the same way. FCI has dense temporal sampling and relatively fine spatial structure, while PMW has coarser and more intermittent coverage. If the model treats all channels as equally reliable gridded images, it may learn blurred or displaced structures from PMW. Therefore, missing-value masks and viewing-geometry information are not only technical details, but also part of how the input uncertainty is represented to the model.

Therefore, this thesis regards early fusion as a benchmark fusion method rather than the ultimate solution to the multi-sensor problem. This method is easy to train and compare, and it has sufficient information to verify the main assumption, whether PMW contains additional information that is useful for FCI retrieval based on CHIMP. If the early fusion model shows significant advantages and obvious failure modes, these results can guide more advanced designs, such as independent PMW encoders, attention-based fusion, or reliability-aware feature weighting.

2.5 High-Performance Computing in Deep Learning

High-performance computing is an important part of modern deep learning because both training and inference often involve large tensors, repeated numerical operations, and large data volumes. In satellite precipitation retrieval, this issue becomes more obvious because each sample is not a small independent image. The input may contain many spectral channels, several time steps, and a large spatial domain. Therefore, the runtime of the system is not only determined by the neural network forward pass, but also by data reading, preprocessing, memory transfer, output assembly, and file writing.

Previous studies have shown that data movement and I/O can become important bottlenecks in deep-learning workflows. Chien et al. (2018) showed that the performance of deep-learning frameworks depends strongly on data ingestion, CPU preprocessing, and checkpointing, not only on accelerator computation. Pumma et al. (2019) also studied scalable deep learning from an I/O perspective and showed that I/O analysis and optimization are important for deep-learning workloads on HPC systems. These studies are relevant to this thesis because the CHIMP retrieval pipeline also has to read many gridded satellite files, construct temporal input windows, and write output products.

Using a GPU does not automatically make the whole application fast. If the CPU-side pipeline cannot prepare data quickly enough, the GPU can remain idle even when the model itself is suitable for acceleration. Leclerc et al. (2023) showed this problem clearly in the context of deep-learning data loading, where data reading and processing can dominate the runtime and prevent full GPU utilization. This is also a useful way to understand satellite retrieval pipelines, improving the model computation alone may not improve the end-to-end runtime if the surrounding data pipeline is still slow.

Inference has its own performance requirements. Unlike training, inference is often evaluated by throughput and latency, because the goal is to process new input data and produce output within a practical time limit. The MLPerf Inference benchmark by Reddi et al. (2020) emphasizes that inference performance should be measured under realistic scenarios, and that different settings may require different trade-offs between latency, throughput, and batch size. For precipitation retrieval, this means that the full workflow should be considered, not only the time spent inside the neural network.

This is especially important for 3D convolutional models. Zlateski et al. (2016) studied inference throughput for 3D convolutional networks and showed that memory limits, input patch size, and CPU-GPU execution strategy can strongly affect throughput. This is relevant here because CHIMP uses a 3D encoder-decoder model and applies tiled inference over a large spatial domain. Small tiles may reduce memory use, but they can also create many small GPU workloads and repeated overhead. Larger tiles or batched tiles may improve GPU utilization, but they also increase memory pressure.

Overall, the HPC part of this thesis is treated as an application-level inference optimization problem. The aim is not to design a new GPU architecture or rewrite the model with custom kernels. Instead, the goal is to profile the CHIMP retrieval workflow, identify where the time is spent, and improve the parts of the pipeline that limit end-to-end throughput.

3

Data and Preprocessing

3.1 Study Domain and Target Grid

All experiments in this thesis use data prepared on the MEPS domain. In the prepared dataset, this domain is represented on the arome4km grid, with a horizontal grid spacing of approximately 4 km. This target grid is used in SMHI's numerical weather prediction and precipitation analysis workflow. The same grid is used for the dense MTG-FCI observations, the passive microwave observations from ATMS and AWS, and the BALTRAD radar-derived precipitation reference.

Using a common target grid is important because the input sensors have very different native observation geometries. FCI is a geostationary imager with dense and regular temporal sampling, while ATMS and AWS are polar-orbiting passive microwave sounders with swath-like and intermittent coverage. By preparing these sources on the same grid, the model can receive gridded satellite observations and predict near-surface precipitation on the same output domain. This keeps the neural-network task well defined and allows the retrieved fields to be directly compared with the BALTRAD reference and SMHI-style gridded precipitation products.

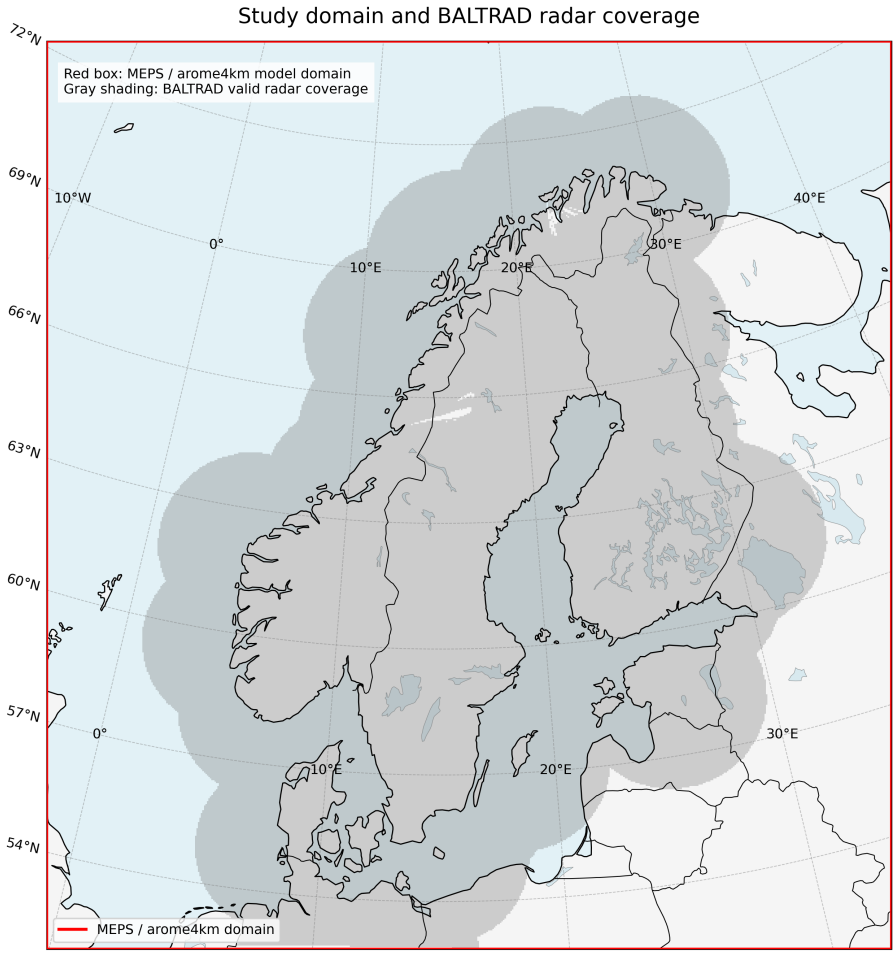

Figure 3.1: Study domain and BALTRAD radar coverage. The red box shows the MEPS/arome4km model domain used in this thesis, while the grey shading indicates valid BALTRAD radar coverage.

Figure 3.1 shows the target domain used throughout the thesis and the valid BALTRAD radar coverage inside it. The red box defines the common MEPS/arome4km grid used for the satellite inputs and retrieval output, whereas the grey shading shows where radar-derived reference data are available for training and evaluation.

3.2 FCI Data

The Flexible Combined Imager (FCI) is the main dense satellite input in this work. The FCI data are used at 10-minute intervals and are resampled onto the same MEPS/arome4km grid as the reference precipitation data. The FCI input contains 16 channels: visible, near-infrared, water-vapour, and infrared channels. In the CHIMP input configuration these channels are stored as the fci input source. The local FCI reader stacks the channel variables into a single observation tensor with channel as the last dimension before writing the processed input files.

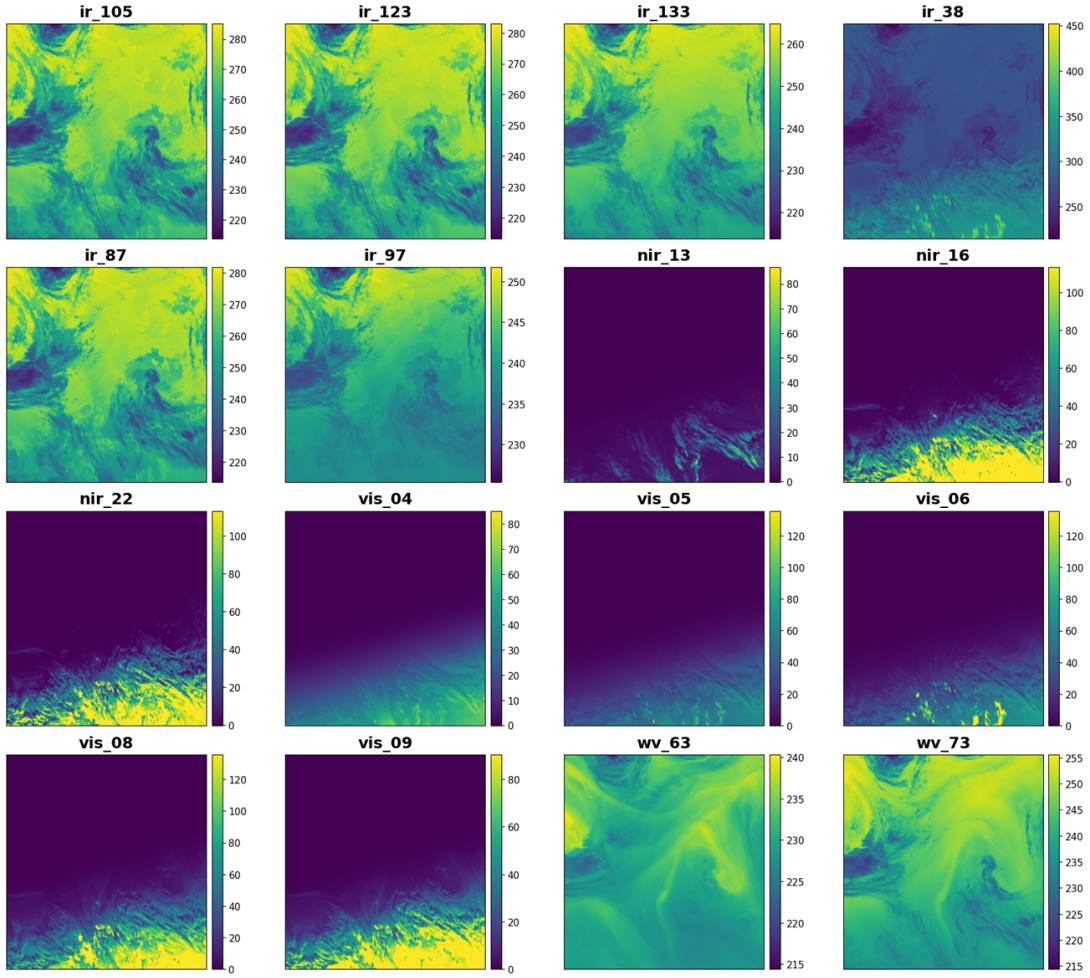

Figure 3.2: Example 16-channel FCI scene on the MEPS/arome4km grid for 2025-06-01 00:00 UTC. The visible and near-infrared channels are reflectance-like inputs, while the water-vapour and infrared channels are brightness-temperature inputs in K.

Figure 3.2 shows one example FCI scene after preprocessing on the target grid. The visible and near-infrared channels mainly show reflected solar radiation, and they contain strong day-night and illumination effects. The infrared and water vapour channels are available independently of sunlight. They provide smoother but physically useful information about cloud-top temperature and upper-tropospheric moisture patterns. The cloud systems are clearly visible in many channels, especially in the infrared and water vapour bands, and their spatial structures provide information about cloud organization and motion.

FCI is available at a high temporal frequency and is therefore well suited for constructing temporal input sequences. In this project the main FCI-only baseline uses 24 time steps, corresponding to the most recent four hours when the input sampling is 10 minutes. The same sequence length is used for the FCI-only and FCI+PMW models, which keeps the comparison fair and ensures that any difference mainly comes from the added PMW information rather than from a longer temporal context.

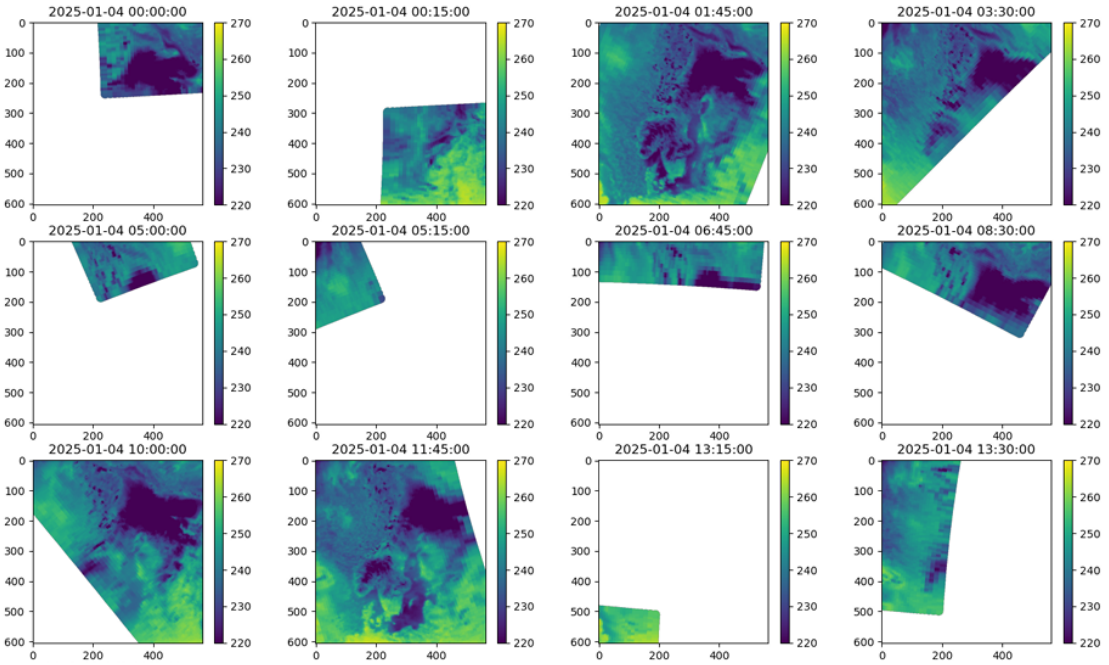

Figure 3.3: Example ATMS PMW coverage over one day for channel 17, corresponding to the 166 GHz channel. For each time slot, observations from the most recent 15 minutes are used.

This temporal context allows the 3D convolutional model to use not only the current cloud state, but also the recent motion and development of cloud systems.

3.3 PMW Data: ATMS and AWS

This thesis uses passive microwave observations from ATMS and AWS. ATMS observations are taken from three polar-orbiting satellite platforms: Suomi-NPP, NOAA-20, and NOAA-21. AWS is used as an additional microwave-sounding data source. Passive microwave observations were added through two input sources, pmw_atms and pmw_aws. These sensors provide information that is different from the visible and infrared channels of FCI. They both contain 23 channels. The no-zenith configuration uses 22 ATMS channels and 19 AWS channels.

Unlike FCI, PMW data are not available everywhere at every time step. Their observations are available only when a satellite swath passes over part of the MEPS domain. Figure 3.3 shows one day of ATMS channel 17, corresponding to the 166 GHz channel, after remapping to the target grid. The white areas indicate missing observations. Even on a day with several overpasses, the PMW input covers only parts of the domain at each time. The model therefore has to work with sparse information. In the preprocessing and training setup, PMW observations are included when available, and missing PMW coverage is represented as missing values. The CHIMP input pipeline can then build sequences where FCI remains the dense backbone while PMW acts as an additional source of physical information during overpasses.

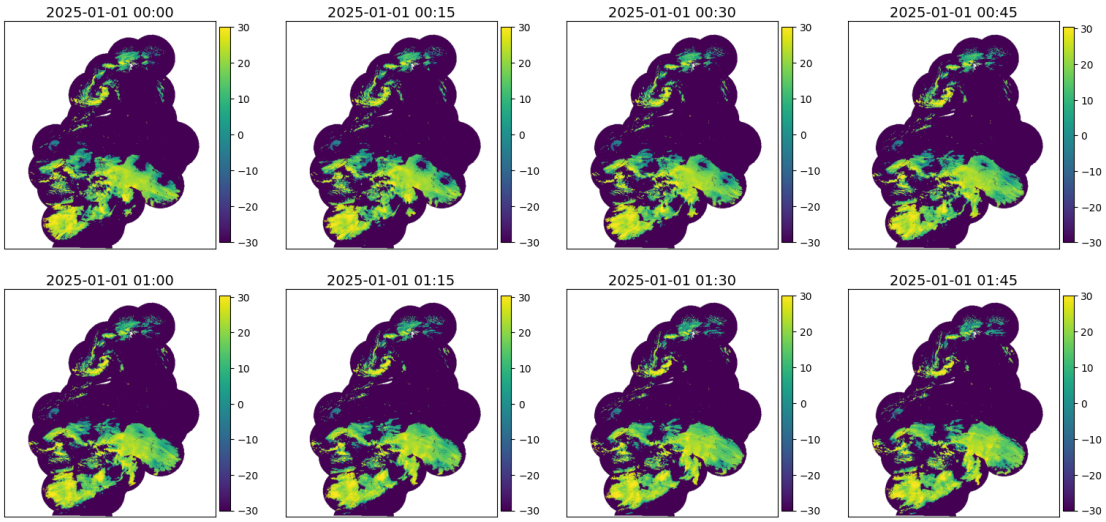

Figure 3.4: Example BALTRAD reflectivity fields on the MEPS/arome4km grid for several time steps on 1 January 2025.

3.4 BALTRAD Reference Precipitation Data

BALTRAD is used as the reference precipitation product for supervised training and evaluation. In this thesis it serves as the ground-truth target against which the model learns near-surface precipitation. The reference data are provided on the same target grid as the satellite inputs, which allows each training sample to pair a satellite input sequence with a corresponding precipitation field. In the prepared files, the main radar variable is reflectivity. This variable is used as the basis for constructing the precipitation target "surface_precip_zr" in mm/h, which is the output variable learned by the CHIMP model.

Figure 3.4 shows an example sequence of BALTRAD reflectivity fields on 1 January 2025. It contains localized precipitation structures and small-scale texture that are not directly visible in the FCI images. The figure also shows a limitation of radar-based reference data: the valid radar domain does not cover the full rectangular model grid. Areas outside the radar coverage appear as missing data and must be treated differently from clear-sky pixels. In this work, these missing regions are kept masked during preprocessing and evaluation, rather than being interpreted as zero precipitation. During training, the BALTRAD data include a quality index that is used to mask low-quality radar pixels. Pixels with a quality index below 0.8 were masked and were not used as valid training targets.

To obtain the precipitation target from the radar reflectivity field, a Z–R relationship is applied. The radar reflectivity factor Z and precipitation rate R are related by

$$Z = 200R^{1.6}$$

Here, R is the precipitation rate derived from the radar field. The conversion is applied only to valid reflectivity values, and the resulting BALTRAD-based precipitation field is used as the radar-derived reference for training and evaluation.

The use of BALTRAD also defines the evaluation target. The model is not trained to reproduce raw satellite observations, but to estimate a radar-based gridded precipitation field. Therefore, a visually plausible satellite-based retrieval can still score poorly if the precipitation feature is shifted by a few pixels relative to the reference. This displacement sensitivity is especially important for convective rain bands and localized heavy precipitation.

3.5 Temporal Alignment and Input Window Construction

The input samples used in this thesis are constructed from time sequences rather than from single satellite images. For each retrieval time, the model receives the most recent satellite observations and predicts the precipitation field for the latest available time step. The main FCI-only and FCI+PMW experiments use 24 FCI time steps, corresponding to four hours of 10-minute observations. This setting follows the project requirement that the model should be able to use multiple recent satellite time slots, rather than a single image, to estimate precipitation at the latest time.

A practical issue is that the different data sources do not use exactly the same temporal sampling. FCI is available every 10 minutes, while the BALTRAD radar reference and PMW products are organized around 15-minute quarter-hour times. FCI and BALTRAD were aligned by time using symbolic links and the CHIMP data pipeline. For instance, the BALTRAD file ending at 10:15 can also be presented to CHIMP as the target for neighbouring FCI retrieval times such as 10:10 or 10:20. This allows the CHIMP training pipeline to build samples at the FCI temporal frequency without changing the radar data themselves.

After alignment, the same temporal construction was used for both training and inference. For each retrieval time, CHIMP builds an input window from the available satellite observations. FCI provides the regular 10-minute sequence, while PMW observations are included when they are available within the corresponding time window.

3.6 Exploratory Data Analysis and Data Split

Before training the retrieval models, exploratory data analysis was carried out to check data availability, channel consistency, and temporal alignment. The purpose of this step was not to perform a full meteorological analysis of every channel, but to check whether the input data were numerically reasonable and suitable for model training. Since the model combines channels with very different physical meanings, this check was useful for finding abnormal ranges, missing values, strongly correlated features, and possible inconsistencies between the input configuration and the actual files.

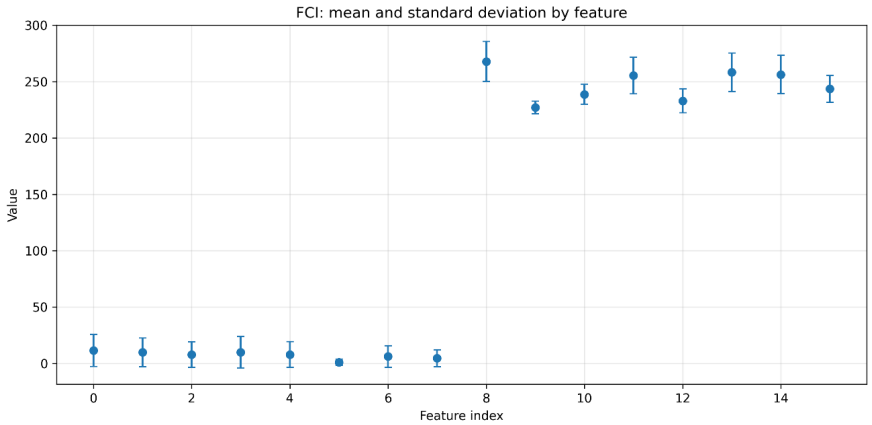

Figure 3.5: FCI mean and standard deviation by feature from CHIMP EDA

For FCI, the EDA results show a clear separation between the visible/near-infrared channels and the infrared/water-vapour channels. In Figure 3.5, the first group of features has much smaller mean values, while the later features have brightness-temperature-like values mostly between roughly 220 K and 280 K. The standard deviations also differ between channels. This shows that the raw FCI input channels are not on the same numerical scale. Therefore, the FCI input was standardized before training, as specified in the model configuration with n_features = 16, normalize = "standardize", and scale = 4.

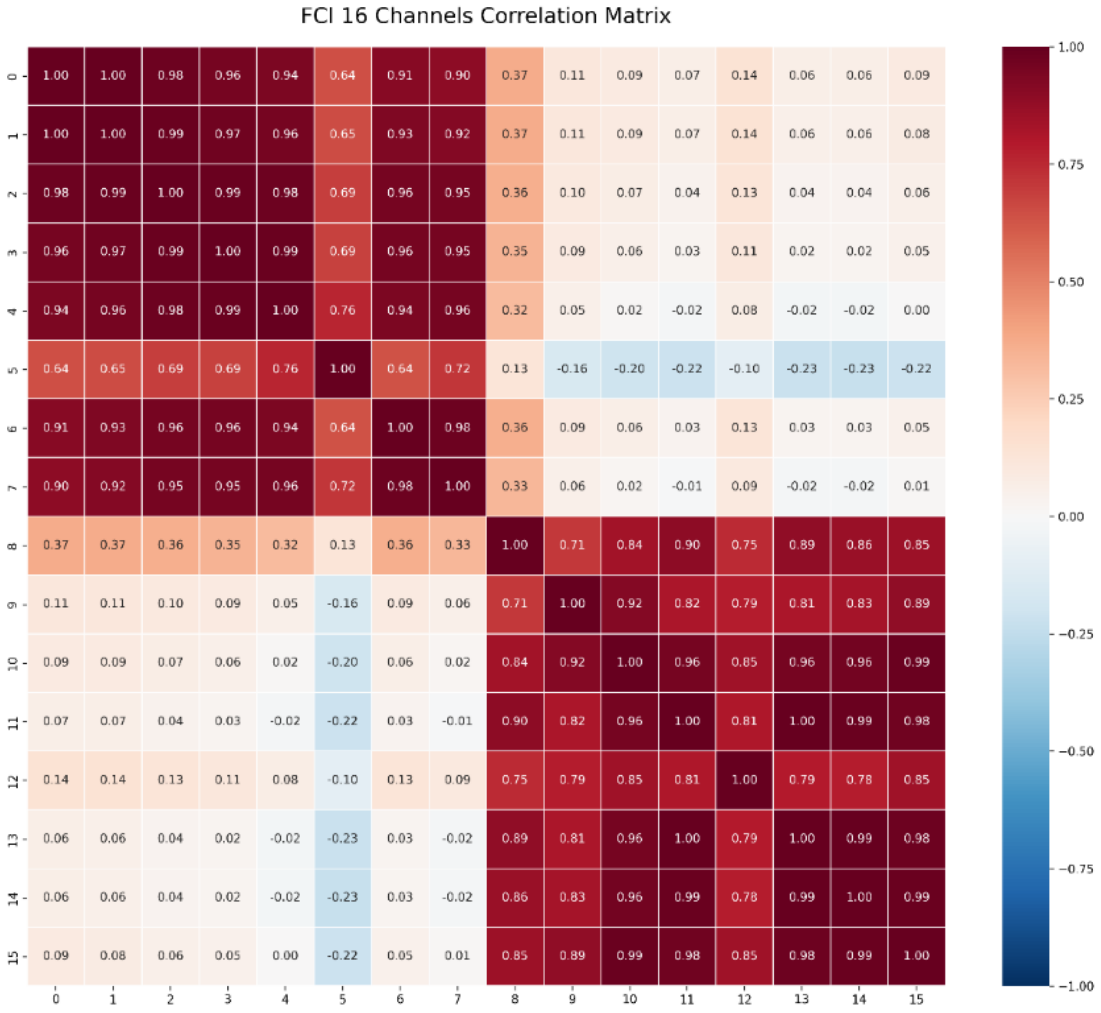

Figure 3.6: Correlation matrix of the 16 FCI input channels.

The FCI correlation matrix in Figure 3.6 also shows that several channels are strongly correlated. This is especially visible within the infrared group, where many channels have correlations close to one. The visible and near-infrared channels also form their own correlated group, but they are much less correlated with the thermal infrared channels. This is because neighbouring spectral channels often respond to similar cloud or surface properties. The high correlation suggests that some channels may contain partly redundant information, but in this thesis all 16 FCI channels were kept. This keeps the FCI-only baseline simple and consistent with the full FCI input design, instead of adding a separate feature-selection step. However, the correlation analysis also indicates a possible direction for later optimization. If GPU memory becomes a limiting factor, especially for larger scene sizes or longer temporal windows, a reduced FCI channel set could be tested to lower the input tensor size.

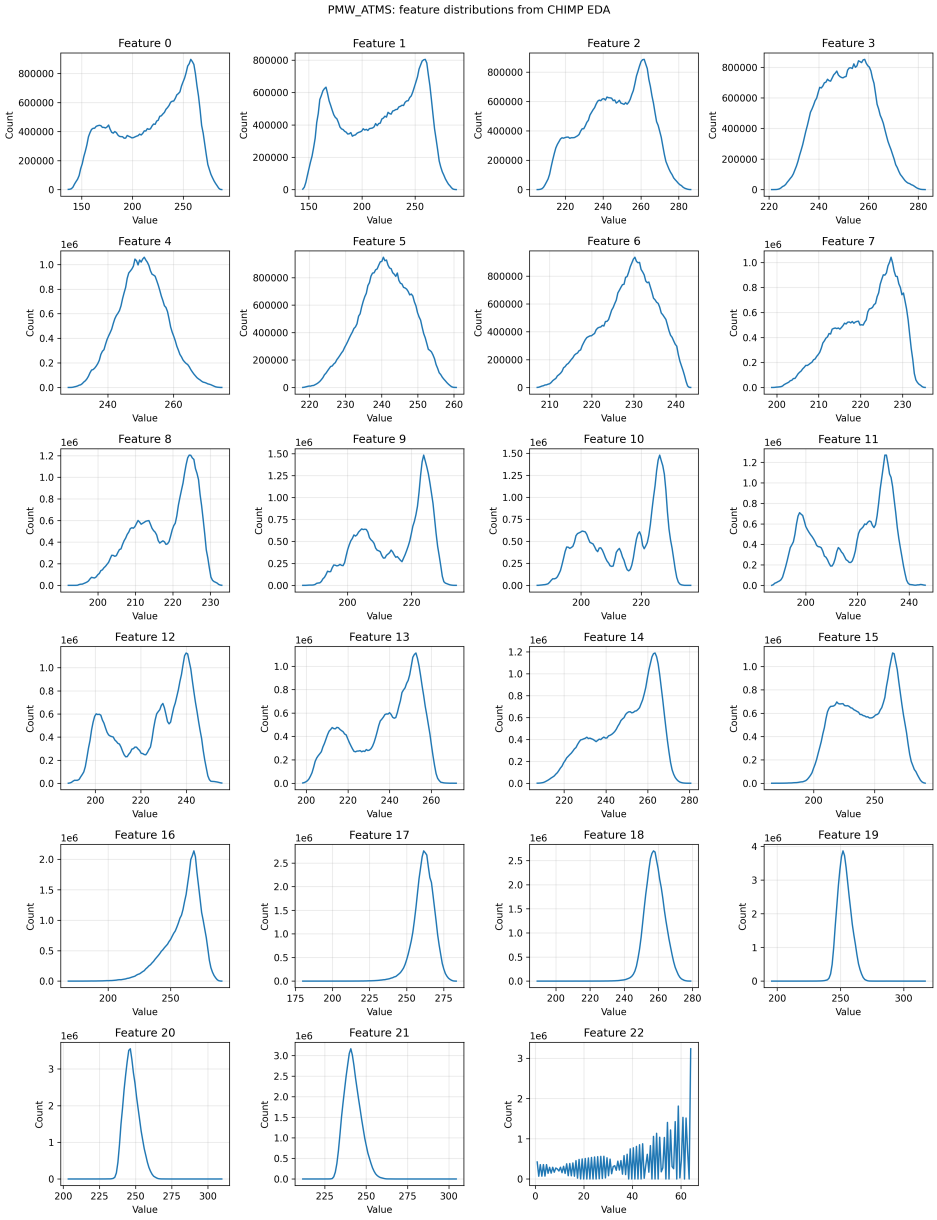

Figure 3.7: ATMS feature distributions from EDA

For PMW, the EDA was mainly used to inspect feature distributions and confirm the difference between the available feature set and the final model input. Figure 3.7 shows the ATMS feature distributions. Most microwave channels have brightness-temperature distributions with one or more clear peaks. The final feature is the satellite zenith angle or viewing zenith angle. It is the angle between the local zenith line (straight up from the ground at the specific pixel) and the line of sight to the satellite. In model FCI_PMW_V1 zenith-related channels were removed from the final training configuration. This gives 22 ATMS features and 19 AWS features in the final no-zenith model.

The training data cover the year 2025, excluding the three predefined final evaluation periods: 1-14 January 2025, 14-27 July 2025, and 1-14 September 2025. These periods

were selected because they include meteorologically interesting cases, including winter precipitation, convective summer precipitation, and a strong September rain event. During the model training phase, the last three days of each month were also held out as internal validation periods for monitoring training.

Table 3.1: Summary of training, validation, and final evaluation periods.

Subset	**Period or rule**
Training	2025 data excluding the final evaluation periods and internal validation days
Internal validation	Last three days of each month, used only to monitor model development
Final evaluation	1-14 January 2025, 14-27 July 2025, and 1-14 September 2025

4

Methodology

4.1 CHIMP-Based Retrieval Framework

The retrieval system in this thesis is based on CHIMP, a deep-learning framework for satellite precipitation retrieval developed by Chalmers University of Technology. CHIMP already provides the core components needed for this work: input dataset abstractions, sequence loading, tiled inference, encoder-decoder model configuration, quantile output handling, and command-line training and processing tools. The main work in this thesis was therefore not to replace CHIMP, but to adapt it to the selected FCI and PMW inputs and to make the inference pipeline practical for the resulting multi-sensor setup.

Figure 4.1 summarizes the adapted CHIMP-based framework used in this thesis. It can be divided into three connected layers: the data-source layer, the model and training layer, and the inference layer. The main framework adaptation was to extend the existing CHIMP data interface from a mainly geostationary setup to a multi-input setup that can also accept sparse PMW sources.

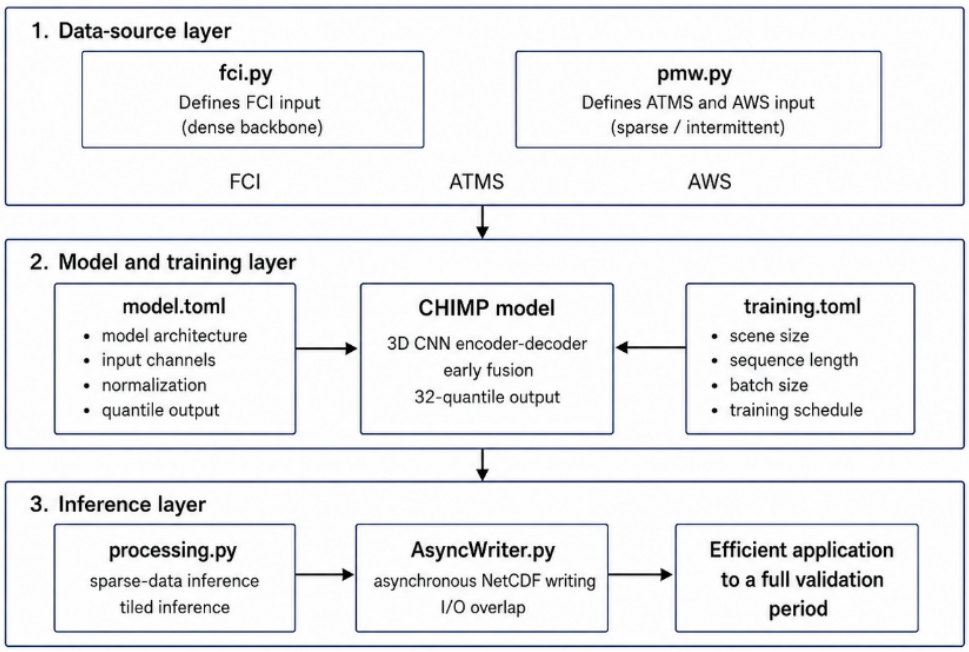

Figure 4.1: Architecture of the CHIMP-based framework used in this thesis

In the data-source layer, project-specific input readers define how the satellite data enter CHIMP. The FCI input handling is implemented in chimp/data/fci.py, where the 16 FCI channels are read and stacked into a gridded observation tensor and presented to the CHIMP input pipeline.

The PMW handling is implemented in `chimp/data/pmw.py`, which introduced ATMS and AWS as separate CHIMP input sources. This is useful because the two PMW sensors have different channel sets and different availability patterns. In the no-zenith configuration, ATMS is represented with 22 input features and AWS with 19 input features.

The model and training setups are controlled mainly by two configuration files: model.toml and training.toml. The model configuration defines the CHIMP architecture, the input sources, and the output variable. For the FCI-only experiments, the input section contains only the fci source. For the FCI+PMW experiments, the input section is extended with pmw_atms and pmw_aws. The output is surface_precip_zr, represented as a quantile output with 32 quantiles. This makes the model probabilistic internally, although the expected value of the predicted distribution is used for most deterministic evaluation metrics.

The training configuration defines how samples are drawn and how the model is optimized. The same general training schedule was used across the main model versions: a warmup stage, a main training stage, and a fine-tuning stage. The configuration files also record the scene size, sequence length, batch size, gradient accumulation, data-loader workers, and whether missing input sources are allowed.

The inference path is handled through `chimp/processing.py`, which was adapted to support sparse multi-input retrieval, allowing trained FCI+PMW models to run. Additional optimization work modified `Tiling.py` and added `chimp/AsyncWriter.py` to optimize the I/O process and introduce asynchronous writing. `benchmark.py` was used to measure runtime and throughput of the inference phase under different settings, such as different tile sizes and optimized tile batching. This was necessary because the HPC part of the thesis focuses on the full end-to-end retrieval pipeline.

4.2 Model Configurations

Three main model configurations were developed and compared. The first model used only FCI input. This model was needed as a baseline, because FCI is the dense and always-available satellite source in this project. The second model added passive microwave observations from ATMS and AWS. This model was used to test whether PMW gives extra information about precipitation intensity and total precipitation amount. A further PMW configuration included zenith-angle information, in order to test whether viewing geometry helps the model interpret swath-based microwave observations.

The main configurations are summarized in Table 4.1.

Table 4.1: Main model configurations used in this thesis

Model version	Input sources	Scene size	Sequence length	PMW zenith angle	Purpose
FCI-only v1	FCI	128	24	No	Main geostationary baseline
FCI+PMW v1	FCI + ATMS + AWS	128	24	No	Main multi-sensor fusion model
FCI+PMW with zenith angle	FCI + ATMS + AWS	128	24	Yes	Test effect of viewing-geometry information

4.2.1 FCI-Only Models

FCI_v1 used 16 FCI channels as input, with a scene size of 128 and a sequence length of 24. Since the FCI data are sampled every 10 minutes, 24 time steps correspond to four hours of recent satellite observations. This configuration follows the original idea that the model should see the current cloud field, its recent motion and development. In the configuration file, the input source is only fci, with n_features = 16, normalize = "standardize", and scale = 4. The model output is surface_precip_zr, predicted as 32 quantiles.

4.2.2 FCI+PMW Models

The second model, FCI_PMW_v1, omits zenith angles to isolate the impact of adding ATMS and AWS microwave brightness temperatures to the FCI backbone. It has 22 ATMS features and 19 AWS features. PMW input is sparse, so the training configuration sets require_input = false. This allows the model to train on samples where PMW is not available, while still learning to use PMW when it is present.

The third model, FCI_PMW with zenith angle added, keeps the same FCI+PMW structure but includes the zenith-angle channels. It was introduced as an ablation experiment to test whether viewing geometry helps the model handle swath-edge effects. Toward the edge of the swath, PMW footprints become larger and less spatially sharp. This configuration therefore adds one ATMS and four AWS zenith-angle variables to evaluate whether this information affects the retrieval.

4.3 3D CNN Architecture Design

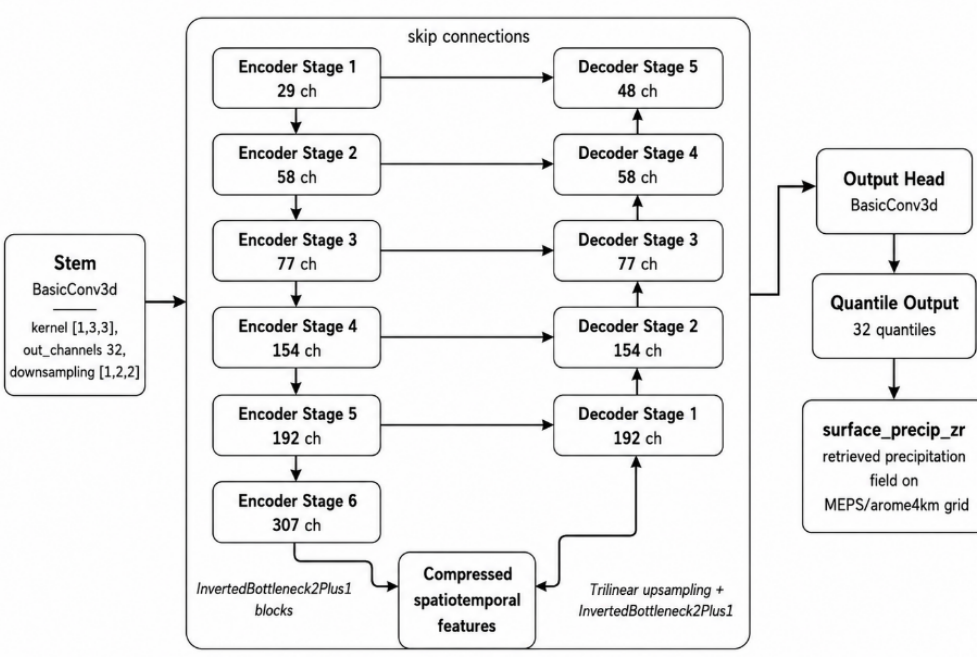

Figure 4.2: 3D CNN Retrieval Architecture used in this thesis

The retrieval model uses the encoder-decoder architecture shown in Figure 4.2. The downsampling factors are not the same at every stage. Some stages downsample both time and space, while others mainly downsample space. The input is a spatiotemporal field. If the model reduces the time dimension too quickly, it may lose information about cloud movement. If it never reduces time, the computation and memory cost become too high. The chosen setup lets the network learn motion and development over the input window, but still compresses the tensor enough for training on the available GPUs.

The main convolutional block is `InvertedBottleneck2Plus1`. This block is useful for this problem because the input is spatiotemporal. The model has to learn both spatial cloud structures and their evolution over time. In this context, 3D convolutions operate over two spatial dimensions and one temporal dimension, so the network can use information from neighbouring pixels and neighbouring time steps together. A 2+1D design separates part of the spatial and temporal processing, which can reduce the cost compared with fully dense 3D convolutions while still allowing temporal information to influence the retrieval. The architecture uses RMSNorm and GELU activations throughout the encoder and decoder. Later stages use larger expansion factors and, in deeper layers, excitation ratios to increase representational capacity.

The decoder contains five stages with channel sizes `[192, 154, 77, 58, 48]`. The purpose of the decoder is not just to make the tensor large again. It has to turn the learned cloud and PMW features into a precipitation field on the target grid. Downsampling and upsampling are applied in both spatial and temporal dimensions

depending on the stage, allowing the model to learn features at several scales. Skip connections are enabled in the final baseline architecture. With skip connections, high-resolution encoder features can be reused by the decoder, helping the output preserve precipitation structure and location.

The output head is a probabilistic quantile head with 32 quantiles for the retrieved precipitation target. Instead of predicting only one deterministic precipitation value, the model predicts a distribution represented by quantiles. For evaluation and visualization, the expected value can be used as the mean precipitation estimate. The quantile output is also useful because precipitation is uncertain, intermittent, and highly skewed, especially for heavy rainfall.

4.4 Multi-Sensor Fusion and Sparse Input Handling

The FCI+PMW model uses early fusion, where FCI, ATMS, and AWS are provided as input sources to the same CHIMP model. This choice keeps the multimodal model close to the FCI-only baseline, so the effect of adding PMW can be tested more directly. However, FCI and PMW have very different availability patterns. FCI is dense and regularly available at 10-minute intervals, while ATMS and AWS are sparse, swath-based, and often missing for a given retrieval time. The model must be able to use PMW when it exists, but still produce a complete precipitation field when PMW is not available.

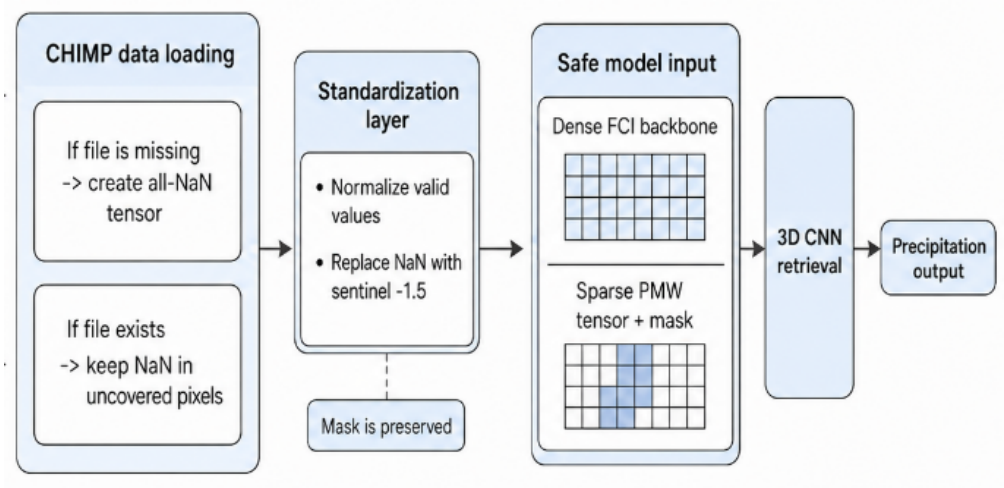

Figure 4.3: Sparse PMW Data Handling Pipeline

In the CHIMP data pipeline, missing PMW data are handled at two levels: missing time steps and missing spatial regions. As shown in Figure 4.3, the first case happens when there is no PMW file for a given reference time. During sample matching, the training data loader uses the reference precipitation time as the anchor and searches for the corresponding input files. If an input file is not found, the missing file path is marked as "None" instead of stopping the data loading process. Later, when the sample is loaded, CHIMP creates an array with the expected input shape

filled completely with NaN values. In practice, this means that a missing PMW time step is still represented as an input tensor, but the whole PMW field is marked as unavailable.

The second case is spatial missingness inside an existing PMW file. Even when an ATMS or AWS overpass exists, the instrument only observes a swath. Large parts of the MEPS domain can still be outside the swath. These unobserved regions are kept as NaN values when the PMW data are read. If a training crop contains both observed and unobserved PMW pixels, the resulting tensor contains valid microwave values in the covered region and NaN values in the missing region.

At the same time, the missing values are not simply forgotten. If NaN values enter the network directly, they can propagate through the feature maps and make the output invalid. During standardization, valid input values are normalized in the usual way, while non-finite values are replaced by a sentinel value. In this implementation, missing values are replaced by -1.5 after standardization. The value was chosen as a simple sentinel in the standardized feature space: it is clearly below the normal centre of the distribution, but not so extreme that it creates very large numerical values in the tensor. It therefore acts as a placeholder for missing observations while the separate mask records which pixels are truly missing.

Then, the framework does not simply treat these -1.5 values as ordinary measurements. When the NaN values are replaced, a boolean mask is also created to record where the original data were missing. The data tensor and the mask are then combined into a MaskedTensor. This mask allows the framework to distinguish between real observation data and values that were inserted only to make the tensor safe for GPU computation. This design allows the model to receive tensors with a consistent shape, even when the real PMW coverage changes from one sample to another. The mask is also used later in the learning process so that placeholder values do not contribute as real target information. During masked loss calculation, positions marked as missing can be excluded from the error calculation.

From the modelling point of view, the dense FCI sequence acts as the stable backbone of the retrieval. PMW acts as an additional physical constraint when it is present. During PMW overpasses, the model can use microwave information that is more directly related to hydrometeors and precipitation intensity. When PMW is missing, the PMW input source becomes a masked and safely filled tensor, and the prediction has to rely mainly on FCI. This behaviour is important for a practical retrieval system, because a model that only works during PMW overpasses would not be useful for continuous precipitation estimation.

4.5 Training Setup

All main models were trained with the same general training procedure. Training was designed and organized into three stages, as shown in Figure 4.4. The first stage was a warmup stage with 10 epochs. In this stage, the learning rate was set to 1e-3. This stage uses a sample rate of 1, meaning that each available training scene was sampled approximately once per epoch. It also used AdamW as the optimizer.

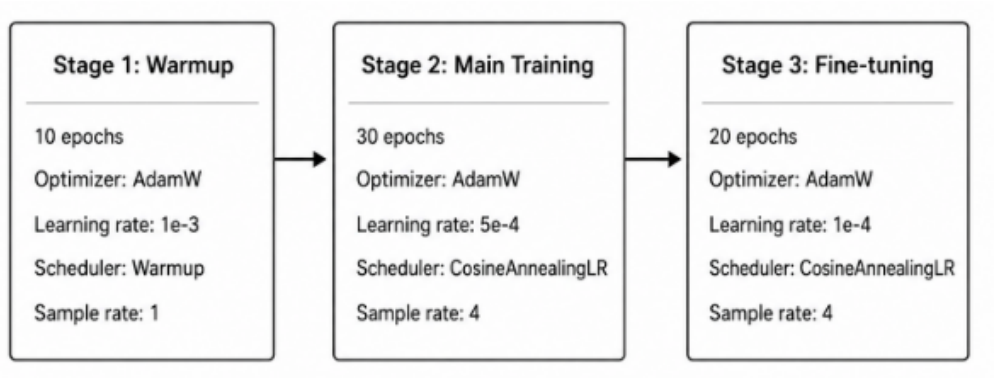

Figure 4.4: Three-stage training structure

The purpose of this stage was to let the model quickly learn a stable first mapping between the satellite inputs and the BALTRAD precipitation target. At this point the model does not need to be very precise yet. It mainly needs to move away from random output and learn the basic scale of the target variable.

The second stage was the main training stage. It used 30 epochs with a lower starting learning rate of 5e-4. This stage was where the model learned most of the useful spatiotemporal features, such as cloud motion, cloud organization, and the relationship between satellite signals and radar-derived precipitation. The third stage was a fine-tuning stage with 20 epochs. The learning rate was reduced again to 1e-4. This stage was used to make smaller updates to the network after the main structure of the retrieval had already been learned. In practice, this helped avoid making large parameter updates late in training, when the model was already close to a useful solution.

The batch size was set to 1 for the main experiments. This was mainly due to memory limits. Each sample is not a small image; it contains a multi-channel time sequence. For the main FCI-only model, one sample contains 16 FCI channels over 24 time steps on a 128×128 scene. To make training more stable, gradient accumulation was set to 4. This means that the model updates its weights after accumulating gradients over several mini-batches, while still keeping each individual forward pass small enough to fit into GPU memory.

The number of data-loader workers was set to 6. This was used to load and prepare samples in parallel while the GPU was training. The data pipeline had to read multiple gridded files, construct temporal windows, crop scenes, and handle missing PMW values. Using several workers helped reduce the waiting time between training steps, although data loading was still an important part of the total runtime.

The same basic training schedule was used for the FCI-only and FCI+PMW configurations, with changes only where required by memory and input availability. For the PMW models, `require_input = false` was used because PMW observations are sparse and should not be required at every time step.

Training was mainly carried out on the Minerva cluster using NVIDIA L40S and L4 GPUs. The L40S GPUs were more suitable for the larger or heavier training runs because of their larger memory. The L4 GPUs were useful for smaller runs and later

inference benchmarking. Mixed precision was used where supported by the training environment, which helped reduce memory use and speed up tensor operations. Because cluster jobs could be interrupted after long runtimes, checkpointing was adjusted so that checkpoints were saved more frequently, reducing the risk of losing progress between stages.

The training curve and metrics monitored by TensorBoard during training. The metrics included correlation coefficient, bias, and MSE for surface_precip_zr. These metrics were used mainly to follow training progress and check whether the model was learning a meaningful retrieval.

4.6 Evaluation Metrics

The retrieval models were evaluated with several metrics instead of only one score. This is necessary because precipitation retrieval has different kinds of errors. A model can predict the correct total amount of precipitation but place the rain band slightly too far east. It can also locate the rain area correctly but underestimate the intensity. In both cases, one metric alone would give an incomplete picture. For this reason, the evaluation in this thesis uses metrics for total precipitation amount, pixel-wise error, spatial agreement, and event detection.

All metrics were computed against the BALTRAD-based reference precipitation field on the MEPS/arome4km grid. During evaluation, missing radar regions were not treated as clear sky. They were kept masked, so that areas outside the valid radar domain did not affect the metric values.

The first metric is volume error. It measures whether the model overestimates or underestimates the total precipitation amount over the evaluated domain and time period. It is computed as

$$Volume\ Error = (\sum P_{\text{pred}} - \sum P_{\text{ref}}) / \sum P_{\text{ref}}$$

Here, Ppred is the retrieved precipitation and Pref is the reference precipitation. A negative value means that the model underestimates the total precipitation mass, while a positive value means overestimation. In earlier experiments, FCI-only retrievals often looked conservative, so volume error is a direct way to test whether the model is losing too much water overall.

The second metric is mean absolute error per day, written as MAE/day. This metric measures the average absolute difference between the retrieved and reference precipitation fields after aggregation to daily values. A lower MAE/day means that, on average, the retrieved daily precipitation is closer to the BALTRAD reference. This metric is useful because small local displacement errors do not dominate it as strongly as some grid-point metrics at high temporal resolution.

The third metric is coefficient of determination, or R^2. This metric measures how much of the variance in the reference precipitation is explained by the model output.

In this thesis, R^2 is mainly used as a measure of how well the model reproduces the spatial and temporal variability of precipitation. A higher R^2 means that the model follows the reference field better. However, precipitation is sparse and noisy, so R^2 can be low even when the model captures the main rain system. This is why it is used together with other metrics.

The fourth metric is Pearson correlation. Correlation measures whether high and low values in the model output tend to occur in the same places and times as high and low values in the reference. It is less sensitive to absolute bias than volume error. For example, a model can underestimate the total precipitation amount but still have a reasonable correlation if it places the rain systems in roughly the correct regions. This makes correlation useful for judging whether the model has learned the large-scale precipitation structure.

The fifth metric is Matthews Correlation Coefficient, or MCC. MCC is used here as an event-detection metric after applying precipitation thresholds. Two thresholds are used: 0.1 mm/day and 1.0 mm/day. The 0.1 mm/day threshold tests whether the model can separate precipitation from almost no precipitation. The 1.0 mm/day threshold focuses more on stronger precipitation events. MCC is useful because it takes hits, misses, false alarms, and correct negatives into account in one score. This makes it more balanced than simple accuracy, especially because most grid points may have little or no precipitation.

Mean Squared Error (MSE) was also monitored during training. It measures the average squared difference between the predicted and reference precipitation values. Compared with MAE, MSE gives larger weight to larger errors.

During training, TensorBoard was used to monitor both optimization behaviour and retrieval-related validation metrics. The logged quantities included training loss, validation loss, mean loss, learning rate, and bias, correlation coefficient, and MSE for surface_precip_zr. These metrics were computed on the internal validation data used during training. They were not used as the full final evaluation, but they were useful for checking whether the model was learning in the expected direction during training.

The learning-rate curve was used to confirm that the warmup and cosine-annealing schedules were applied correctly. The training and validation losses were used to check whether the model was still improving or beginning to overfit. MSE for surface_precip_zr gave a direct indication of the average squared error on the precipitation target. Bias for surface_precip_zr was used to see whether the model systematically overestimated or underestimated precipitation. Correlation coefficient for surface_precip_zr was especially useful because precipitation retrieval is not only about matching the exact intensity at every pixel, but also about placing rain systems in roughly the correct regions.

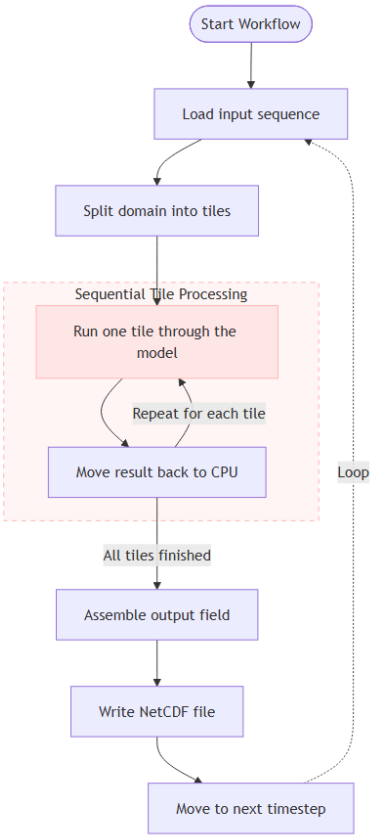

Figure 4.5: Original baseline inference workflow

4.7 Inference Optimization

The optimization work followed a simple order. First, the original pipeline was benchmarked. Then Nsight Systems was used to identify where time was spent. After that, several changes were tested: tile-size tuning, batched tile inference, input prefetching, asynchronous output writing, and time-domain multi-GPU inference. The goal was not to rewrite CHIMP completely, but to reduce the main bottlenecks with practical changes to the existing pipeline.

4.7.1 Baseline tiled inference

The original CHIMP inference pipeline splits the MEPS/arome4km domain into overlapping tiles so it can fit on the available GPUs. Each tile is processed separately, and the outputs are then assembled into the final precipitation field.

Figure 4.5 shows that this approach is safe and memory-friendly, but it has a performance problem. The tiles are processed one by one, so each forward pass is small. The GPU receives only a limited amount of work at a time. This can create repeated Python overhead, repeated host-to-device transfers, and repeated synchronization between CPU and GPU. As a result, the GPU may spend much of the time waiting instead of computing.

This baseline was used as the reference for all later optimization experiments. The same model, input period, output settings and computation cards were used when comparing optimized versions, so that the measured speedup came from the inference pipeline changes rather than from changing the retrieval task.

4.7.2 Nsight Systems profiling

Before changing the code, Nsight Systems was used to profile the inference pipeline. The reason was that slow inference does not always mean slow GPU kernels. In many deep-learning workflows, the bottleneck is outside the model itself. It can come from file access, Python control flow, data loading, tensor copying, CPU-side assembly, or output writing.

The profiling was carried out on the inference phase after the FCI+PMW pipeline had been made runnable. At that point, the model could produce retrievals, but processing a validation period was still slow. For example, the retrieval over 14–27 July took more than eight hours before optimization. This made it necessary to inspect the CPU and GPU timeline instead of only looking at final runtime. The Nsight trace was used to check when the CPU was active, when CUDA kernels were running, when host-to-device copies happened, and when the GPU was waiting. This helped analyze whether the bottleneck came from GPU computation or from the surrounding data pipeline.

One might assume that the main problem would be the 3D CNN itself, since the model is large and uses spatiotemporal tensors. However, the profiling result suggested that the model kernels were not the first bottleneck. The system was limited by pipeline stalls and small workloads, not just by raw GPU compute. In the original implementation, tiles were processed one at a time. This created many small forward passes and repeated transfers between CPU and GPU, which increased the overhead. The profiling also showed that output writing could block the inference loop. In the original pipeline, one timestep followed a mostly serial order, which increased the overhead.

Based on the profiling, the optimization work focused on three practical problems. First, the GPU needed larger workloads, which motivated batched tile inference. Second, input loading needed to be overlapped with computation, which motivated input prefetching. Third, NetCDF writing should not block the main loop, which motivated the asynchronous writer. The detailed performance changes from these optimizations are reported later in the results chapter.

4.7.3 Tile-size benchmarking

After Nsight Profiling, the first experiment was to test different tile sizes during inference. Tile size is an important parameter because it changes the amount of work sent to the GPU in each forward pass.

If the tile is too small, each model call contains only a small amount of computation. In this case, the GPU is not fully used even though the model itself can run quickly.

A larger tile gives the GPU more work per forward pass. This can increase GPU utilization because each kernel launch processes more pixels. It can also reduce the number of tiles needed to cover the full domain. However, larger tiles are not always better. They use more GPU memory, increase the cost of each forward pass, and can create more repeated computation in overlapping border regions. If the tile becomes too large, the GPU may appear busy, but the end-to-end throughput can still become worse.

For this reason, tile size was treated as a trade-off rather than a value to maximize. The benchmark tested several tile-size settings using the same trained model, the same input period, and the same output configuration. For each setting, the benchmark recorded total runtime, throughput, average GPU utilization, maximum GPU utilization, and GPU memory utilization.

4.7.4 Batched tile inference

The next optimization was to change the tile processing strategy from tile-by-tile inference to batched tile inference. In the original strategy, the code had to prepare the input tensor for each tile, move it to the GPU, run the model, move or store the output, and then continue to the next tile. When this is repeated many times for every retrieval timestep, the overhead becomes large.

To reduce this overhead, the tiling loop was modified so that several spatial tiles are collected into one batch before they are passed to the model. Instead of calling the model once per tile, the optimized version calls the model once per tile batch. In the implementation tested here, a batch size of 32 tiles was used for the main batched-tile run. This keeps the batch large enough to improve GPU occupancy, while still fitting within GPU memory.

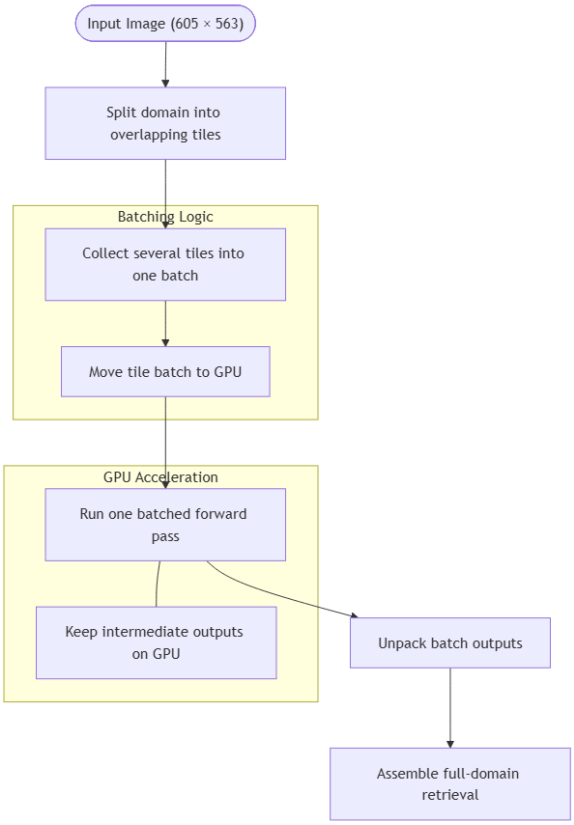

Figure 4.6: Batched tile inference workflow

The batch inference structure is shown in Figure 4.6. The model still receives input tensors with the same channel structure and predicts the same precipitation variable. The difference is only in how the inference workload is scheduled. The batch dimension is used to process several spatial tiles together. Because of this, the scientific retrieval task is unchanged, but the GPU receives larger and more regular work.

A key detail in the implementation was to avoid unnecessary synchronization inside the tile loop. Operations such as .cpu() and .item() can force the program to wait for the GPU to finish before continuing. If these calls happen after every tile, they reduce the benefit of batching. In the optimized version, the inference loop was split into two phases. In the first phase, tile batches are sent to the GPU and computed with as little CPU synchronization as possible. In the second phase, after the batch output is ready, the results are unpacked and inserted into the full output field.

This design reduces overhead in several ways. First, it reduces the number of model calls per timestep. Second, it reduces the number of kernel-launch groups caused by many small tile forwards. Third, it gives the GPU a larger workload, which improves occupancy. Fourth, it makes CPU-GPU synchronization less frequent, because the code does not need to stop after every single tile.

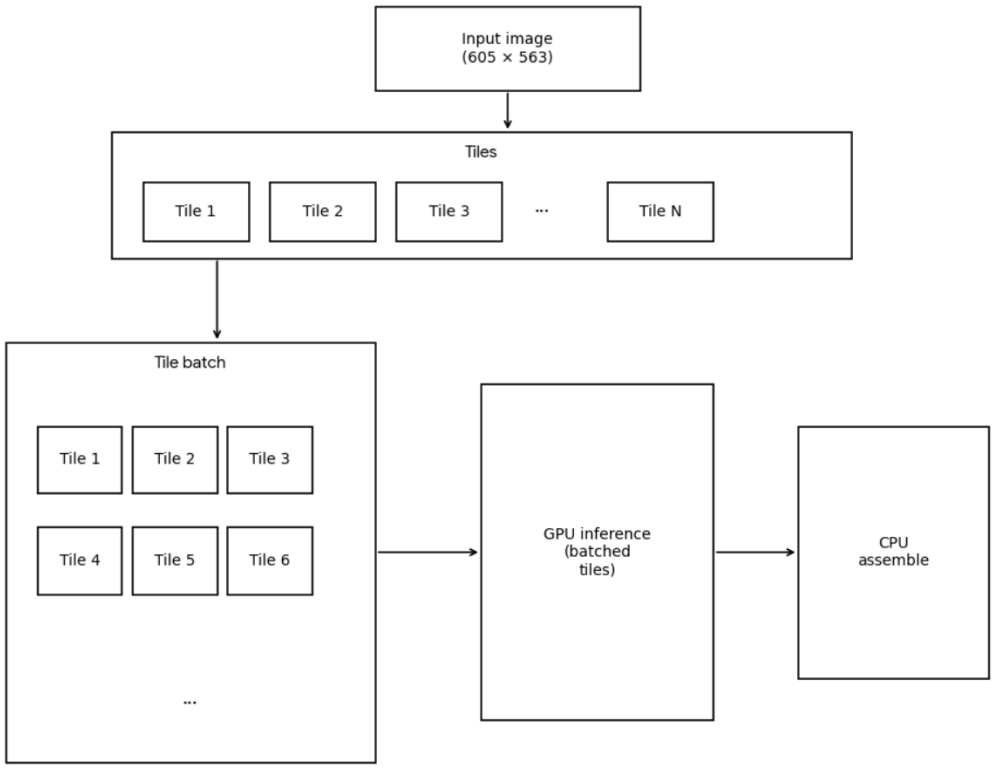

Figure 4.7: Schematic diagram of batched tile inference

The schematic diagram is shown in Figure 4.7. Instead of only increasing tile size, the pipeline was changed to process several moderate-sized tiles together as one batch. This tile-to-batch optimization solves a specific bottleneck found during profiling: the GPU was receiving too many small tile workloads. By batching tiles, the inference pipeline becomes closer to the kind of workload GPUs are designed for, without changing the trained CHIMP model or the precipitation output format.

4.7.5 Input Prefetching

After batched tile inference, the next bottleneck was input loading. Even if the GPU computation becomes faster, each retrieval timestep still needs a full input window before inference can start. For the main model, this input window contains the most recent 24 FCI frames. This means that the CPU has to read several files, build the temporal sequence, handle missing PMW input, and prepare tensors before the GPU can run the model.

In the original serial pipeline, these steps happen one after another, as shown in Figure 4.8.

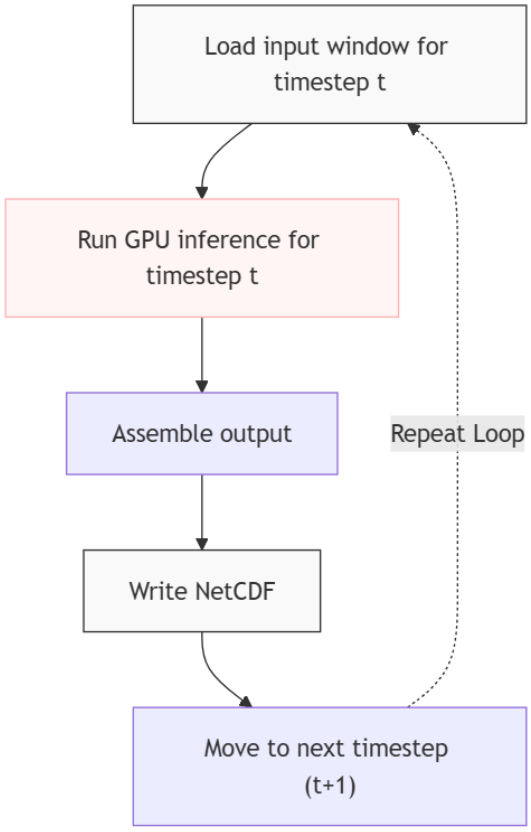

Figure 4.8: Original serial pipeline workflow

This structure is simple, but it creates waiting time. The GPU can only start inference after the input window has been fully loaded and prepared. When the CPU is reading files or checking whether inputs exist, the GPU may have no useful work to do. The Nsight profiling showed this problem clearly, that there were periods where the CPU was busy with file-system operations while the GPU stayed mostly idle.

To reduce this waiting time, input prefetching was added. While the main thread runs inference for timestep t, a background thread starts preparing the input window for timestep t+1. When the current inference step is finished, the next input may already be ready or partly ready. This does not remove the cost of input loading, but it hides part of that cost behind GPU computation.

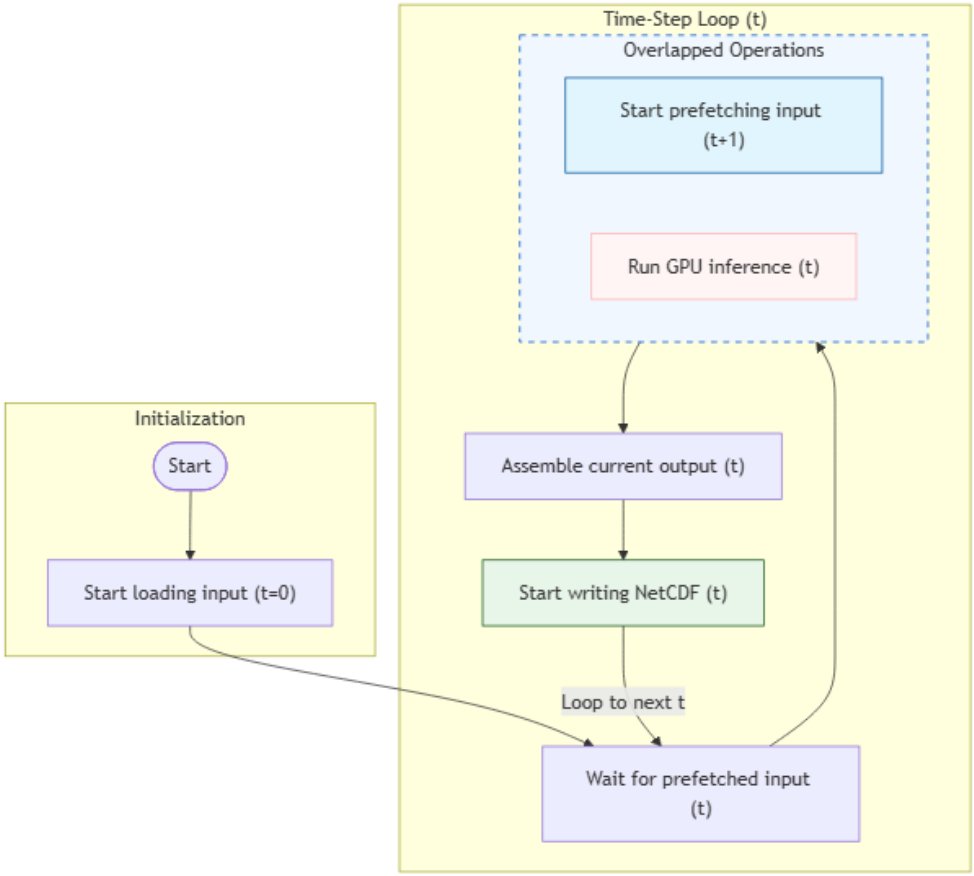

Figure 4.9: Asynchronous/pipelined workflow after input prefetching

The workflow can be described in Figure 4.9. The background worker prepares the next input sample and stores it so that the main inference loop can use it when needed. The main thread keeps control of the GPU execution. There is still a possible waiting point. If input loading takes longer than GPU inference, the main thread still has to wait for the prefetched sample. So input prefetching is not a complete solution by itself. But it reduces unnecessary idle time when the loading time can be partly hidden behind model execution.

In this thesis, input prefetching was used as one part of the I/O optimization. It works together with batched tile inference and asynchronous writing. Batched tile inference improves the GPU workload once the input is ready. Input prefetching helps make the next input ready earlier.

4.7.6 Asynchronous NetCDF Writing

After each retrieval timestep, CHIMP writes the output precipitation field to a NetCDF file. In the original inference pipeline, this writing step was part of the main loop. The program first loaded the input window, then ran GPU inference, then assembled the output on the CPU, and then wrote the NetCDF file before moving to the next timestep. This made the workflow easy to follow, but it also meant that

file writing could block the next inference step.

The problem is that NetCDF writing is not GPU work. During this step, the GPU does not help much. If the main thread waits for the file to be written before starting the next timestep, the GPU can become idle again. This is especially inefficient after batched tile inference, because the GPU computation becomes faster, so the relative cost of waiting for I/O becomes more visible.

To reduce this blocking time, an asynchronous writer was introduced. The idea is to move NetCDF writing out of the main inference loop. Instead of writing the output file directly, the main thread creates the output dataset and places it into a write queue. A background writer thread then takes items from this queue and writes them to disk. This allows the main thread to continue preparing and processing the next timestep.

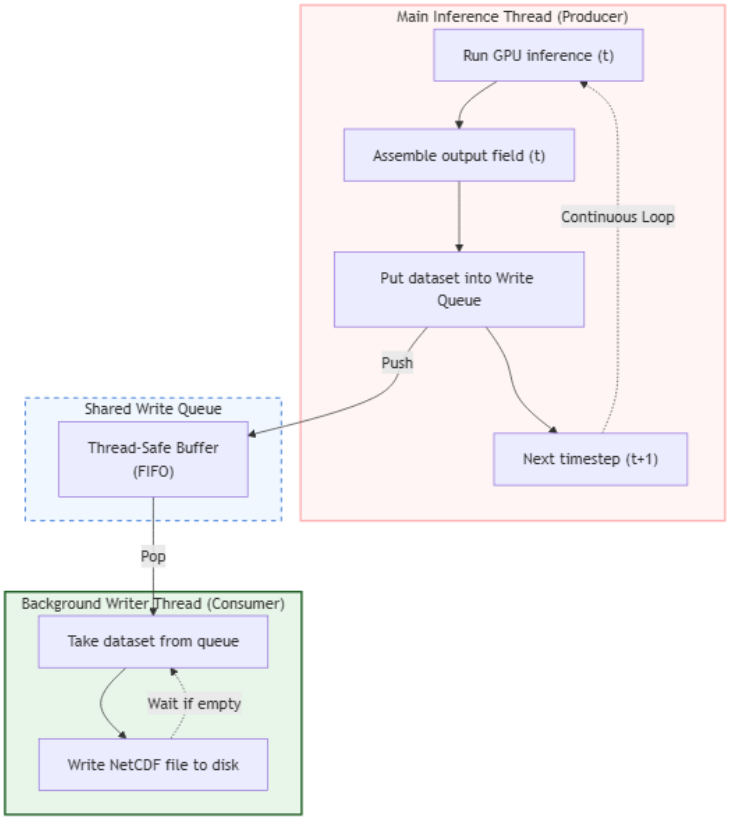

Figure 4.10: Asynchronous writing workflow

The workflow of asynchronous writing is shown in Figure 4.10, where the writing step is overlapped with later work. While the writer thread is saving the output from timestep t, the main thread can start loading or processing timestep t+1. This reduces the amount of time where the whole pipeline is waiting for disk I/O.

The asynchronous writer was implemented in AsyncWriter.py. A queue-based design was used because it is simple and controlled. The main loop only needs to hand over completed output datasets, while one writer thread handles the actual file serialization.

This avoids creating many uncontrolled write tasks. It also keeps output writing separated from the GPU inference code, which makes the implementation easier to debug.

This optimization is closely connected to input prefetching. Input prefetching overlaps loading the next input window with current inference. Asynchronous writing overlaps saving the previous output with current or next inference. Together, they change the pipeline from a mostly serial structure into a partly overlapped structure. This structure is shown in Figure 4.11.

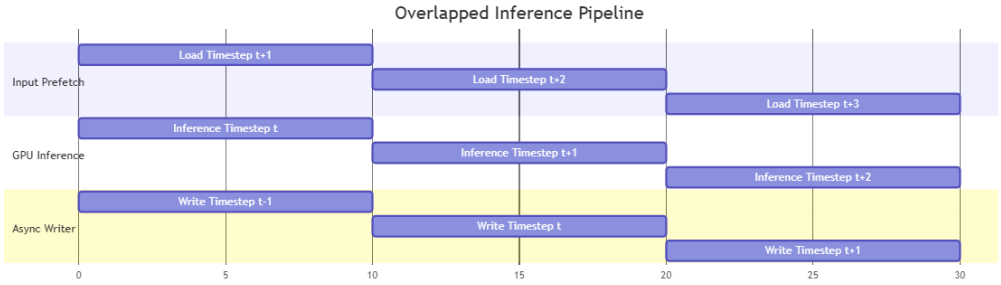

Figure 4.11: I/O pipeline optimization with input prefetching and asynchronous writing

The original inference pipeline waits for input loading and output writing at every timestep. In the optimized version, the next input window is prefetched in the background, while the previous output is written by an asynchronous NetCDF writer. This reduces I/O stalls in the main inference loop.

There are still limits to this method. If the storage system is too slow, or if output files are produced faster than they can be written, the queue can grow and the main thread may eventually have to wait. For this reason, asynchronous writing does not remove the need for efficient output formats or a stable file system. But in this thesis, it was a practical optimization because the inference loop no longer had to stop for every NetCDF write.

4.7.7 Time-Domain Multi-GPU Inference

The final inference optimization was to run the retrieval over multiple GPUs. A direct way to use multiple GPUs would be to split the spatial tiles of one timestep across GPUs. However, this was not the main approach used in this thesis. Spatial splitting would make the implementation more complex because tile outputs from different GPUs would have to be gathered and assembled for the same timestep. It could also add communication overhead between GPUs, especially if the pipeline needs to synchronize after every timestep.

In this thesis, the inference period was split along the time dimension. Each GPU was assigned a separate part of the time series and ran the full CHIMP inference pipeline on that part. For example, for one day of retrievals, the first GPU can process the first half of the day, while the second GPU processes the second half. This strategy works because the retrieval at each timestep is mostly independent during inference. The model uses a recent input window to produce the precipitation

estimate for the latest time step, but it does not need the model output from the previous timestep as input. Because of this, different time segments can be processed in parallel without changing the model architecture.

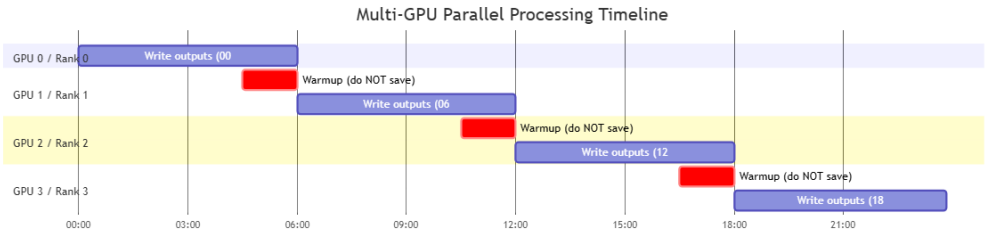

Figure 4.12: Workflow of multi-GPU parallel processing timeline (example of 4 GPUs)

The optimized workflow is shown in Figure 4.12. Each GPU holds a full copy of the trained model and processes its own time segment. Since each retrieval uses a recent input window, the start of each time segment must still have enough previous input frames available. For the 24-step configuration, the model needs the recent four-hour FCI sequence. So when a segment starts, the input loader must still be able to access the earlier frames needed to build the first few windows.

This approach was chosen because it is practical for the CHIMP retrieval workflow. The final evaluation periods contain many timesteps, so there is enough work to divide between GPUs. It also avoids the communication overhead that would come from distributing the tiles of a single timestep across multiple GPUs.

The method also has limits. If the period is very short, splitting it across several GPUs may not help much because each process has startup overhead and has to load the model. If the file system becomes the main bottleneck, several GPUs reading and writing at the same time can also increase I/O pressure. For this reason, time-domain multi-GPU inference was tested together with the earlier I/O optimizations, rather than as a replacement for them.

5

Results

5.1 FCI-Only and FCI+PMW Retrieval Results

This section compares the FCI-only baseline with the FCI+PMW model. The FCI-only model uses only geostationary MTG-FCI observations and therefore shows what can be retrieved from dense cloud-top information alone. The FCI+PMW model uses the same FCI sequence but adds sparse passive microwave observations from ATMS and AWS when they are available.

Training was divided into warm-up, main training, and fine-tuning stages. Figure 5.1 shows the FCI-only training curves as an example. The correlation coefficient increases during training, while the MSE and mean loss decrease and become more stable. The FCI+PMW training curves showed a similar overall behaviour, with a slightly lower final MSE.

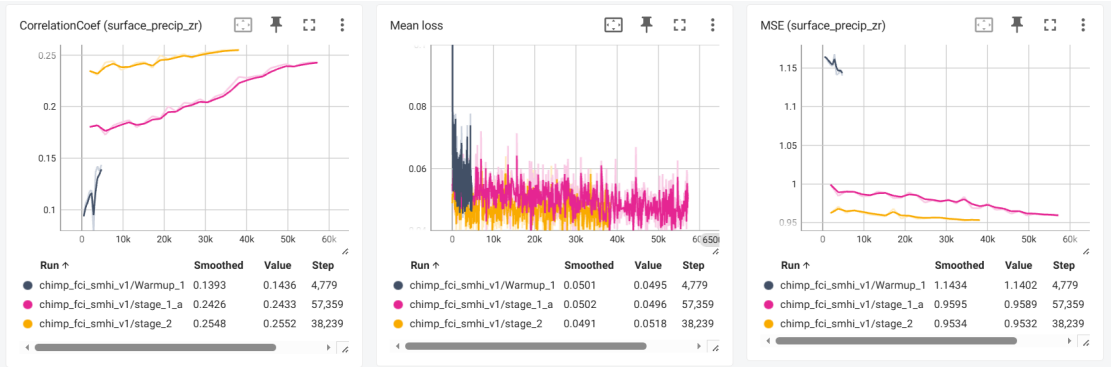

Figure 5.1: Training curves of the FCI-only baseline model. The panels show the validation correlation coefficient, mean loss, and MSE for surface precipitation retrieval.

The consolidated evaluation results are shown in Table 5.1. The table groups the two models within each evaluation period, so that the effect of adding PMW can be compared directly for each period and for the combined test set.

Table 5.1: Evaluation metrics for the FCI-only and FCI+PMW models

Period	Model	MAE	R^2	Corr	MCC 0.1	MCC 1.0	Volume Error
January 2025	FCI-only V1	3.8837	-1.0274	0.5273	0.1050	0.3365	+90.65%
	FCI+PMW V1	2.6402	0.0267	0.5243	0.2083	0.3477	+10.79%
July 2025	FCI-only V1	1.5432	0.2841	0.5521	0.2519	0.4882	-33.12%
	FCI+PMW V1	1.5330	0.2657	0.5326	0.3130	0.4427	-38.40%
September 2025	FCI-only V1	2.4362	0.0821	0.3775	0.2329	0.4798	-35.26%
	FCI+PMW V1	2.1294	0.2003	0.5447	0.4894	0.5174	-53.61%
All periods	**FCI-only V1**	**2.6997**	**-0.1923**	**0.4140**	**0.2155**	**0.4428**	**+13.66%**
	FCI+PMW V1	**2.1426**	**0.1722**	**0.4853**	**0.3493**	**0.4433**	**-25.67%**
	Change	**-0.5571**	**+0.3645**	**+0.0713**	**+0.1338**	**+0.0005**	**-39.33 pp**

The main difference is that the combined R^2 becomes positive when PMW is added. It increases from -0.1923 for FCI-only to 0.1722 for FCI+PMW. The improvement is especially clear in January, where the FCI-only model has a strongly negative R^2 and a volume error of +90.65%. With PMW, the January R^2 increases to 0.0267 and the volume error decreases to +10.79%. This suggests that microwave information helps reduce the large winter overestimation produced from cloud-top information alone.

The qualitative comparison in Figure 5.2 shows the same pattern in the retrieved fields. The FCI-only model captures some broad precipitation systems, but its fields are smoother than the BALTRAD reference and local intense cells are often weakened or displaced. In several cases, the FCI+PMW retrieval is broader and stronger than the FCI-only retrieval. This is in line with the lower MAE and higher MCC at the 0.1 mm/day threshold. However, both retrievals remain smoother than the radar reference, showing that local intensity calibration is still difficult.

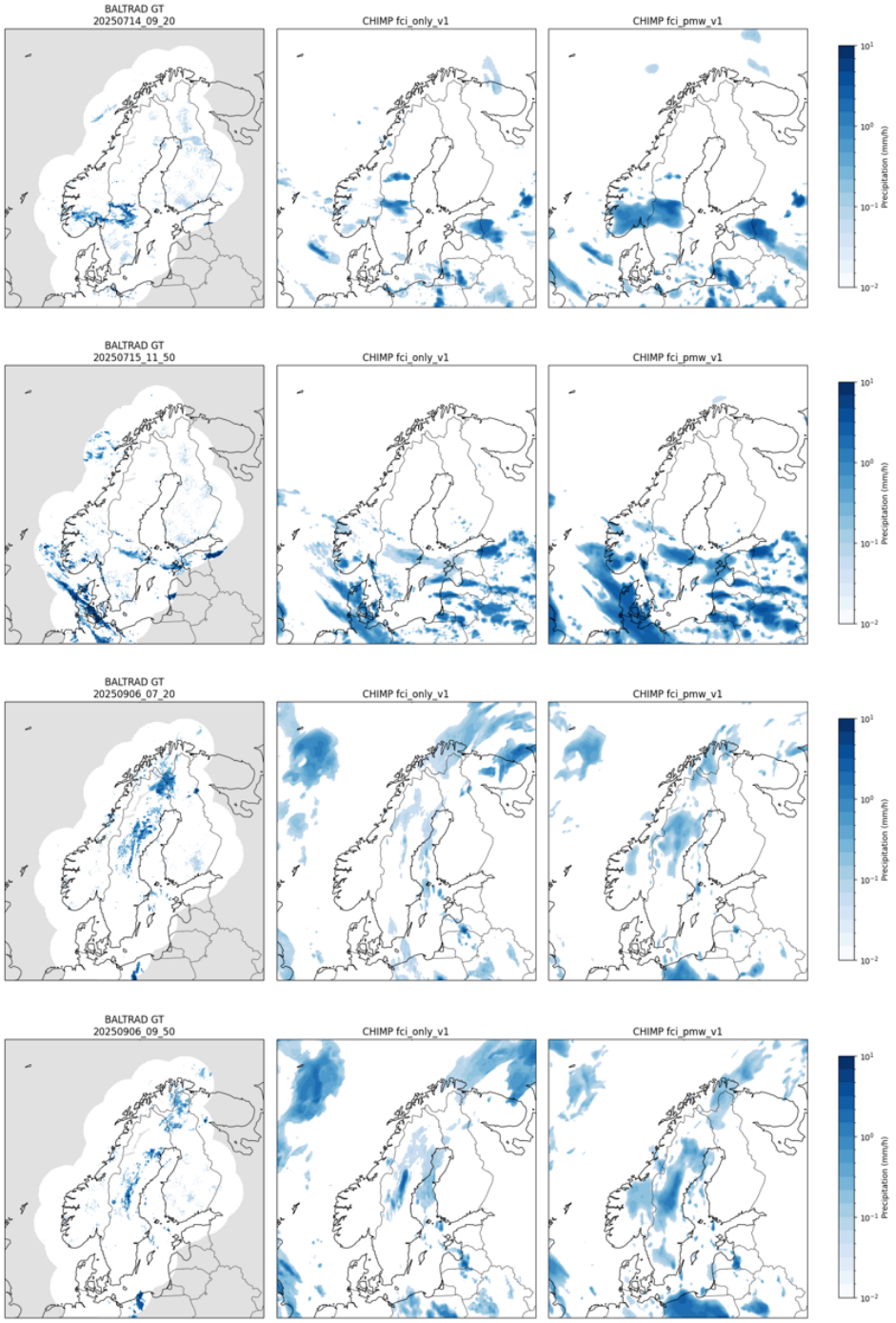

Figure 5.2: Qualitative comparison between BALTRAD reference precipitation, FCI-only retrieval, and FCI+PMW retrieval. The first column shows BALTRAD, the second column shows the FCI-only model, and the third column shows the FCI+PMW model.

5.2 Comparison of Retrieval Behaviour

The conditional distribution analysis in Figure 5.3 gives a more direct view of how the two retrievals behave. After bias removal, the FCI-only model remains strongly compressed toward moderate precipitation values: the conditional median increases only slowly as the reference precipitation increases. The FCI+PMW model follows the one-to-one line more closely over a wider range of reference precipitation. This means that PMW does not only change the total metrics; it also helps the model respond more strongly when the reference precipitation intensity increases.

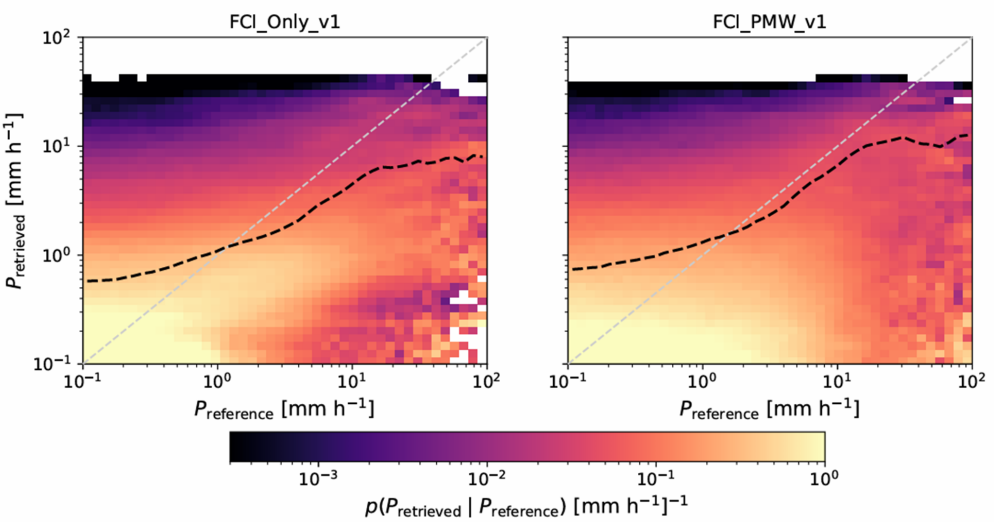

Figure 5.3: Conditional distribution of retrieved precipitation given the BALTRAD reference after bias removal. The dashed grey line is the one-to-one reference and the dashed black line shows the conditional median.

The quantitative comparison is summarized visually in Figure 5.4. The period-wise metric comparison is provided in Appendix A. The MAE decreases from 2.6997 mm/day for FCI-only to 2.1426 mm/day for FCI+PMW. The correlation coefficient also improves, from 0.4140 to 0.4853, and MCC at the 0.1 mm/day threshold increases from 0.2155 to 0.3493. These changes indicate better spatial agreement and better precipitation occurrence detection, especially for weak or widespread precipitation that is difficult to infer from cloud-top information alone.

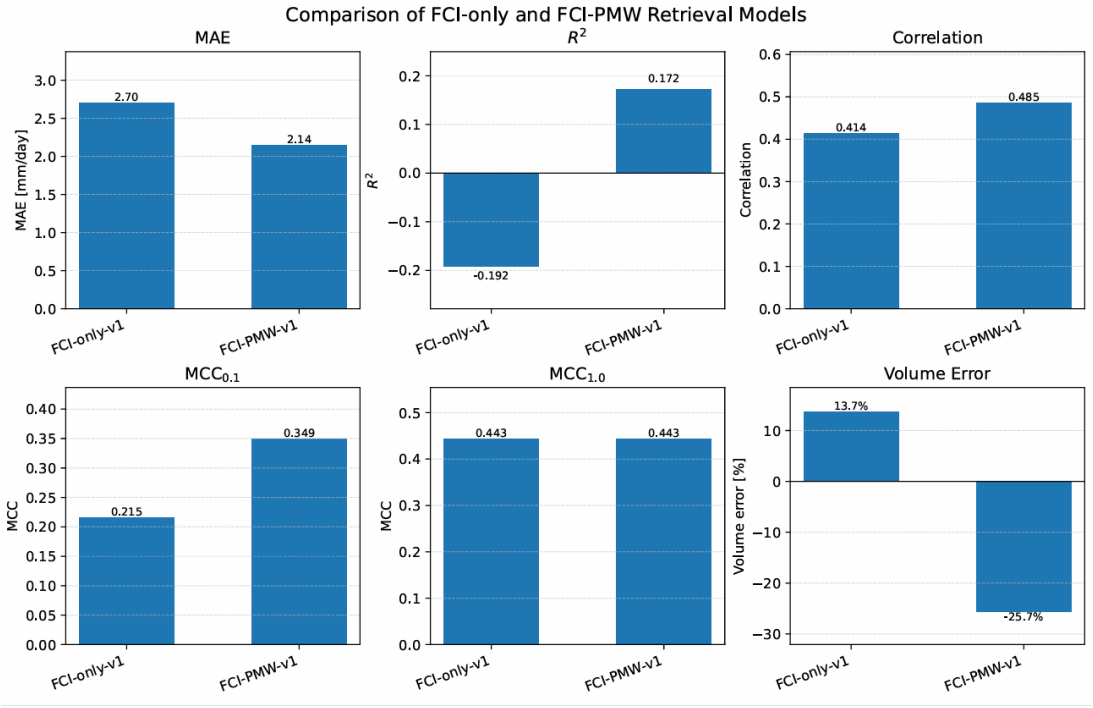

Figure 5.4: Final metric comparison between the FCI-only and FCI+PMW models. The panels show MAE, R^2, correlation, MCC at 0.1 mm/day, MCC at 1.0 mm/day, and accumulated volume error.

The accumulated volume error shows a more complex behaviour. FCI-only has an overall wet bias of +13.66%, mainly because of the January period, while FCI+PMW changes to a dry bias of -25.67%. Thus PMW reduces the strong winter overestimation, but the model becomes more conservative in July and September. PMW therefore improves several skill metrics, but calibration of the total precipitation amount remains a limitation.

This result is physically reasonable. FCI mainly provides information about cloud-top temperature and cloud-top structure, while PMW observations are more directly affected by hydrometeors inside precipitation systems. The PMW channels can therefore act as an additional physical constraint, helping the model decide whether cold or thick cloud systems are actually associated with precipitation. However, PMW observations are also sparse in time and have a coarser spatial resolution than FCI. As a result, they can improve precipitation occurrence and large-scale structure, but they do not automatically guarantee accurate local intensity or total volume estimation.

5.3 Ablation Study: Impact of PMW Data Availability

The previous section compared the FCI-only model and the FCI+PMW model over the full evaluation period. However, PMW observations are not available everywhere

and at all times. They are only available when a polar-orbiting satellite passes over the domain. Therefore, the improvement from PMW may depend strongly on whether microwave observations are actually present in the input window.

To test this, an ablation study was carried out by splitting the evaluation into two time windows. The first window is from 10:00 to 14:00, where PMW observations are relatively dense over the domain. This window is referred to as W1, or the PMW-active window. The second window is from 14:00 to 18:00, where PMW observations are mostly missing. This window is referred to as W2, or the PMW-missing window. The FCI-only model and the FCI+PMW model were evaluated separately in these two windows. This setup makes it possible to check whether the fusion model performs better only when PMW data are available, and whether its performance changes when PMW input becomes sparse or absent.

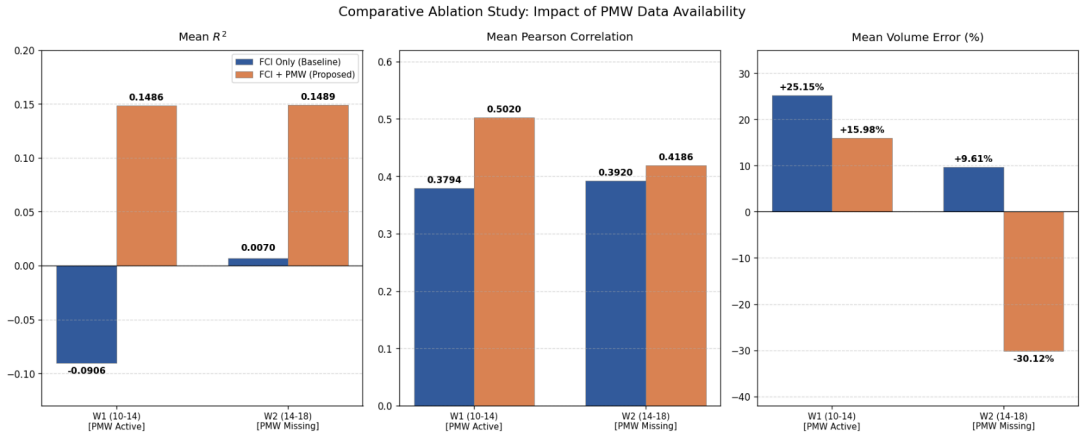

Figure 5.5: Ablation study comparing FCI-only and FCI+PMW retrievals during a PMW-active window and a PMW-missing window. The panels show mean R^2, mean Pearson correlation, and mean volume error.

As shown in Figure 5.5, the FCI+PMW model delivers a clear R^2 improvement within the PMW-active window. The mean R^2 increases from about -0.09 for the FCI-only model to about 0.14 for the FCI+PMW model. The correlation improves from 0.38 to 0.50. The volume error also improves, from around 25.15% to around 15.98%. This suggests that when PMW data are available, they provide useful additional information for spatial consistency and the intensity calibration of the retrieval.

The result changes in the PMW-missing window. In W2, the FCI+PMW model still improves several continuous retrieval metrics compared with the FCI-only model. The R^2 increases from about 0.01 to about 0.15, and the mean Pearson correlation increases slightly from 0.39 to 0.42. Both improvements are smaller than in W1. Also, the volume error changes from a small wet bias of about +9.61% for FCI-only to a clear dry bias of about -30.12% for FCI+PMW. This means that when PMW input is missing, the fusion model loses part of the advantage seen in W1 and becomes more conservative again.

This result suggests that the FCI+PMW model makes effective use of PMW observations when they are available. During the PMW-active period, the microwave

channels provide an additional physical constraint on the precipitation field. This helps the model compensate for the limitations of FCI cloud-top information alone and improves both R^2 and spatial correlation. The reduced wet bias in W1 also indicates that PMW helps constrain the precipitation amount.

In this experiment, the model performs best under high-PMW-availability conditions, but this benefit is not fully stable. Overall, this ablation study shows that PMW can improve the retrieval when microwave observations are available, but when the PMW constraint is absent, the model still retains some skill, but its precipitation intensity calibration becomes less reliable.

5.4 Effect of PMW Zenith-Angle Channels

An additional experiment was carried out to test the effect of PMW zenith-angle channels. In FCI+PMW V1, the zenith angle was removed and only the brightness-temperature channels were kept. In this experiment, the zenith-angle channels were kept as input channels. The purpose of this test was to check whether viewing geometry could help the model interpret the PMW observations more accurately.

The result is shown in Table 5.2. Compared with FCI+PMW V1, the zenith-angle version changes the model behaviour quite strongly. This version reaches a higher R^2 of 0.2015 and a higher correlation coefficient of 0.5573. It also has the highest absolute volume bias, with a Volume Error of +28.54%.

Table 5.2: Comparison of FCI-only, FCI+PMW V1, and FCI+PMW with zenith angle

Model	MAE	R^2	Corr	MCC 0.1	MCC 1.0	Volume Error
FCI-only V1	2.6997	-0.1923	0.4140	0.2155	0.4428	+13.66%
FCI+PMW V1	2.1426	0.1722	0.4853	0.3493	0.4433	-25.67%
FCI+PMW with zenith angle	2.4760	0.2015	0.5573	0.0843	0.3543	+28.54%

However, the other metrics show that this improvement is not stable. The MAE increases to 2.4760 mm/day, which is worse than FCI+PMW V1. The MCC also drops strongly, especially at the 0.1 mm/day threshold. MCC_0.1 decreases to 0.0843, and MCC_1.0 decreases to 0.3543. This means that although the total precipitation volume shifts upward, the pixel-wise error and rain/no-rain detection become worse.

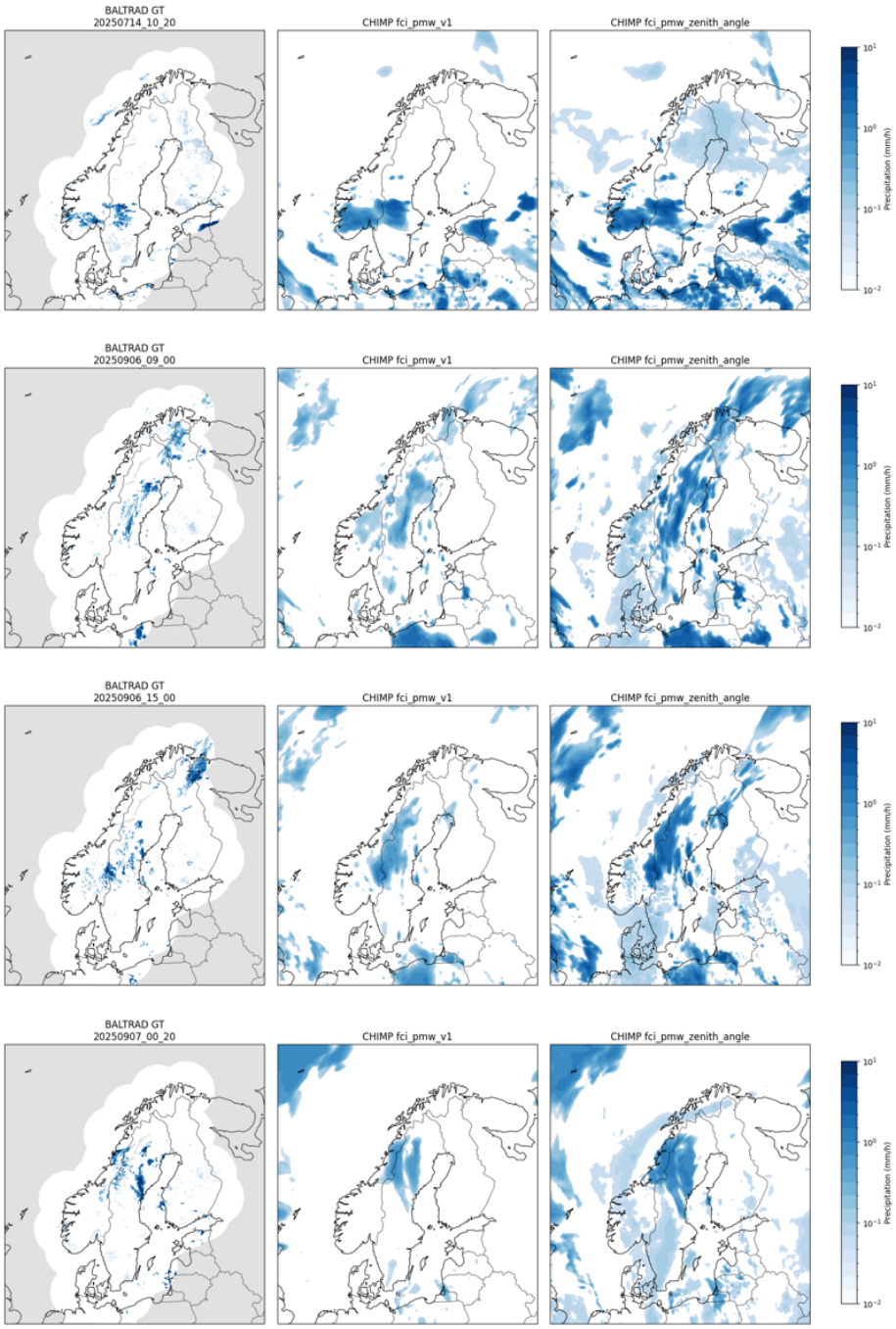

Figure 5.6: Example retrievals from FCI+PMW V1 and FCI+PMW with zenith angle compared with BALTRAD reference precipitation.

The example frames in Figure 5.6 show the same pattern. FCI+PMW with zenith angle produces stronger and more widespread precipitation areas than version 1. In some places, this helps reduce the underestimation of precipitation amount. However, in other places it also introduces extra precipitation structures that are not clearly present in the BALTRAD reference. This leads to higher local errors and weaker MCC scores. Therefore, the zenith-angle channels do not provide a simple improvement in this experiment.

Because of this behaviour, FCI+PMW with zenith angle was not selected as the main fusion model in this thesis. FCI+PMW V1 gives a more balanced result, with lower MAE and better MCC scores. The zenith-angle experiment is mainly useful as a sensitivity test, showing that viewing-geometry information can strongly affect the model calibration and precipitation distribution, even when it does not improve the overall retrieval skill.

5.5 Inference Performance Results

5.5.1 Profiling of the Original Inference Pipeline

Besides precipitation retrieval accuracy, inference performance is also important in this work. The inference pipeline must not only produce reasonable precipitation fields, but also run fast enough for practical use. Before applying optimizations, the original inference pipeline was profiled with Nsight Systems. The goal was to identify whether the bottleneck came from the neural network computation, GPU memory usage, CPU processing, or file I/O.

Figure 5.7 shows a typical Nsight Systems timeline of the original pipeline. At the beginning of the run, most of the time is spent on CPU-side work and file-related operations. The GPU is mostly idle during this period. Only after the input data have been prepared does the GPU start to execute CUDA kernels and memory transfers. According to the profiling result, the GPU work appears in short bursts rather than as a continuous workload.

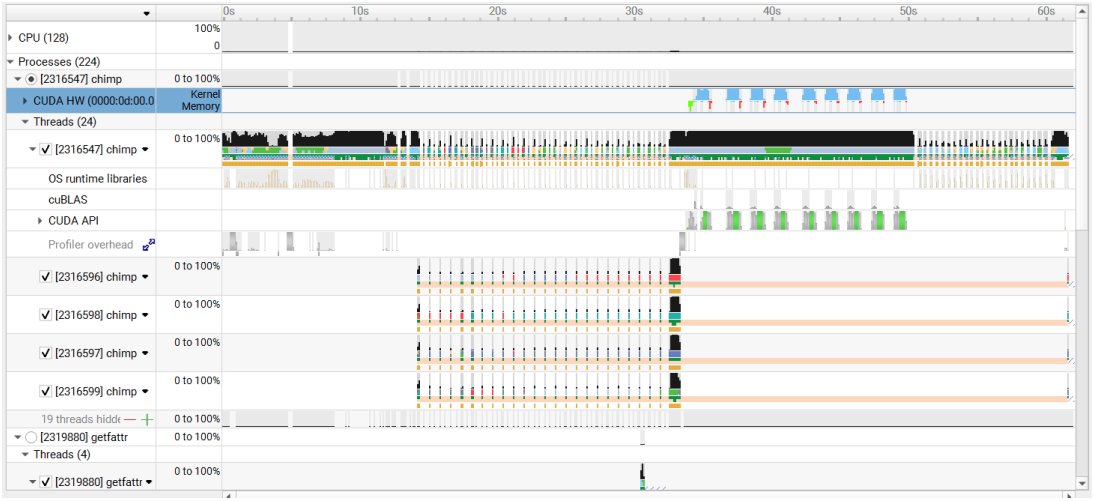

Figure 5.7: Nsight Systems timeline of the original CHIMP inference pipeline. The GPU work appears in short bursts, while the CPU and file-related operations dominate large parts of the timeline.

This profiling result suggests that the main bottleneck is not the 3D CNN computation itself. Instead, the model is often waiting for the input pipeline. The original pipeline loads input data, prepares tiles, runs inference, assembles the output, and writes NetCDF files in a mostly sequential way. This means the GPU can wait during input loading and the main thread can also wait during output writing. As a result,

the GPU does not receive enough work to stay busy. This is also reflected in the benchmark result that the original baseline only reaches an average GPU utilization of 10.5%.

5.5.2 Effect of Tile Size

The first experiment was to change the tile size used during inference. Larger tiles may seem useful because they provide more spatial context and reduce the number of tiles. However, larger tiles also increase memory use and the amount of computation per tile. Therefore, tile size was tested separately before changing the pipeline structure.

Table 5.3: Inference performance with different tile sizes in the original pipeline

Tile Size	Total Time (s)	Throughput (steps/s)	Avg GPU Util	Max GPU Util	Avg GPU Mem
64	3041.06	0.047	3.40%	8.00%	2.30%
128	2444.42	0.059	10.50%	28.00%	9.50%
256	2899.65	0.05	27.20%	37.20%	25.90%
512	5110.9	0.028	30.10%	54.40%	19.30%

The results for different tile sizes are shown in Table 5.3. They show that increasing tile size does not directly improve end-to-end runtime. The best runtime among these settings is obtained with tile size 128. When the tile size is increased to 256, the average GPU utilization increases from 10.5% to 27.2%, but the total runtime also increases from 2444.42 s to 2899.65 s. With tile size 512, the GPU utilization becomes even higher, but the total runtime increases to 5110.90 s and the throughput drops to 0.028 steps/s.

This means that higher GPU utilization alone does not necessarily mean better performance. Larger tiles make the GPU busier, but each step also becomes more expensive. In this case, the extra computation and memory pressure are larger than the benefit of reducing the number of tiles. Tile size 64 is also inefficient, because the GPU workload becomes too small and creates more communication overhead. Therefore, tile size 128 is the best setting for this thesis project, and it is used as the reference setting for the following optimization experiment.

This result shows that the performance problem is not caused by one single problem. The bottleneck is more structural. The pipeline needs to send work to the GPU more efficiently and overlap I/O with computation.

5.5.3 Optimized Inference Pipeline

Based on the profiling and tile-size experiments, three main optimizations were tested: batched tile inference, asynchronous I/O, and multi-GPU execution, as described in Chapter 4. The results are shown in Table 5.4.

Table 5.4: Runtime performance of different inference pipeline versions

Version	GPUs	Time (s)	Steps/s	Avg GPU Util	Max GPU Util	Avg GPU Mem	Speedup vs baseline	Time reduction
Original baseline	1	2444.42	0.059	10.50%	28.00%	9.50%	1.00×	0.00%
Batched tiles	1	2021.01	0.071	66.10%	79.80%	60.50%	1.21×	17.30%
Batched tiles + async I/O	1	1879.02	0.077	68.00%	78.50%	60.30%	1.30×	23.10%
Batched tiles + multi-GPU	2	1691.49	0.085	58.20%	74.50%	54.90%	1.45×	30.80%
Batched tiles + multi-GPU + async I/O	2	1545.06	0.093	67.00%	87.50%	60.50%	1.58×	36.80%

The first optimization was batched tile inference. In the original implementation, tiles were processed one by one. This created many small GPU tasks and frequent CPU-GPU synchronization. In the batched version, several tiles are grouped together and passed through the model in one forward pass. This change reduced the runtime from 2444.42 s to 2021.01 s. The throughput increased from 0.059 to 0.071 steps/s, and the average GPU utilization increased from 10.5% to 66.1%. This is the largest single improvement in GPU usage. It shows that the original pipeline underused the GPU mainly because the work units were too small.

The second optimization was asynchronous I/O. In the baseline pipeline, input loading, model inference, output assembly, and NetCDF writing were mostly handled in a sequential way. The next inference step could not start until the previous output had been written. In the optimized version, input loading and output writing are partly overlapped with GPU computation. The main thread can continue inference while a background thread prepares the next input or writes the previous result. With batched tiles and asynchronous I/O, the runtime decreased further to 1879.02 s, corresponding to a 1.30× speedup over the original baseline.

Multi-GPU performance was also tested by distributing the inference workload over two GPUs. With batched tiles and two GPUs, the runtime decreased to 1691.49 s, corresponding to a 1.45× speedup. When asynchronous I/O was added on top of the multi-GPU version, the runtime was further reduced to 1545.06 s. This gives a 1.58× speedup and a 36.8% time reduction compared with the original baseline.

The results show that the optimizations are partly cumulative. Batched tiles mainly improve GPU usage by making each forward pass larger and more efficient. Asynchronous I/O reduces waiting time around the model execution. Multi-GPU execution adds more compute capacity. However, the speedup is not perfectly linear with two GPUs, because the pipeline still contains CPU-side work, data loading, output assembly, and synchronization overhead. This means that after the model computation becomes faster, the remaining non-GPU parts of the pipeline start to limit the total speedup.

5.5.4 Temporal Sharding

After testing ordinary multi-GPU inference, an additional experiment was carried out using temporal sharding. Different from the multi-GPU setup above, this method

splits the full inference period into separate time segments. Each GPU processes one time segment independently. For example, one GPU can process the earlier part of the day, while another GPU processes the later part. It works well in this case because each output timestep can be produced independently once its input sequence is available.

Table 5.5: Additional benchmark using temporal sharding

Version	GPUs	Time (s)	Steps/s	Avg GPU Util	Max GPU Util	Avg GPU Mem	Speedup vs baseline	Time reduction
Original baseline	1	2444.42	0.059	10.5%	28.0%	9.5%	1.00×	0.0%
Batched tiles + multi-GPU + async I/O	2	1545.06	0.093	67.0%	87.5%	60.5%	1.58×	36.8%
Temporal sharding + batched tiles + async I/O	2	1076.86	0.134	53.4%	75.1%	56.5%	2.27×	55.9%

The result is shown in Table 5.5. The temporal sharding version gives the best end-to-end runtime. It reduces the total time from 2444.42 s to 1076.86 s. The throughput increases from 0.059 to 0.134 steps/s. This gives a 2.27× speedup and a 55.9% reduction in runtime.

The average GPU utilization of this version is lower than the batched multi-GPU version. However, the total runtime is much better. This is likely because temporal sharding reduces communication and scheduling overhead. It also shows that GPU utilization should not be interpreted alone. A high utilization number can mean that the GPU is busy, but it does not always mean that the whole pipeline is faster. In this case, temporal sharding reduces waiting time and scheduling overhead, so the total runtime improves even though the average utilization is lower.

5.5.5 Summary

Overall, the profiling and benchmarks show that the original inference pipeline was mainly limited by data loading, tile handling, and I/O overhead. The neural network computation itself was not the main bottleneck. Simply increasing tile size did not solve the problem. Larger tiles increased GPU utilization, but they also increased total runtime.

The most useful improvements came from changing the pipeline structure. Batched tile inference made each GPU forward pass more efficient. Asynchronous I/O reduced blocking around input preparation and output writing. Ordinary multi-GPU execution further reduced runtime by distributing the batched tile workload over two GPUs. With these changes, the runtime decreased from 2444.42 s to 1545.06 s, corresponding to a 1.58× speedup and a 36.8% time reduction.

The best benchmark result was obtained with temporal sharding. The full inference period was split into separate time segments and processed independently. This reduced the runtime to 1076.86 s, giving a 2.27× speedup and a 55.9% time reduction. However, this method assumes that the full time sequence is already available before

inference starts. Therefore, it is more suitable for offline evaluation, reprocessing, or backfilling historical data. For a real-time or near-real-time setting, where new satellite data arrive step by step, the more practical optimized pipeline is the batched tiles + multi-GPU + asynchronous I/O version.

This result is important because it shows that inference performance can be improved at the system level without changing the trained precipitation model. The pipeline feeds data to the GPU more efficiently and reduces unnecessary waiting time. For an operational or near-real-time setting, this type of optimization is very important in practical use.

6

Discussion

This chapter gives the answers to the three RQs, explains the results, identifies the limitations of the study, and outlines directions for future work.

6.1 Answers to Research Questions

6.1.1 Answer to RQ1: FCI–PMW Fusion

The first research question asked how FCI and PMW observations can be fused for precipitation retrieval. In this thesis, the fusion was done at the model-input level. FCI, ATMS, and AWS were treated as separate input sources, and they were passed into the same 3D encoder-decoder retrieval model after being mapped onto the same grid.

For the FCI-only model, the adaptation was relatively direct. The input configuration was changed to use the 16 FCI channels and kept 3D encoder-decoder as the model structure. The result shows that FCI can provide useful information for precipitation retrieval. It learns the broad structure and movement of precipitation systems, especially when the cloud-top signal is clear.

The FCI+PMW model required more changes. PMW observations from ATMS and AWS are not available at every time step and do not have the same dense spatial coverage as FCI. This is handled by representing missing PMW data through masked tensors. When no PMW file was available for a given time step, or when part of the MEPS domain was outside the PMW swath, the missing values were filled with a safe placeholder after standardization, while a mask recorded which values were originally missing. This allowed the model to receive tensors with consistent shapes, without treating missing PMW pixels as real microwave measurements.

So the answer to RQ1 is that FCI and PMW can be fused in the CHIMP retrieval model by using FCI as the continuous spatio-temporal input and PMW as a sparse masked auxiliary input. The masking mechanism is essential because it makes irregular PMW coverage compatible with a fixed-shape neural network input.

6.1.2 Answer to RQ2: Effect of Adding PMW

The second research question asked whether adding PMW observations improves precipitation retrieval compared with the FCI-only baseline. The results show that

PMW can improve the retrieval overall.

The FCI-only model provides a useful baseline. It can capture the broad movement and shape of precipitation systems from the FCI image sequence. This shows that the temporal information in FCI is valuable. However, the FCI-only model also has limitations in intensity estimation. The overall R^2 is negative, and the model shows different bias behaviour in different evaluation periods. It strongly overestimates precipitation in the January evaluation period, which is likely related to winter snowfall and mixed-phase conditions, while it underestimates precipitation volume in the July and September evaluation periods. This suggests that FCI cloud-top information alone is not always sufficient for stable precipitation intensity estimation.

Adding PMW observations reduces this limitation. The FCI+PMW V1 model gives a more balanced result than the FCI-only baseline in several metrics. It lowers MAE, changes the overall R^2 from negative to positive, improves correlation, and gives a clear improvement in MCC at the 0.1 mm/day threshold. This suggests that PMW provides information that is not fully contained in the FCI sequence. Although PMW observations are sparse, they are more directly related to hydrometeors inside precipitation systems. When a PMW overpass is present, it can therefore act as an additional physical constraint on the retrieval.

This result is also visible in the ablation study. During periods with PMW observations, the fusion model performs better than the FCI-only model, especially in R^2, correlation, and volume error. When PMW observations are missing, the FCI+PMW model still keeps some advantage, but the improvement becomes weaker and the volume estimate becomes drier.

The answer to RQ2 is therefore that PMW observations do improve precipitation retrieval compared with FCI-only, especially when PMW is available. However, the improvement is conditional. The current fusion model benefits from PMW, while also showing some sensitivity to missing PMW input. Future work should make the model more robust to sparse microwave coverage and improve the calibration of precipitation amount.

6.1.3 Answer to RQ3: Inference Optimization

The third research question asked what the main computational bottlenecks of the current inference pipeline are and what methods can improve inference performance. The results show that the main bottleneck was not the neural network computation alone. Instead, a large part of the runtime came from the execution pipeline around the model, especially input loading, file checking, CPU-side assembly, and output writing.

Nsight Systems showed long periods where the CPU was busy with many small I/O-related operations, while the GPU was mostly idle. This means that simply using a faster GPU would not solve the main problem. The model computation itself was relatively fast once the input tensors were ready. The larger issue was that the GPU often had to wait for the surrounding pipeline.

We used several optimizations such as batched tile inference, asynchronous I/O, and temporal sharding. Batched tile inference improved the GPU workload by processing multiple spatial tiles in one forward pass instead of sending one small tile at a time. Asynchronous I/O reduced waiting time by overlapping input loading, GPU inference, and output writing. The performance results show that these changes improved the inference pipeline step by step. In the benchmark, the runtime decreased from 2444.42 s for the original baseline to 1545.06 s for the optimized two-GPU setup, corresponding to a speedup of about 1.58×.

The answer to RQ3 is therefore that the main bottleneck of the inference pipeline was the data and execution pipeline around the model, not only the 3D CNN computation itself. Batched tile inference, asynchronous input/output, and multi-GPU execution can reduce runtime without changing the trained model or affecting retrieval accuracy.

6.2 Discussion of the PMW Zenith-Angle Experiment

We added the zenith-angle channels in the FCI and PMW training to test whether viewing-geometry information could help the model interpret PMW observations more correctly.

The result was not a simple improvement over PMW V1. This model produced a much wetter output distribution. At the same time, the MAE increased and the MCC scores became worse. This means that the model predicted more precipitation overall, but the additional precipitation was not always correct in location or intensity. In other words, the model shifted the dry volume bias of PMW V1 into a wet bias, but did not improve the local retrieval quality.

One possible explanation is that the zenith-angle channels changed the calibration learned by the model. It is a structured signal related to viewing geometry, scan position, and observation quality. It does not measure precipitation directly. When it is added as an ordinary input channel, the network may treat it in the same way as a brightness-temperature channel and learn correlations between scan geometry, PMW coverage patterns, and the BALTRAD target. Some of these correlations may be dataset-specific rather than physically related to precipitation. This can change the learned calibration between PMW brightness temperatures and precipitation intensity. In this experiment, the effect appears as a broader upward shift in the retrieved precipitation distribution. This reduces underestimation in some cases, but it also overshoots the accumulated volume and increases false or misplaced precipitation.

This also explains why the metric changes are mixed. A general upward shift can move a dry-biased model toward larger precipitation amounts, but this does not guarantee a better precipitation field. If the increase is too broad, or if it happens in regions where the radar reference does not show precipitation, the pixel-wise error and threshold-based detection scores become worse.

Therefore, the zenith-angle experiment should be treated as a diagnostic result rather

than as a final model improvement. It shows that adding physically meaningful auxiliary information can still change the model behaviour in an unstable way. More input channels are not always better, especially when the new channels are strongly structured and can influence the model calibration.

Overall, the zenith-angle experiment shows that in this study, viewing-geometry information can strongly affect the model output. It removed the dry bias of PMW V1 but overshot into a wet bias, while also increasing MAE and weakening precipitation detection. For future work, zenith-angle information should be handled more carefully, for example through stronger normalization checks, separate geometry encoding, quality masking, or more controlled ablation tests.

6.3 Limitations

This thesis has several limitations. First, the reference data are based on BALTRAD radar-derived precipitation, which is useful but not a perfect ground truth. Radar precipitation estimates can contain errors from beam geometry, terrain effects, snow or mixed-phase precipitation, and the reflectivity-to-rain-rate conversion. Therefore, the reported metrics should be interpreted as agreement with the radar reference, not as absolute truth.

There are also limitations in the model and inference setup. The model configuration was constrained by GPU memory, so scene size, sequence length, and number of input channels had to be balanced rather than freely optimized. The number of training epochs was slightly shortened because the training cycle was very long. A full training run on the yearly FCI/PMW dataset could take more than one week. The FCI+PMW fusion strategy is also relatively simple, PMW is added as a sparse masked input to the same retrieval model. This works indeed, but ablation results show that the model is still sensitive to missing PMW observations. Finally, the HPC optimization results were measured on the available cluster environment and mainly target inference pipeline bottlenecks such as tile batching and I/O overlap. The exact speedup may therefore depend on the filesystem, hardware, and workload.

6.4 Future work

There are several directions that could be explored in future work. First, the robustness of the FCI+PMW fusion model could be improved. The ablation results show that PMW observations are useful when they are available, but the advantage of the fusion model becomes weaker when PMW input is missing. This is important because PMW overpasses are naturally sparse and cannot provide continuous temporal coverage.

Second, the fusion architecture could be improved. This thesis used a relatively simple early-fusion design, where FCI, ATMS, and AWS are provided as input sources to the same CHIMP model. Future models could use separate encoders for FCI and PMW, attention-based fusion, confidence weighting, or geometry-aware modules.

These methods may allow the model to use PMW when it is helpful without becoming too dependent on it.

Third, future work could include longer and more diverse training and evaluation. Because the training cycle is long, the number of model variants and training epochs in this thesis had to be limited. A larger study could train the models for more epochs, tune the scene size and sequence length more systematically, and evaluate the results over more months or multiple years. This would make it easier to judge whether the conclusions hold across different seasons, regions, and precipitation types.

The inference pipeline can be improved further for operational use. For real-time use, future work should focus more on streaming inference, robust input prefetching, asynchronous writing, and better handling of delayed or missing files. Also the low-level optimizations, such as tensor layout tuning, quantization, or custom GPU kernels can be implemented in the future.

7

Conclusion

This thesis adapted the CHIMP precipitation retrieval framework to MTG-FCI and FCI+PMW retrieval over the MEPS domain, and evaluated both retrieval accuracy and inference performance. The work started from an FCI-only baseline, then added passive microwave observations from ATMS and AWS as sparse input sources. Since PMW observations are not continuously available, missing microwave data had to be handled explicitly through the sparse input and masking mechanism. This made it possible to keep FCI as the dense spatio-temporal backbone while using PMW as an additional source of information when overpasses were available.

The FCI-only model showed that FCI sequences contain useful information for precipitation retrieval. The model was able to capture broad precipitation structures and the general movement of precipitation systems. However, it also showed clear limitations in intensity estimation. The overall R^2 was negative, and the volume error changed between the evaluation periods. In particular, the model strongly overestimated precipitation in the winter evaluation period, while it underestimated the accumulated precipitation volume in the other two periods. This shows that FCI cloud-top information alone is not always sufficient for stable precipitation intensity calibration.

Adding PMW observations improved the retrieval in several important aspects. The FCI+PMW model reduced MAE, changed the overall R^2 from negative to positive, improved correlation, and clearly improved MCC at the 0.1 mm/day threshold. The improvement was most visible when PMW observations were available. In those periods, the microwave information acted as an additional physical constraint and helped the model estimate the precipitation field more reliably. However, the experiments also showed a limitation. The fusion model remained sensitive to PMW availability, and the volume estimate became drier when PMW input was missing or sparse. Overall, this means that the current early-fusion approach is useful, but not yet fully robust to intermittent PMW coverage.

On the computational side, the thesis showed that the main inference bottleneck was not only the 3D CNN computation. Profiling with Nsight Systems showed that the original pipeline spent a large amount of time on input loading, small tile processing, CPU-side operations, and NetCDF writing. The GPU was often idle because the surrounding pipeline did not feed it efficiently. Several practical optimizations were therefore applied. Batched tile inference increased the GPU workload per forward pass, asynchronous I/O reduced blocking from input and output operations, and

multi-GPU execution further reduced runtime. The optimized two-GPU pipeline achieved a 1.58× speedup in the benchmark test. Temporal sharding gave the best offline benchmark result, with a 2.27× speedup, but it is more suitable for historical reprocessing than for real-time use.

Overall, this thesis shows that CHIMP can be adapted to use both FCI and sparse PMW observations for precipitation retrieval over the MEPS domain. The results indicate that FCI alone can provide a useful baseline, while PMW adds extra physical information and can improve the retrieval when microwave observations are available. The improvement is not uniform across all situations, but the comparison and ablation study show that PMW can improve precipitation detection and spatial consistency, although precipitation amount calibration remains a limitation. On the computational side, by batching tiles, overlapping I/O, and using multiple GPUs, the retrieval workflow became considerably faster without changing the trained model. These findings answer the main research questions and show both the potential and the remaining practical challenges of using multimodal satellite data for precipitation retrieval.

Bibliography

[1] C. Kidd and V. Levizzani, "Status of satellite precipitation retrievals," Hydrology and Earth System Sciences, vol. 15, no. 4, pp. 1109-1116, 2011, doi: 10.5194/hess-15-1109-2011.

[2] C. Prigent, "Precipitation retrieval from space: An overview," Comptes Rendus Geoscience, vol. 342, no. 4-5, pp. 380-389, 2010, doi: 10.1016/j.crte.2010.01.004.

[3] M. Kuehnlein, T. Appelhans, B. Thies, and T. Nauss, "Precipitation Estimates from MSG SEVIRI Daytime, Nighttime, and Twilight Data with Random Forests," Journal of Applied Meteorology and Climatology, vol. 53, no. 11, pp. 2457-2480, 2014, doi: 10.1175/JAMC-D-14-0082.1.

[4] R. J. Joyce, J. E. Janowiak, P. A. Arkin, and P. Xie, "CMORPH: A method that produces global precipitation estimates from passive microwave and infrared data at high spatial and temporal resolution," Journal of Hydrometeorology, vol. 5, no. 3, pp. 487-503, 2004, doi: 10.1175/1525-7541(2004)005<0487:CAMTPG>2.0.CO;2.

[5] G. J. Huffman et al., "Integrated Multi-satellitE Retrievals for the Global Precipitation Measurement (GPM) mission (IMERG)," in Satellite Precipitation Measurement, V. Levizzani et al., Eds. Cham: Springer, 2020, pp. 343-353, doi: 10.1007/978-3-030-24568-9_19.

[6] G. J. Huffman et al., "The TRMM Multisatellite Precipitation Analysis (TMPA): Quasi-Global, Multiyear, Combined-Sensor Precipitation Estimates at Fine Scales," Journal of Hydrometeorology, vol. 8, no. 1, pp. 38-55, 2007, doi: 10.1175/JHM560.1.

[7] P. Nguyen, M. Ombadi, S. Sorooshian, K. Hsu, A. AghaKouchak, D. Braithwaite, H. Ashouri, and A. R. Thorstensen, "The PERSIANN family of global satellite precipitation data: a review and evaluation of products," Hydrology and Earth System Sciences, vol. 22, pp. 5801-5816, 2018, doi: 10.5194/hess-22-5801-2018.

[8] P. Nguyen et al., "PERSIANN Dynamic Infrared-Rain Rate (PDIR-Now): A Near-Real-Time, Quasi-Global Satellite Precipitation Dataset," Journal of Hydrometeorology, vol. 21, no. 12, pp. 2893-2906, 2020, doi: 10.1175/JHM-D-20-0177.1.

[9] T. Ushio et al., "A Kalman Filter Approach to the Global Satellite Mapping of Precipitation (GSMaP) from Combined Passive Microwave and Infrared Radiometric Data," Journal of the Meteorological Society of Japan, vol. 87A, pp. 137-151, 2009, doi: 10.2151/jmsj.87A.137.

[10] A. S. Gebregiorgis, P.-E. Kirstetter, Y. E. Hong, J. J. Gourley, G. J. Huffman, W. A. Petersen, X. Xue, and M. R. Schwaller, "To What Extent Is the Day 1 GPM IMERG Satellite Precipitation Estimate Improved as Compared to TRMM TMPA-RT?" Journal of Geophysical Research: Atmospheres, vol. 123, no. 3, pp. 1694-1707, 2018, doi: 10.1002/2017JD027606.

[11] European Space Agency, "Flexible Combined Imager," ESA. Available: `https://www.esa.int/Applications/Observing_the_Earth/Meteorological_missions/meteosat_third_generation/Flexible_Combined_Imager`.

[12] EUMETSAT, "Meteosat Third Generation Instruments," EUMETSAT. Available: `https://www.eumetsat.int/meteosat-third-generation-instruments`.

[13] J. Schmetz, P. Pili, S. Tjemkes, D. Just, J. Kerkmann, S. Rota, and A. Ratier, "An Introduction to Meteosat Second Generation (MSG)," Bulletin of the American Meteorological Society, vol. 83, no. 7, pp. 977-992, 2002, doi: 10.1175/1520-0477(2002)083<0977:AIOTMSG>2.3.CO;2.

[14] D. M. Aminou, "MSG's SEVIRI Instrument," ESA Bulletin, no. 111, pp. 15-17, 2002.

[15] P. P. Martin, Y. Durand, D. M. Aminou, C. Gaudin-Delrieu, and J.-L. Lamard, "FCI instrument on-board MeteoSat Third Generation satellite: design and development status," in International Conference on Space Optics - ICSO 2020, Proc. SPIE, vol. 11852, pp. 125-140, 2021, doi: 10.1117/12.2599152.

[16] World Meteorological Organization, "Details for Instrument FCI," OSCAR/Space. Available: `https://space.oscar.wmo.int/instruments/view/fci`.

[17] T. T. Wilheit, A. T. C. Chang, and L. S. Chiu, "Algorithms for the retrieval of rainfall from passive microwave measurements," Remote Sensing Reviews, vol. 11, no. 1-4, pp. 163-194, 1994, doi: 10.1080/02757259409532264.

[18] C. Kidd, T. Matsui, and S. Ringerud, "Precipitation Retrievals from Passive Microwave Cross-Track Sensors: The Precipitation Retrieval and Profiling Scheme," Remote Sensing, vol. 13, no. 5, Art. no. 947, 2021, doi: 10.3390/rs13050947.

[19] R. R. Ferraro, F. Weng, N. C. Grody, and L. Zhao, "Precipitation Characteristics over Land from the NOAA-15 AMSU Sensor," Geophysical Research Letters, vol. 27, no. 17, pp. 2669-2672, 2000, doi: 10.1029/2000GL011665.

[20] C. D. Kummerow et al., "The Evolution of the Goddard Profiling Algorithm to a Fully Parametric Scheme," Journal of Atmospheric and Oceanic Technology, vol. 32, no. 12, pp. 2265-2280, 2015, doi: 10.1175/JTECH-D-15-0039.1.

[21] Y. You, N.-Y. Wang, and R. Ferraro, "A Prototype Precipitation Retrieval Algorithm over Land Using Passive Microwave Observations Stratified by Surface Condition and Precipitation Vertical Structure," Journal of Geophysical Research: Atmospheres, vol. 120, no. 11, pp. 5295-5315, 2015, doi: 10.1002/2014JD022534.

[22] L. Tang, Y. Tian, and X. Lin, "Validation of Precipitation Retrievals over Land from Satellite-Based Passive Microwave Sensors," Journal of Geophysical Research:

Atmospheres, vol. 119, no. 8, pp. 4546-4567, 2014, doi: 10.1002/2013JD020933.

[23] S. Ringerud, C. Peters-Lidard, S. J. Munchak, and Y. You, "Applications of Dynamic Land Surface Information for Passive Microwave Precipitation Retrieval," Journal of Atmospheric and Oceanic Technology, vol. 38, no. 2, pp. 167-180, 2021, doi: 10.1175/JTECH-D-20-0048.1.

[24] S. Pfreundschuh, P. J. Brown, C. D. Kummerow, P. Eriksson, and T. Norrestad, "GPROF-NN: a neural-network-based implementation of the Goddard Profiling Algorithm," Atmospheric Measurement Techniques, vol. 15, pp. 5033-5060, 2022, doi: 10.5194/amt-15-5033-2022.

[25] S. Pfreundschuh, C. Guilloteau, P. J. Brown, C. D. Kummerow, and P. Eriksson, "GPROF V7 and beyond: assessment of current and potential future versions of the GPROF passive microwave precipitation retrievals against ground radar measurements over the continental US and the Pacific Ocean," Atmospheric Measurement Techniques, vol. 17, pp. 515-538, 2024, doi: 10.5194/amt-17-515-2024.

[26] M. Reichstein et al., "Deep learning and process understanding for data-driven Earth system science," Nature, vol. 566, pp. 195-204, 2019, doi: 10.1038/s41586-019-0912-1.

[27] K. Hsu, X. Gao, S. Sorooshian, and H. V. Gupta, "Precipitation estimation from remotely sensed information using artificial neural networks," Journal of Applied Meteorology, vol. 36, no. 9, pp. 1176-1190, 1997, doi: 10.1175/1520-0450(1997)036<1176:PEFRSI>2.0.CO;2.

[28] S. Pfreundschuh, I. Ingemarsson, P. Eriksson, D. A. Vila, and A. J. P. Calheiros, "An improved near-real-time precipitation retrieval for Brazil," Atmospheric Measurement Techniques, vol. 15, no. 23, pp. 6907-6933, 2022, doi: 10.5194/amt-15-6907-2022.

[29] M. Sadeghi, A. A. Asanjan, M. Faridzad, P. Nguyen, K. Hsu, S. Sorooshian, and D. Braithwaite, "PERSIANN-CNN: Precipitation Estimation from Remotely Sensed Information Using Artificial Neural Networks-Convolutional Neural Networks," Journal of Hydrometeorology, vol. 20, no. 12, pp. 2273-2289, 2019, doi: 10.1175/JHM-D-19-0110.1.

[30] V. Afzali Gorooh, A. A. Asanjan, P. Nguyen, K. Hsu, and S. Sorooshian, "Deep Neural Network High Spatiotemporal Resolution Precipitation Estimation (Deep-STEP) Using Passive Microwave and Infrared Data," Journal of Hydrometeorology, vol. 23, no. 4, pp. 597-617, 2022, doi: 10.1175/JHM-D-21-0194.1.

[31] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015, Lecture Notes in Computer Science, vol. 9351, pp. 234-241, 2015, doi: 10.1007/978-3-319-24574-4_28.

[32] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2117-2125, doi:

10.1109/CVPR.2017.106.

[33] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 4489-4497, doi: 10.1109/ICCV.2015.510.

[34] S. Pfreundschuh, P. Eriksson, D. Duncan, B. Rydberg, N. Håkansson, and A. Thoss, "A neural network approach to estimating a posteriori distributions of Bayesian retrieval problems," Atmospheric Measurement Techniques, vol. 11, pp. 4627–4643, 2018, doi: 10.5194/amt-11-4627-2018.

[35] A. Moraux, S. Dewitte, B. Cornelis, and A. Munteanu, "Deep Learning for Precipitation Estimation from Satellite and Rain Gauges Measurements," Remote Sensing, vol. 11, no. 21, Art. no. 2463, 2019, doi: 10.3390/rs11212463.

[36] A. Moraux, S. Dewitte, B. Cornelis, and A. Munteanu, "A Deep Learning Multimodal Method for Precipitation Estimation," Remote Sensing, vol. 13, no. 16, Art. no. 3278, 2021, doi: 10.3390/rs13163278.

[37] C. Kidd, D. Kniveton, M. C. Todd, and T. J. Bellerby, "Satellite rainfall estimation using combined passive microwave and infrared algorithms," Journal of Hydrometeorology, vol. 4, no. 6, pp. 1088-1104, 2003, doi: 10.1175/1525-7541(2003)004<1088:SREUCP>2.0.CO;2.

[38] C. Wang, G. Tang, and P. Gentine, "PrecipGAN: Merging microwave and infrared data for satellite precipitation estimation using generative adversarial network," Geophysical Research Letters, vol. 48, no. 5, e2020GL092032, 2021, doi: 10.1029/2020GL092032.

[39] P. Sano, G. Panegrossi, D. Casella, F. Di Paola, L. Milani, A. Mugnai, M. Petracca, and S. Dietrich, "The Passive microwave Neural network Precipitation Retrieval (PNPR) algorithm for AMSU/MHS observations: description and application to European case studies," Atmospheric Measurement Techniques, vol. 8, pp. 837-857, 2015, doi: 10.5194/amt-8-837-2015.

[40] L. Bagaglini, P. Sano, D. Casella, E. Cattani, and G. Panegrossi, "The Passive Microwave Neural Network Precipitation Retrieval Algorithm for Climate Applications (PNPR-CLIM): Design and Verification," Remote Sensing, vol. 13, no. 9, Art. no. 1701, 2021, doi: 10.3390/rs13091701.

[41] F. J. Turk et al., "Adapting Passive Microwave-Based Precipitation Algorithms to Variable Microwave Land Surface Emissivity to Improve Precipitation Estimation from the GPM Constellation," Journal of Hydrometeorology, vol. 22, no. 7, pp. 1755-1781, 2021, doi: 10.1175/JHM-D-20-0296.1.

[42] S. W. D. Chien, S. Markidis, C. P. Sishtla, L. Santos, P. Herman, S. Narasimhamurthy, and E. Laure, “Characterizing Deep-Learning I/O Workloads in TensorFlow,” in 2018 IEEE/ACM 3rd International Workshop on Parallel Data Storage and Data Intensive Scalable Computing Systems (PDSW-DISCS), 2018, pp. 54–63, doi: 10.1109/PDSW-DISCS.2018.00011.

[43] S. Pumma, M. Si, W. Feng, and P. Balaji, “Scalable Deep Learning via I/O Analysis and Optimization,” ACM Transactions on Parallel Computing, vol. 6, no. 2, pp. 1–34, 2019, doi: 10.1145/3331526.

[44] G. Leclerc, A. Ilyas, L. Engstrom, S. M. Park, H. Salman, and A. Madry, “FFCV: Accelerating Training by Removing Data Bottlenecks,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023, pp. 12011–12020, doi: 10.1109/CVPR52729.2023.01156.

[45] V. J. Reddi et al., “MLPerf Inference Benchmark,” in 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), 2020, pp. 446–459, doi: 10.1109/ISCA45697.2020.00045.

[46] A. Zlateski, K. Lee, and H. S. Seung, “ZNNi: Maximizing the Inference Throughput of 3D Convolutional Networks on CPUs and GPUs,” in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2016, pp. 854–865, doi: 10.1109/SC.2016.72.

A

Supplementary Results

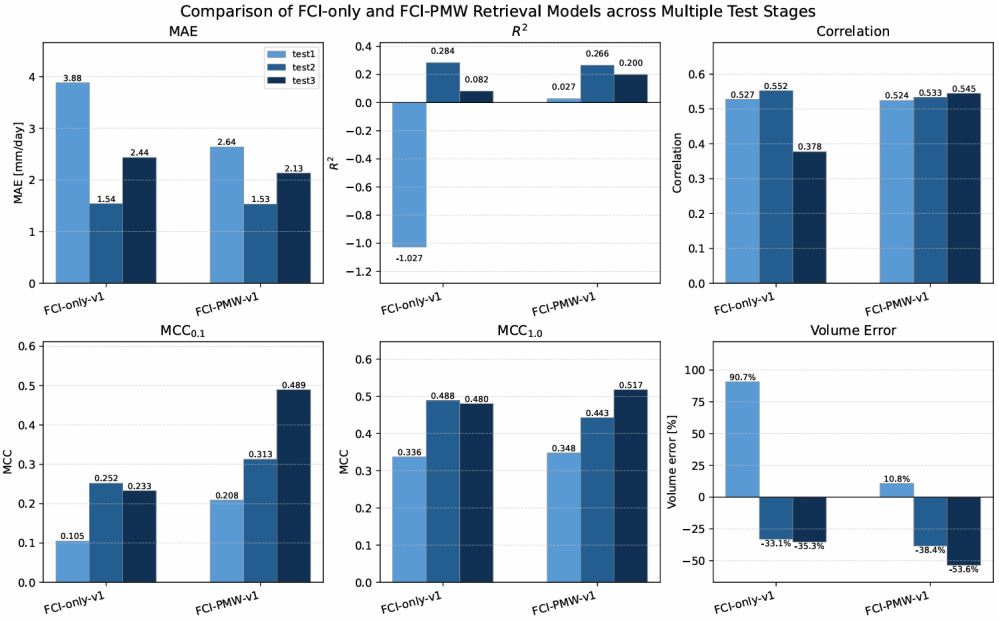

Figure A.1: Comparison of FCI-only V1 and FCI+PMW V1 retrieval performance across the three evaluation periods.