Precipitation Nowcasting using Nowcasting-GPT

by

Zeineh Bou Cher

to obtain the degree of Master of Science
at the Delft University of Technology,

Acknowledgments

I am profoundly thankful to my advisor, Prof. Dr. Ir. Justin Dauwels, for his extensive guidance, enduring patience, and the generous dedication of his time throughout my academic endeavor. I also express my deep gratitude to my daily supervisor, Cristian Meo, whose detailed guidance and essential input crucially shaped the trajectory of my research. The Signals and Systems Group at TU Delft University also deserves my thanks for their support and for creating an environment that significantly aided my work.

I must acknowledge my colleague, Ankush Roy, whose insightful discussions and steadfast encouragement played a vital role in the completion of my thesis. His companionship has greatly enriched and brought joy to this scholarly journey. I extend my deepest appreciation to my family—my parents, siblings, and my husband, whose unwavering support, profound love, and encouragement have been fundamental to my progress. Their constant motivation was particularly vital during the most challenging periods.

Finally, I extend my gratitude to everyone who lent their support throughout this project. Your encouragement and support have been invaluable. I am deeply appreciative of all contributions, whether emotional or academic.

Zeineh Bou Cher
Delft, June 2024

Abstract

Intense precipitation can have extensive economic outcomes, from disrupting outdoor activities to causing severe infrastructural damage, such as landslides, and endangering public safety. The urgency to mitigate these impacts underscores the need for improved early warning systems. Enhanced short-term weather prediction, or nowcasting, is critical for addressing these severe weather events effectively. Traditional meteorological forecasting methods, while foundational, are often constrained by simplistic physical assumptions and fail to capture the complex, nonlinear patterns of intense weather events. These methods also struggle with high computational demands and lack the resolution needed to detect crucial microscale atmospheric phenomena for accurate short-term forecasts.

To address these challenges, this research introduces a novel deep learning approach utilizing a streamlined architecture that combines a Vector Quantized Variational Autoencoder (VQVAE) and an Autoregressive (AR) Transformer. This model aims to predict weather conditions up to 180 minutes ahead, using data analyzed at 30-minute intervals. The proposed model displays comparable performance with the state-of-the-art conventional methods and other deep learning nowcasting models in predicting precipitations and sometimes extreme events. This study seeks to enhance forecasting accuracy and efficiency, providing valuable contributions to the field of meteorological nowcasting.

Contents

List of Figures

1 Introduction

1.1. Background

Intense precipitation can have extensive economic outcomes, starting with its impact on outdoor activities, leading to event postponements or cancellations. This disruption extends to ground transportation delays, flight cancellations, power station interruptions, and marine service terminations. Additionally, such precipitation can devastate agricultural crops, escalating economic pressures. As precipitation intensity increases, the consequences transition from merely disrupting events to posing substantial risks to infrastructure, including landslides, endangering public safety, and, in severe instances, threatening human lives.

To mitigate these far-reaching consequences, the implementation of an early warning system becomes crucial. Such a system, exemplified by Weather Nowcasting, empowers governments and responsible entities to take timely action and prevent hazards. Nowcasting predicts rainfall intensities over a specific region for a short period, typically up to 6 hours, serving as a vital tool for proactive disaster management.

The variability of weather phenomena across spatial scales, ranging from global to synoptic, mesoscale, and microscale in descending order of size, presents a complex challenge for meteorological forecasting. Numerical Weather Prediction (NWP) constitutes the primary methodology for weather prediction, relying on sophisticated mathematical models to simulate atmospheric and oceanic conditions. However, NWP encounters notable limitations. Firstly, its computational demands are substantial due to the intricacy of the mathematical algorithms employed. Additionally, NWP exhibits constraints in temporal and spatial resolution, limiting its ability to detect microscale phenomena essential for short-term forecasting, such as convective patterns crucial for rainfall prediction. Despite advancements in computing technology enhancing NWP's resolution in recent times, it remains inferior to the precision attained by nowcasting systems. This deficiency may arise from the chaotic nature of the atmosphere, posing inherent challenges to its mathematical representation. To address this gap, radar-extrapolation techniques have emerged. These methods utilize radar observations of the current atmospheric state to predict future weather conditions. They offer the advantages of simpler models, higher resolution, and enhanced forecasting accuracy within the first few hours (typically less than 6 hours). Currently, radar-extrapolation approaches remain essential components of the majority of operational nowcasting systems.

In recent years, researchers have incorporated deep learning models into nowcasting tasks. These advanced models, akin to traditional approaches, predominantly leverage radar precipitation fields to represent weather conditions and forecast future precipitation patterns. Among the pioneering models is ConvLSTM by Shi et al.[1], which treats nowcasting as a video prediction task, employing convolution within LSTM to simultaneously capture spatial and temporal features. Another approach treats nowcasting as an image transformation task, utilizing the U-Net structure. The success of deep generative models like GAN and VAE in various fields has inspired researchers to integrate them into nowcasting. For instance, DeepMind (Google) [2] developed a conditional GAN model for nowcasting purposes, aiming for predictions within the 90-120 minute timeframe. Additionally, Bi et al.[3] in 2023 implemented a VQGAN model with an EVL loss function, focusing on extreme precipitation nowcasting. The work of Bi et al. has served as a benchmark for subsequent research in this area.

Compared to traditional radar-extrapolation methods, deep learning models offer several advantages. They are data-driven, flexible, and not constrained by explicit physics requirements. Moreover, they possess superior non-linear modeling capabilities facilitated by the use of activation functions, crucial for capturing complex weather events like convection initiation. However, challenges such as blurry results and interpretability persist in deep learning nowcasting, setting it apart from other well-established deep learning tasks.

1.2. Research: Motivation and Goal

Despite these advancements, Nowcasting still faces challenges in delivering precise and efficient short-term weather predictions. Traditional NWP models often fall short within the crucial zero to two-hour forecast window due to their high computational load and coarse resolution. Similarly, radar-based methods can oversimplify atmospheric dynamics, missing transient details crucial for accurate predictions.

Deep learning presents a promising solution. Models like ConvLSTM and UNet have shown potential in leveraging large datasets to discern complex atmospheric behaviors. However, these models often produce overly smooth forecasts lacking fine detail, primarily due to their reliance on Mean Squared Error (MSE) for loss calculation.

Recent studies, including the work by Bi et al. [3], have shown that adversarial training can significantly enhance the realism and sharpness of nowcasting predictions. Inspired by these advances, this thesis aims to develop a novel deep generative model using a streamlined architecture that incorporates a Vector Quantized Variational Autoencoder (VQVAE) and an Autoregressive (AR) Transformer. This model is designed to improve upon both computational efficiency and forecasting accuracy.

The goal is to predict weather conditions up to 180 minutes ahead using data from the preceding 90 minutes, analyzed at 30-minute intervals. The project will focus on optimizing the VQVAE for effective data representation and experimenting with various transformer configurations to refine the model's predictive capabilities. The effectiveness of the model will be evaluated comprehensively across different regions and scales within the Netherlands.

1.3. Thesis Outline

This thesis explores the complexities of nowcasting using deep learning and generative models, systematically building on the knowledge from each chapter to conduct a comprehensive evaluation of the proposed model.

1. Chapter 2 : provides a thorough examination of existing nowcasting methods. It traces developments from traditional Numerical Weather Prediction (NWP) and radar-based techniques to cutting-edge deep learning models, assessing their strengths and limitations. The chapter also delves into the latest advancements in generative models for nowcasting, setting the stage for the methodologies introduced in this research.
2. Chapter 3: delves into the specifics of the dataset obtained from the Royal Netherlands Meteorological Institute (KNMI). It encompasses a detailed examination of data preprocessing techniques and provides a statistical analysis of the dataset to understand its characteristics better. Additionally, it outlines the criteria and process for selecting meteorological events relevant to the study.

The chapter concludes with a comprehensive discussion on how the data is divided into training, validation, and testing sets, ensuring robust model training and evaluation.

3. Chapter 4: presents the theoretical foundations and practical aspects of the deep generative model used in this thesis. It elaborates on the basic principles and functionalities of the Vector Quantized Variational Autoencoder (VQVAE) and the Autoregressive (AR) Transformer. The discussion extends to a detailed examination of the loss functions used for training these models. Furthermore, the chapter explains how these models are combined and fine-tuned, illustrating their collaborative effectiveness in improving the precision and efficiency of nowcasting predictions.
4. Chapter 5: introduces the comprehensive set of evaluation metrics used to assess the accuracy of weather predictions generated by the VQVAE and AR Transformer models. The evaluations focus on the Netherlands, specifically targeting the entire region and the catchment areas.
5. Chapter 6: presents a detailed analysis of various Transformer configurations, exploring their effectiveness at both the national and catchment levels. It includes an in-depth study of the attention mechanisms within the models, providing insights into how these mechanisms influence the predictive accuracy of the nowcasting models.
6. Chapter 7: offers a detailed examination of the findings, placing the results in the context of established nowcasting technologies such as PySteps and Nuwa-Pytorch. It evaluates the strengths and shortcomings of the proposed approach relative to these models, with a comparative analysis that encompasses results from both the entire Netherlands region and specific catchment areas.
7. Chapter 8: concludes by summarizing the key findings and contributions of the research. It also outlines potential future directions for extending the work, including integration with other meteorological forecasting systems and exploring new generative model architectures.

2 Literature Review

2.1. Introduction

In this chapter, we embark on a comprehensive exploration of nowcasting methods, tracing the evolution from traditional approaches to the forefront of modern technological advancements. We begin with a deep dive into traditional methods like Numerical Weather Prediction (NWP) and radar-based techniques, assessing their strengths and limitations. Subsequently, we transition into sophisticated deep learning approaches that leverage large datasets to capture complex atmospheric dynamics more effectively. By critically analyzing these methodologies, this literature review aims to lay a solid foundation for understanding the current capabilities and identifying future directions in meteorological forecasting.

2.2. Traditional nowcasting methods

Numerical Weather Prediction

Numerical Weather Prediction (NWP) models serve as the foundation of meteorological forecasting, employing complex mathematical formulations to predict atmospheric dynamics. While these models possess the ability to generate forecasts across various timescales and demonstrate comprehensive capabilities, they demand substantial computational resources and often exhibit limitations in very short-term forecasts, particularly within the zero to two-hour window [4]. This scenario underscores a critical trade-off in operational meteorology: balancing the demand for quick, accurate forecasts with the computational demand imposed by NWP models.

Radar-Based Approach: Euler and Lagrangian Persistance Methods

To tackle this challenge, radar-based approaches present a practical solution, alleviating the computational load associated with NWPs. These methods predominantly rely on the assumption of a steady-state condition in atmospheric advection fields to forecast precipitation dynamics. However, this simplification of the forecasting process comes with the risk of overlooking transient and dynamic atmospheric variables that significantly influence precipitation patterns.

Within the framework of these radar-based strategies, Eulerian and Lagrangian persistence (as highlighted by Germann et al. [5]) play pivotal roles. The Eulerian model assumes that the future precipitation state at a specific position x and future time $t_0+\tau$ will mirror the most recent observations at the initial time t_0, as represented by Equation 2.1.

$$\hat{\Psi}(t_0 + \tau, x) = \Psi(t_0, x) \tag{2.1}$$

Conversely, the Lagrangian perspective operates under the assumption that air parcels maintain a consistent state, attributing any alteration in the precipitation field solely to the background flow [5]. As described in equation 2.2, the future state of precipitation at a specific location (x) and time ($t_0+\tau$) is influenced by the properties of air parcels at a previous position ($x-\lambda$) as well as the initial time (t_0). Most radar-based nowcasting methods are based on the Lagrangian persistence method.

$$\hat{\Psi}(t_0 + \tau, x) = \Psi(t_0, x - \lambda) \tag{2.2}$$

Although Lagrangian persistence has established itself as a pillar in precipitation nowcasting, its assumptions often fall short when predicting the actual movement of precipitation. To improve the accuracy of nowcasting models, there is a shift towards integrating probabilistic and stochastic methods. These approaches not only consider the advection process but also introduce uncertainty into the forecasts, providing more adaptable approach to Lagrangian persistence by accommodating variations within the advection field.

PySTEPS

Introducing PySTEPS, an open-source library for radar-based nowcasting [6], combines the optical flow method with advanced extrapolation techniques in an integrated manner. The optical flow method typically involve two key stages: first, the motion field is inferred from the observed data, and then the recent observations are extrapolated forward along the motion field to generate a forecast.

PySTEPS presents two methodologies: daterministic method which employs the S-PROG algorithm (Spectral PROGnosis) and the probabilistic method which utilizes the STEPS algorithm (short-term ensemble prediction system). STEPS integrates nowcasting results with downscaled NWP (Numerical Weather Prediction) outcomes. The workflow of the Pysteps follows same configuration as mentioned in [[6]]. In particular, PySTEPS operates in the following manner:

1. Analyze radar composites, convert radar reflectivity data into rainfall measurements (mm/h), and then apply a logarithmic transformation to represent the results on a dB scale.
2. Employ the optical flow method to estimate the motion of precipitation, capturing both velocity and direction.
3. Incorporate an advection scheme to extrapolate the position and intensity of precipitation fields, thereby enhancing predictions for short-term timescales.
4. Utilize Fast Fourier Transform (FFT) to decompose the rainfall field into a multiplicative cascade, where each level represents a distinct spatial scale and rainfall intensity.
5. Compute the auto-correlation matrix for each level of the cascade, then derive parameters for an AutoRegressive (AR) model using Yule-Walker equations. Apply this model over time to address temporal changes and correlations within the precipitation structure. The AR model is expressed as follows:

   $$R_j(x,y,t) = \phi_{j,1}R_j(x,y,t-\Delta t) + \phi_{j,2}R_j(x,y,t-2\Delta t) \tag{2.3}$$

   Here, R denotes the radar map, (x,y,t) represents coordinates, ϕ_j indicates the model parameter, and j signifies the number of cascade levels.
6. Introduce stochastic perturbations to the AR models and advection field to accommodate uncertainties in rainfall intensities and motion fields.
7. Reassemble the cascade by integrating the AR model and stochastic perturbations to generate the result of the nowcasting ensemble.

PySTEPS represents a significant advancement in nowcasting, it bridges the gap between the need for rapid, computationally efficient forecasts and the forecast accuracy taking into account the meteorological complexity. By providing a probabilistic framework and allowing for the incorporation of ensemble techniques, PySTEPS not only addresses the deterministic limitations of the Eulerian and Lagrangian models but also provides a platform for the continual refinement and evolution of nowcasting capabilities [6]. Recently adopted by Royal Netherlands Meteorological Institute (KNMI) as their new operational nowcasting system, STEPS will also serve as the benchmark for this thesis project, providing a well-established and robust point of comparison.

2.3. Deep-learning nowcasting methods

The unpredictable and rapidly changing conditions of the atmosphere present significant challenges to traditional nowcasting methods. Eventhough the NWP models and radar-based extrapolation techniques constitute the foundation of weather forecasting, they often struggle with the real time prediction of short term precipitation due to their reliance on complex mathematical equations and forecasting advection fields. These traditional approaches are constrained by assumptions that do not fully account for the atmospheric dynamic, leading to inaccuracies in predicting sudden weather changes. Furthermore, the extensive computational resources and prolonged processing times required by these methods hinder their effectiveness in nowcasting scenarios , where forecasts need to be both timely and accurate.
Deep learning models stand out as a superior alternative for meeting the demands of nowcasting, thanks to their unparalleled ability to process and learn from extensive datasets. These models excel in identifying complex, nonlinear relationships among atmospheric variables, surpassing the predictive capabilities of traditional methods that depend heavily on extrapolating from historical data. Deep learning's proficiency in processing large volumes of data rapidly addresses the critical computational limitations in NWP and enhances the accuracy of nowcasting, offering a powerful tool to navigate the complexities of the atmosphere.

Spatio-Temporal Models
Predicting rainfall precipitation presents a complex challenge that involves spatial-temporal modeling, necessitating an approach adept at handling both the spatial distribution and the temporal evolution of precipitation patterns. Given their strengths in temporal modeling and time series prediction, Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM) networks, both variants of Recurrent Neural Networks (RNN), are particularly promising for weather forecasting. Notably, LSTMs feature a unique gating mechanism that retains and utilizes information over prolonged periods, aiding in modeling the intricate dependencies crucial for accurate weather forecasting across both time and space. In response to this challenge, Shi et al. [1] introduced ConvLSTM, the first deep learning method employed in precipitation nowcasting.

The innovative aspect of ConvLSTM lies in its approach in treating nowcasting similar to video prediction by replacing the fully connected layers (typical of standard LSTMs) with convolutional layers. This modification allows the model to process radar images as $3D$ tensors, applying convolution across different kernel sizes to the input images. This approach enables the extraction of both spatial and temporal features essential for accurate precipitation forecasting.

ConvLSTM has established itself as a foundational architecture for research into deep learning models for nowcasting. Based on this, a novel model known as Trajectory GRU ([7]) has been developed, which is proficient at learning structures that change location over time. Unlike ConvLSTM, Trajectory GRU is capable of aggregating states along dynamically learned paths. Moreover, recent advancements include the integration of attention and context matching mechanisms to enhance the model's long-term forecasting capabilities and interpretability. Studies indicate that these enhanced models surpass the original ConvLSTM in nowcasting accuracy when evaluated using Mean Squared Error (MSE) metrics.

UNet Models
Instead of using LSTM and Trajectory GRU, which view weather forecasting as a task of predicting spatio-temporal patterns, some researchers approach nowcasting from a different angle by viewing it as a form of image-to-image translation. To tackle this, researchers employ well-known $UNet$ architecture, recognized for its convolutional neural network (CNN) design and characteristic "U" shape. $UNet$ processes either a single radar image or a sequence of radar images to produce a single predictive image. However, due to the absence of a dedicated memory component in $UNet$, the prediction process occurs recursively.

Furthermore, researchers have explored enhancements to the fundamental $UNet$ architecture to improve its forecasting accuracy and operational efficiency. One notable advancement is the development of $SmaAtUNet$ [8], which incorporates attention mechanisms and depthwise separable convolutions. This model utilizes Convolutional Block Attention Modules ($CBAMs$) to sequentially prioritize channels and spatial regions of input images, leading to more effective feature extraction. Remarkably, despite having only a quarter of the parameters of $UNet$, $SmaAtUNet$ achieves performance that is approximately similar [8].

Incorporating attention mechanisms expands the opportunities for developing new, innovative models. Bojesomo et al[9] introduced a model that utilizes the 3D Swin Transformer within a $UNet$ framework. This model incorporates essential elements such as patch merging, expanding layers, and a multistage encoding process with the Swin Transformer. The Swin Transformer employs a localized self-attention mechanism, focusing on specific features before shifting its attention to capture a wide range of interactions. Additionally, the decoder employs a cross-attention mechanism to integrate the encoder's output with its input, significantly enhancing the model's ability to synthesize information across different stages of processing.

In comparison to spatio-temporal networks, $UNet$ models offer greater adaptability to the forecast period, which can potentially improve the accuracy of short-term forecasting. Additionally, $UNet$ benefits from a simpler and more straightforward training process. However, concerns have been raised about the recursive nature of this forecasting method, as it may lead to increased error accumulation over longer forecast periods [10].

2.4. Deep-Generative methods

Deep learning models have demonstrated potential in improving nowcasting accuracy; however, these models often produce blurred forecasts when extending the prediction period. This blurry effect primarily stems from the use of mean square error (MSE) as a loss function, which encourages models to average all possible outcomes. While these models perform well in predicting low rainfall intensities, their effectiveness diminishes for high-intensity rainfall events. Furthermore, most existing deep learning approaches are limited as they focus on specific locations and temporal resolutions, rather than providing probabilistic forecasts that cover entire precipitation fields. This lack of generalizability significantly constrains their operational utility, as they fail to deliver consistent predictions across diverse spatial and temporal scales.

In contrast, deep generative models present a robust alternative to traditional forecasting methods, excelling at learning the probability distributions of data. These models generate forecasts based on current radar observations and are capable of delivering skillful probabilistic predictions, potentially overcoming the limitations of conventional deep learning techniques. Generative Adversarial Networks (GANs) [11] play a crucial role in this advancement by enhancing the accuracy and realism of weather predictions. The adversarial training within GANs is particularly effective in generating more detailed predictions and capturing a broader range of meteorological phenomena.

The GA-ConvGRU model, developed by Tian et al. [12], integrates Generative Adversarial Networks (GANs) with a convolutional Gated Recurrent Unit (convGRU) generator, supplemented by a discriminator consisting of five CNN layers (shown in figure 2.1). Employing an encoder-decoder architecture, the system analyzes sequences of radar images for weather forecasting.

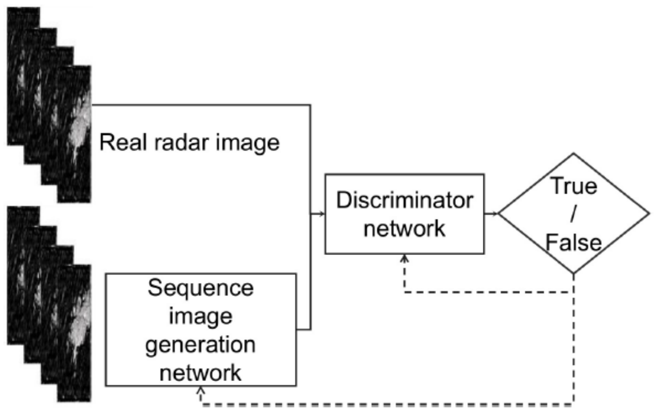


Figure 2.1: A schematic diagram of the model GA-ConvGRU [12]

The generator employs a multistep approach to forecast future frames. Initially, the input sequence undergoes multiple downsampling stages, followed by ConvGRU processing. This encoding process enables the model to compress information into abstract hidden representations, facilitating a more compact representation of the input data.
During the decoding phase, the decoder module begins by generating a predicted image sequence. This process starts with feeding the last hidden state into the corresponding ConvGRU unit, followed by applying an upsampling operator to produce the initial prediction. Subsequently, the outputs from each upsampling stage are subject to iterative refinement. During this refinement, the outputs are integrated with hidden states from the corresponding downsampling stage. This iterative process, guided by spatial and temporal information encoded in the hidden representations, facilitates the accurate prediction of future frames. Additionally, it progressively enhances the level of detail captured in the forecasted images.
The discriminator functions as a binary classifier and plays a crucial role in differentiating between synthetically generated sequences and actual observational data. It evaluates ten pairs of sample images; each pair consists of a predicted image alongside its corresponding real radar observation. The goal is to accurately identify which images are real.
The comprehensive loss for the $GA-ConvGRU$ model, denoted as $\mathcal{L}_{total}$, merges the mean square error loss with the adversarial loss (eq:2.4)[12]. This adversarial training strategy allows the generator to overcome the inherent limitations of mean square loss, resulting in predictions that are both more precise and realistic.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MSE}} + \min_{G}\max_{D} V(D, G) \tag{2.4}$$

Building upon this foundation, Jing et al. introduced the Adversarial Extrapolation Neural Network ($AENN$) model [13]. The $AENN$ model employs a GAN framework that features a conditional generator with an encoder-decoder architecture. Initially, the encoder transforms input radar images into a feature space, effectively capturing spatial attributes.These features are then processed by a $ConvLSTM$ to model temporal dynamics and forecast future states. The forecasts are subsequently converted back into image format by the decoder. The model utilizes two discriminators: one evaluates individual frames (echo-frame discriminator), and the other assesses the sequence as a whole (echo-sequence discriminator). Notably, the $AENN$ model demonstrates remarkable proficiency in forecasting radar echoes with detailed texture. However, it does exhibit a reduction in predictive accuracy with longer forecast duration.

Beyond the GA-ConvGRU and AENN models, numerous deep generative models have emerged to enhance radar-based weather forecasting. Ravuri et al. [2] introduced the Deep Generative Model for Radar ($DGMR$), which integrates two discriminators and a regularization term into its training process. The $DGMR$ model shows improvements in location accuracy, capturing small-scale weather phenomena, maintaining statistical properties of precipitation, and reducing blurry predictions. However, it faces challenges in accurately predicting heavy rainfall intensity over extended lead times or during extreme precipitation events.

Addressing the challenge of predicting high-intensity rainfall, Bi et al. [3] proposed the Vector Quantization Generative Adversarial Network with Transformer ($VQGAN + Transformer$). This model (2.2) incorporates a Vector Quantized Variational Autoencoder ($VQVAE$) to compress radar data into a latent space representation, along with a spatial discriminator and an auto-regressive transformer ($AR - transformer$). The attention mechanism used in the transformer, known as $3DNA$, is well-suited to the inherent $3D$ data structure of radar observations. To handle extreme weather events, Bi et al. introduced the concept of an extreme value loss (EVL), based on the extreme value theory, which emphasizes the accurate prediction of rare but crucial events. By integrating EVL with the cross-entropy loss of the transformer, the $VQGAN + Transformer$ model enhances its ability to handle data imbalances and focus on rare events, thereby improving overall forecasting performance.

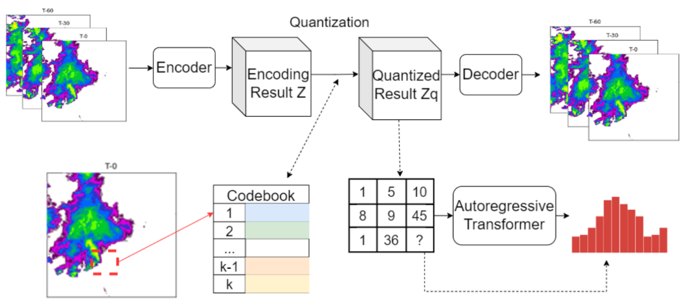


Figure 2.2: A schematic diagram of the model VQGAN [3]

Climax [14] extends the transformer architecture with vector tokenization and aggregation blocks to handle heterogeneous data with varying spatial and temporal resolutions. In this model, vector tokenization divides input variables into patches and linearly embeds them into vectors, which provides flexibility but can lead to high computational complexity. To mitigate this, Climax employs variable aggregation, which reduces sequence length by performing cross-attention for each spatial position. The model utilizes a standard Vision Transformer (ViT) to generate output tokens, followed by a prediction head that outputs the final predictions. These enhancements improve Climax's capability to handle diverse climate data, making it versatile, adaptable, and precise, even when dealing with different spatial and temporal resolutions.

These diverse approaches within the realm of deep generative models highlight ongoing efforts to address the challenges associated with radar-based weather forecasting. From enhancing spatial and temporal consistency to accurately predicting extreme weather events, each model contributes unique insights and methodologies. Together, they significantly advance the predictive capabilities of meteorology.

2.5. Conclusion

This chapter has reviewed the evolution of nowcasting methods, from traditional Numerical Weather Prediction and radar-based techniques to advanced deep learning and generative models. As we move forward, the next chapter will delve into data analysis, focusing specifically on the radar images provided by the Royal Netherlands Meteorological Institute (KNMI). This dataset will be pivotal in training and testing the deep generative model, which is central to evaluating the effectiveness and efficiency of these advanced forecasting techniques.

3 Data

3.1. KNMI Radar Dataset

Weather radars are utilized to monitor precipitation patterns. The Royal Netherlands Meteorological Institute (**KNMI**) operates two advanced C-band weather radars, located in Den Helder and Herwijnen. The latter replaced an earlier radar that was operational in De Bilt until 2017 [15]. Equipped with dual-polarized technology, these radars accurately gauge the initial radar reflectivity, denoted as Z, which quantifies the amount of radiation reflected back at an altitude of 1500 meters.
Radar reflectivity is used to estimate the rainfall rate using a fixed Z-R transformation equation [16], expressed as:

$$Z_h = 200.R^{1.6} \tag{3.1}$$

where Z_h represents the radar reflectivity in (mm^6/mm^3), and R represents the rainfall rate in (mm/hr). Intially, the radar reflectivity Z_h is expressed in dBZ which can be converted to mm^6/mm^3 using the formula $10log_{10}(Z_h)$.In this process, reflectivity below $7dB$ (precipitation intensity $0.1mm/hr$) are neglected and reflectivity above $55dB$ (precipitation intensity $100mm/hr$) are fixed at $55dB$. Following the aforementioned conversion process, the data becomes available as Quantitative Precipitation Estimation (QPE) data on the KNMI website, where it is recognized as the RT dataset. Figure 3.1 shows an example of an RT dataset over different time stamps.

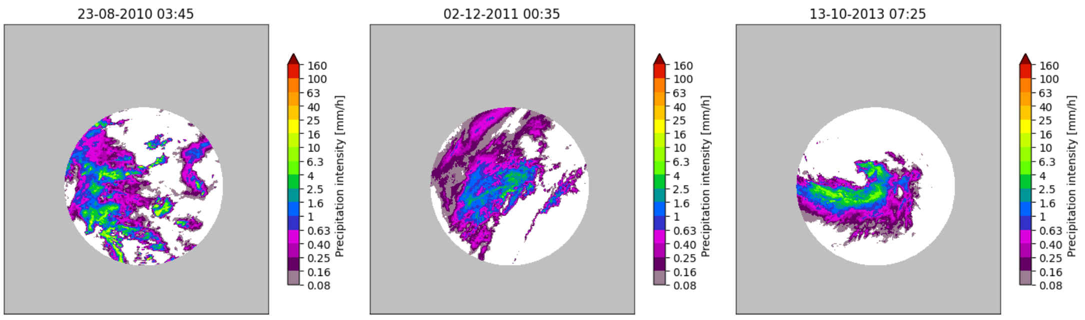


Figure 3.1: Radar maps of the real time dataset at three different time stamps.

KNMI provides an additional dataset known as the MFBS dataset, which exclusively covers the Netherlands area. Figure 3.2 shows an example of the MFBS dataset at different time scales. This dataset maintains the same temporal and spatial resolution as the RT dataset. Calibration of the RT dataset utilizes data collected from 31 automatic and 325 manual rain gauges. Despite the superior accuracy of the MFBS data compared to the RT datasets, its update frequency is limited to monthly intervals due to the substantial labor required for manual measurements. Consequently, the MFBS dataset is employed primarily for event selection, whereas the RT dataset is utilized for training and testing purposes.

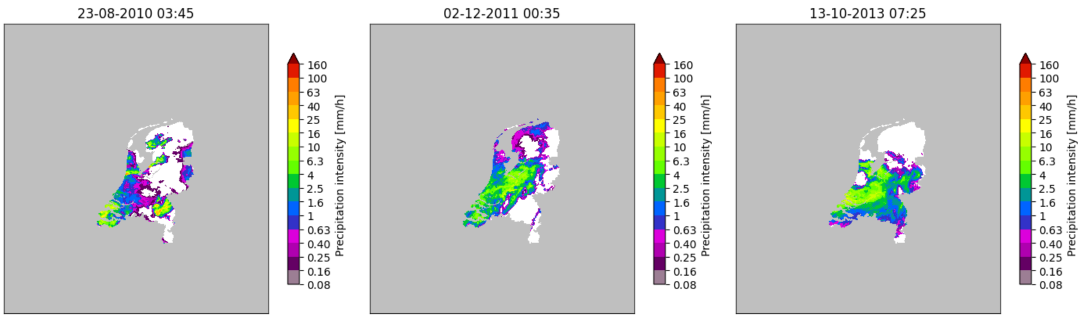


Figure 3.2: Radar maps of MFBS dataset at three different time stamps.

The real-time (RT) dataset contains radar images spanning the years 2008 to 2021. These radar maps have dimensions of 765×700 pixels, with a spatial resolution of 1 km and a temporal resolution of 5 minutes. Figure 3.1 presents an example from the RT dataset; it illustrates that a significant portion of the area displays unavailable data (represented by the grey area). Within this image, a circular area with a radius of 200 km, centered around De Bilt, is populated with data. This circle encompasses regions of the Netherlands as well as parts of Germany and Belgium. However, the primary objective of this thesis is the nowcasting of precipitation within the Netherlands. Therefore, the image is cropped to a size of 256×256 pixels, ensuring that it includes only the Netherlands (an example shown in fig: 3.3).

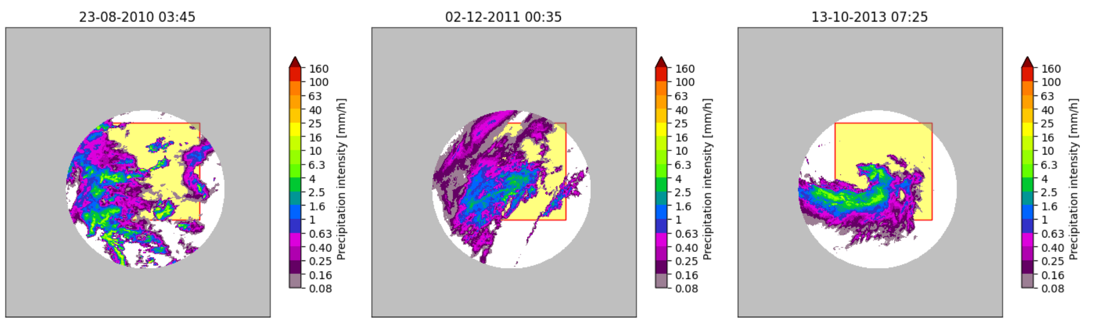


Figure 3.3: Radar maps of the real time dataset at three different time stamps cropped at the Netherland Region.

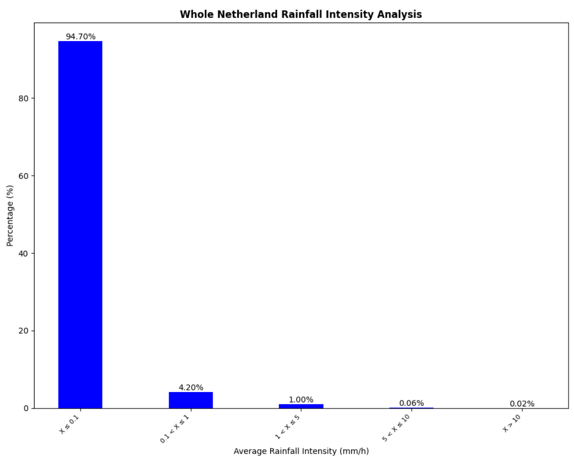
3.2. Data Analysis

To analyze the rainfall distribution over the Netherlands area, a dataset was compiled by consecutively sampling one radar image every hour from **00:00** on **January 1, 2008**, to **24:00** on **December 31, 2014**, resulting in a total of 60,000 images. The frequency of various precipitation intensities was recorded with a precision of $0.1mm/h$. The resulting data, detailed in 3.4a, offer a comprehensive overview of rainfall patterns during the period.

For enhanced visualization, the data is presented in a histogram 3.4b. This histogram reveals that over 94% of the recorded images show very light precipitation with intensities less than $0.01mm/h$, illustrating a predominance of low intensity rainfall events. Precipitation with intensities of $1mm/h$ is less frequently observed, and instances of rainfall exceeding $1mm/h$ are remarkably rare. This distribution underscores the occurrence of minimal rainfall in the Netherlands throughout the examined time frame.

Rainfall intensity X (mm/h)	Percent
$X \leq 0.1$	94.7%
$0.1 < X \leq 1$	4.2%
$1 < X \leq 5$	1.0%
$5 < X \leq 10$	0.06%
$X > 10$	0.02%

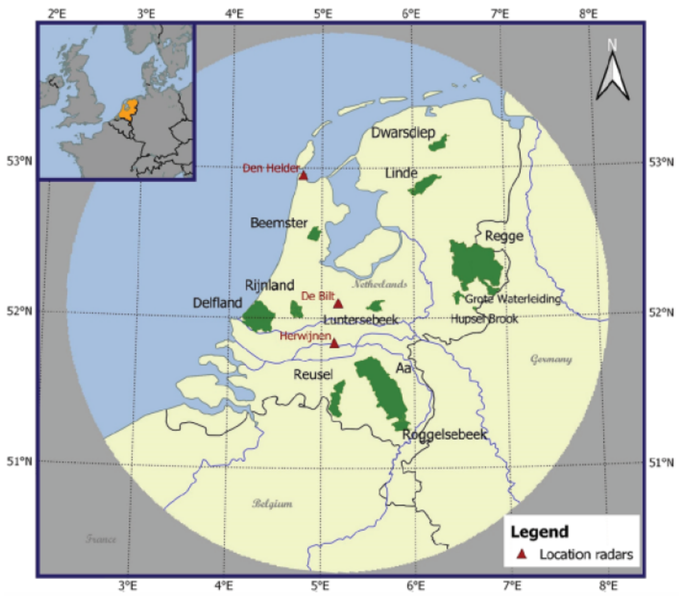
(a) Percentage of occurrence for different rainfall intensity thresholds.



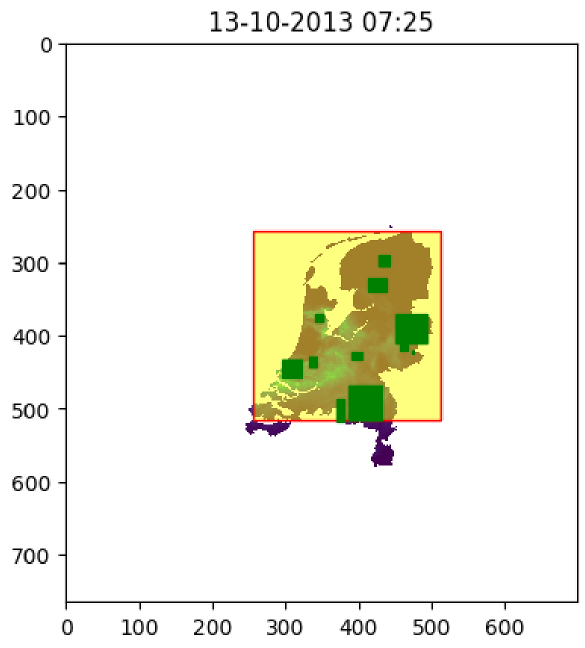
(b) Distribution of Rainfall Intensity Across the Netherlands.

Figure 3.4: Rainfall Intensity Distribution in the Netherlands: The table delineates the percentage occurrence of specified rainfall intensities, while the histogram visualizes the distribution of precipitation intensities across the selected RT image dataset.

To refine our analysis and the selection of significant events, it is crucial to examine the precipitation distribution in catchment areas. These areas are prone to flooding or water accumulation after periods of intense rainfall. Accurate monitoring of rainfall levels in these zones is critical to provide early flood warnings. In this study, 12 Dutch catchment areas will be analyzed. Their locations are illustrated in fig 3.5a, while their detailed positioning on the MFBS is outlined in fig 3.5b.



(a) Netherland map with the illustration of the radars location in de Bilt and Den Helder and the catchment locations.



(b) Figure illustrates the specific location of the catchments in our MFBS dataset

Figure 3.5: The location of the catchment areas and their detailed positioning on the MFBS

In the catchment level analysis, we averaged the rainfall accumulation over three-hour periods for each designated catchment area. This averaged data acts as the primary indicator of rainfall intensities for these areas. For consistency, the radar images analyzed were selected from the same period, 2008 to 2014, which aligns with the comprehensive analysis period for the entire Netherlands region. Figure 3.6 shows the statistical analysis of rainfall precipitation. It reveals that 91.1% of the catchment-averaged precipitation corresponds to pixels with no to light rain ($\leq 1mm/3hrs$), 7.57% corresponds to moderate rain ($1mm/3hrs < averageintensity \leq 5mm/3hrs$), and 1.33% corresponds to heavy rain ($> 5mm/3hrs$).

3.3. Event Selection

As mentioned earlier, the tedious process of manually collecting the measurements of the MFBS data limits its real time availability to be used for training purposes. There-

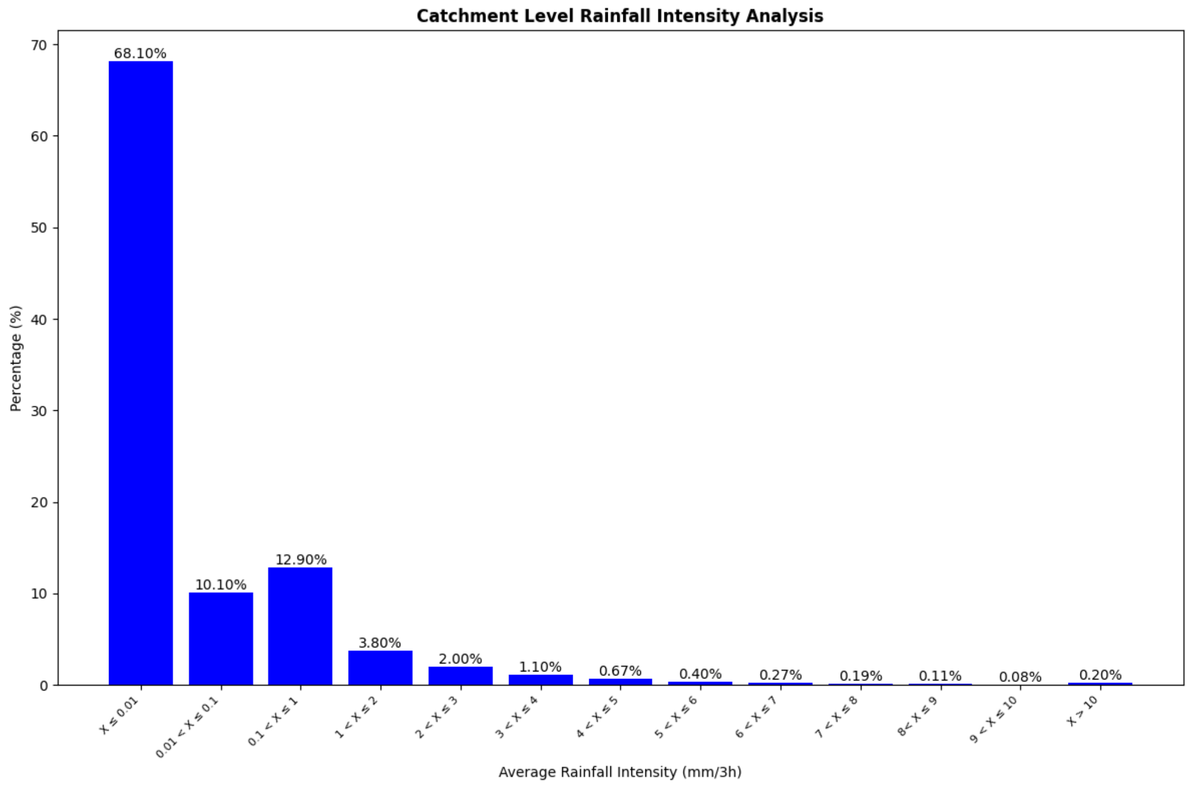


Figure 3.6: Distribution of Rainfall Intensities of the selected RT images based on catchment-level

fore, it will only be used for event selection. On the other hand, RT data will be used for training purposes as it mimics real time events. In this thesis, an event within a catchment is classified as extreme if it results in an average precipitation accumulation that exceeds the catchment's specific extreme threshold over a 3-hour period. To define this extreme threshold for a given catchment, the following steps are undertaken:

1. **Select a Catchment Area**: Choose a specific catchment as the primary focus for the study.
2. **Event Classification**: Determine whether each rainfall event is a non-rain event (average precipitation $< 0.1mm/3h$) or a rain event (average precipitation $\geq 0.1mm/3h$).
3. **Rank and Define Extremes**: Arrange the events by average precipitation and identify the top 1% as extreme events, establishing the threshold for extremes in that catchment.

Relying solely on extreme events (top 1%) does not provide a sufficient diverse dataset to train a deep learning model. Therefore, events that fall within the top 5% of precipitation, categorized as heavy rain events, are also included in the dataset. This expanded dataset is systematically divided into three parts: training, validation, and testing purpose. The training dataset consists of **30632** events from **2008** to **2014**; the validation dataset includes **3453** events from **2015** to **2017**; and the testing dataset, focused exclusively on the extreme events (top 1%), contains **357** events from **2019** to **2021**.

4 Methodology

This chapter explores the methodologies adopted to address the computational challenges of processing our radar dataset, which is fundamentally structured as 256×256 pixels. Due to the quadratic complexity of the attention mechanism, using a transformer for forecasting precipitation maps is computationally intensive. To tackle this, our model incorporates two main components: the Vector Quantized Variational Autoencoder (VQ-VAE) and the transformer. The VQ-VAE crucially reduces the input dimensions by learning discrete codes, making the computational process feasible. We provide a detailed examination of the architecture, functionality, and integration of the VQ-VAE and the AR Transformer, enhancing both the model's efficiency and its nowcasting accuracy. The complete model, illustrated in fig 4.1, demonstrates how these components work in unison to effectively manage and interpret meteorological data.

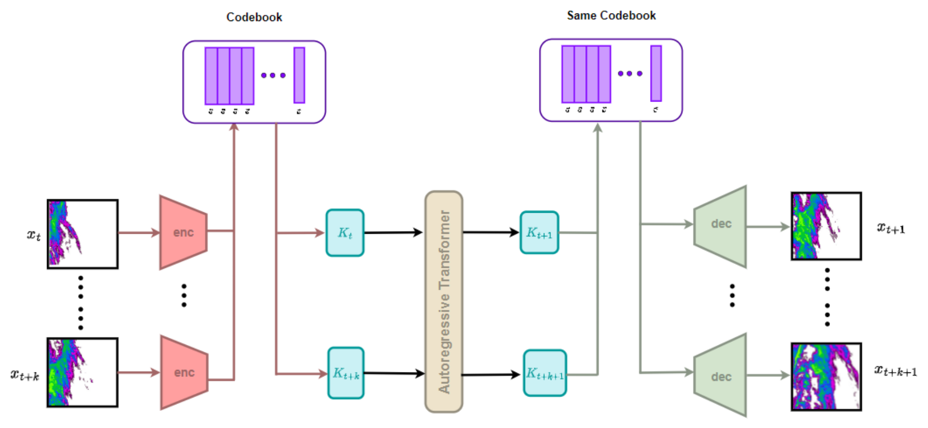


Figure 4.1: VQVAE in a combination with an AR Transformer used in Weather Prediction

4.1. Vector Quantized Variational Autoencoder

In their pivotal work, van den Oord et al. [17] introduced the Vector Quantized Variational AutoEncoder (VQ-VAE), an approach that diverges from traditional autoencoders. Unlike typical autoencoders that encode data into a continuous latent space, the VQ-VAE utilizes vector quantization, leveraging the distinct advantages of discrete latent structures. This method leverages the unique benefits of discrete latent structures, which are exceptionally adept at navigating the complexities associated with the spatial dimensions of data. The VQ-VAE architecture, illustrated in (figure 4.2), employs a predefined codebook that transforms input data into a compact and interpretable format. This approach significantly reduces the redundancy commonly associated with pixel-level feature processing.

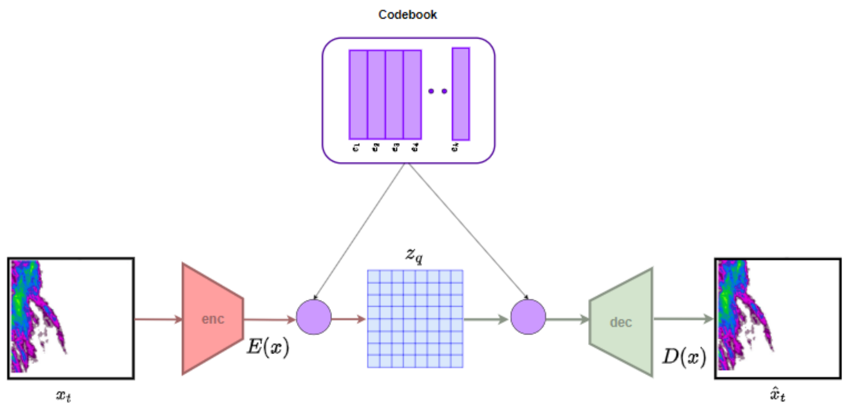
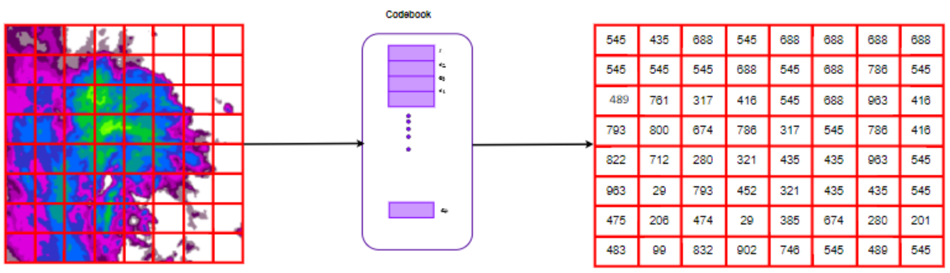


Figure 4.2: VQVAE Visualization

The encoder-decoder configuration of the VQ-VAE is designed to optimize data representation. The encoder progressively narrows the spatial dimensions while expanding the feature channels; thereby compressing the data and reducing irrelevant information. Each input is mapped into a discrete set of codebook entries, optimizing the encoding process while maintaining critical data features. Subsequently, the decoder reconstructs the input from these discrete codes, closely approximating the original data. This encoding-decoding process does not only preserve the integrity of the data but it also enhances its utility for practical applications such as image reconstruction.

When applying VQ-VAE to radar image processing, the first step is to encode the radar image x ($x \in \mathbb{R}^{H \times W}$), to produce a latent representation $\hat{z}$ ($\hat{z} \in \mathbb{R}^{h \times w \times n_z}$) where: H and W represent the height and width of the radar image, respectively; n_z is the embedding dimension; h and w are the spatial dimensions in the latent space.
This continuous latent space is then discretized into a set $\mathcal{Z} = \{z_k\}_{k=1}^{K}$ of K discrete codebook vectors, each with a dimension of n_z.

Upon encoding, each element of $\hat{z}$ is associated with the nearest codebook vector z_k. As a result, the radar image is generated as a set of discrete tokens (fig 4.3). It is worth noting that the selection of codebook vectors depends on the unique characteristics of each input image and the spatial dimensions output produced by the encoder.



545	435	688	545	688	688	688	688
545	545	545	688	545	688	786	545
489	761	317	416	545	688	963	416
793	800	674	786	317	545	786	416
822	712	280	321	435	435	963	545
963	29	793	452	321	435	435	545
475	206	474	29	385	674	280	201
483	99	832	902	746	545	489	545

Figure 4.3: VQ-VAE codebook Visualization

On the decoding side, discrete tokens serve as indices, each pointing to a specific codebook vector in $\mathcal{Z}$. These indices facilitate a lookup operation to fetch the corresponding vectors. The decoder utilizes these vectors to reconstruct the image, employing a series of strategic transformations and decisions to optimally combine the codebook vectors. The objective is to recreate an output that not only closely approximates the original input but also captures its essential characteristics with high fidelity.

To train the VQ-VAE model, a set of distinct loss functions are harnessed and optimized. These loss functions (eq: 4.1) encompass the reconstruction loss, the codebook loss, the commitment loss, and the perceptual loss.

$$\mathcal{L}(E, D, \mathcal{Z}) = \|x - \hat{x}\|_1 + \|\text{sg}[E(x)] - z_{\mathbf{q}}\|_2^2 \quad + \|\text{sg}\left[z_{\mathbf{q}}\right] - E(x)\|_2^2 + \mathcal{L}_{\text{perceptual}}(x, D(z)) \tag{4.1}$$

where:

$$z_{\mathbf{q}} = \mathbf{q}(\hat{z}) := \left(\underset{z_k \in \mathcal{Z}}{\arg\min} \|\hat{z}_{ij} - z_k\|\right) \in \mathbb{R}^{h \times w \times n_z}$$

$q(\cdot)$ is the element-wise quantization of each spatial code $\hat{z}_{ij} \in \mathbb{R}^{n_z}$,

$$\hat{z} = E(x) \in \mathbb{R}^{h \times w \times n_z},$$
$$\hat{x} = D(z_q),$$

and z_q is obtained using the encoder.
The reconstruction $\hat{x} \approx x$ is then given by

$$\hat{x} = D\left(z_{\mathbf{q}}\right) = D(\mathbf{q}(E(x))).$$

The reconstruction loss is the difference between the reconstructed image (obtained from the discrete latent codes) and the original input data. The main goal of the reconstruction loss is to urge the decoder to produce reconstructions that closely align with the original data. During back-propagation, the gradients of the reconstruction loss is calculated with respect to $z_{\mathbf{q}}$. This involves applying the chain rule to propagate the gradients backward through the decoder.

In dealing with the non-differentiable mapping from $z_{\mathbf{e}}$ to $z_{\mathbf{q}}$, the Straight-Through Estimator, known as Copying Gradients, comes into play. This estimation technique involves copying the gradient from the decoder's input back to the encoder's output ($\nabla(z_{\mathbf{e}}) = \nabla(z_{\mathbf{q}})$), allowing the encoder to receive an approximate gradient that would be obtained if the quantization step were differentiable. The encoder parameters are updated using the commitment loss (third term in equation (4.1)), where $z_{\mathbf{q}}$ is treated as a constant. The commitment loss forces the encoder to adhere to a specific embedding, by minimizing the distance between the encoder outputs and the selected embedding vectors. It also imposes penalties on the encoder outputs if they drift too far from their assigned embeddings, ensuring that minor fluctuations in $z_{\mathbf{e}}$ do not lead to changes in the quantized outputs $z_{\mathbf{q}}$.

Separately, the embedding vectors are updated using the VQ loss (the second term in (4.1). This loss is differentiable with respect to the embedding vectors. It encourages the discrete latent codes to be close to the embedding in the code-book and measures how well the latent codes match the nearest embedding vectors.

Inspired by Esser et al. [18] and Micheli et al. [19] where perceptual loss is effectively combined with commitment and reconstruction losses to train VQ-VAE, our model adopts a similar approach. We utilize a pretrained VGG-16 network, originally trained on the comprehensive Image-Net dataset as documented by Johnson et al. [20], to compute the perceptual loss. This network is crucial for its proven ability to extract and analyze high-level features from images, which is essential for evaluating perceptual similarity.
During the training of our VQ-VAE, both the target images (x) and the generated images ($\hat{x}$) by the VQ-VAE are fed into the unchanged (frozen) VGG-16 network, as illustrated in figure 4.4. This network serves as a fixed feature extractor, producing feature maps from its intermediate layers for each image pair. Perceptual loss is then calculated based on these feature maps, emphasizing differences in textures, shapes, and general patterns, which offers more detailed measure of similarity than simple pixel-by-pixel accuracy.

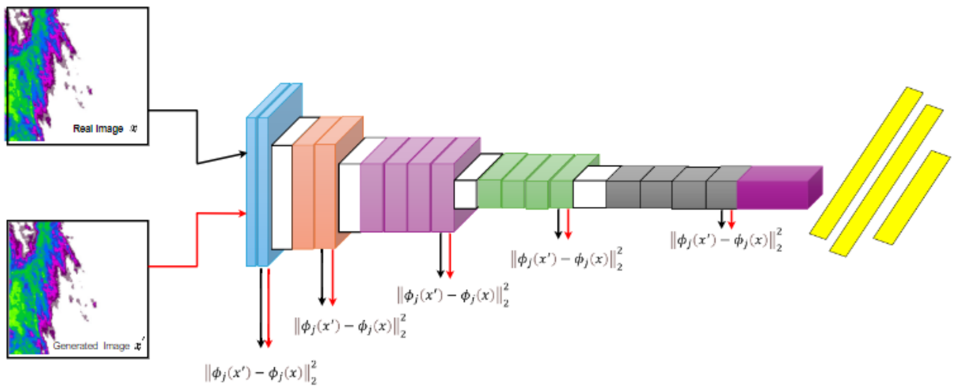

Real Image x
Generated Image x'
$\|\phi_j(x') - \phi_j(x)\|_2^2$
$\|\phi_j(x') - \phi_j(x)\|_2^2$
$\|\phi_j(x') - \phi_j(x)\|_2^2$
$\|\phi_j(x') - \phi_j(x)\|_2^2$
$\|\phi_j(x') - \phi_j(x)\|_2^2$

Figure 4.4: perceptual loss

The mathematical formulation of perceptual loss used in our training process is expressed as follows:

$$L_{\text{perceptual}}(\hat{x}, x) = \frac{1}{C_j H_j W_j} \|\phi_j(\hat{x}) - \phi_j(x)\|^2 \tag{4.2}$$

Here, $\hat{x}$ represents the output image; x denotes the target image; $\phi_j(\hat{x})$ and $\phi_j(x)$ are the feature maps from the j^{th} layer of the CNN of the output and target images, respectively. The terms C_j, H_j, and W_j correspond to the channel count, height, and width of the feature maps at the j^{th} layer, ensuring the normalization of the loss calculation.

During the training process, the weights of the generative model are updated to minimize both direct pixel differences and perceptual differences. This dual focus ensures that the final outputs accurately resemble the target images while also being visually appealing and realistic.

4.2. Auto Regressive Transformer

Introduced by Vaswani et al. [21], the transformer represents a significant shift in sequence prediction models, designed to address the limitations of its predecessors, such as LSTM and GRU. These traditional models often struggle with the vanishing gradient problem during training, a problem that becomes pronounced over long sequences. Additionally, their sequential processing nature limits their efficiency in capturing long-range dependencies Cho et al. [22]. These factors makes them less effective for tasks involving long sequences and computationally inefficient due to their inability to parallelize operations. Unlike LSTM and GRU, the Transformer model eliminates recurrent layers and relies entirely on self-attention mechanisms [21]. This architecture not only overcomes the challenges associated with long-range dependencies but also enhances computational efficiency and scalability by enabling parallel processing of sequence data.

The Transformer model leverages an attention mechanism that allows it to understand interactions between elements in an input sequence, irrespective of their positions. This provides a significant advantage over traditional models, which process inputs linearly. The model comprises two core parts: the encoder and the decoder. Each part plays a crucial role in handling the complexities of sequence prediction more effectively than its predecessors.

The key distinction between the encoder and decoder layers in the Transformer model lies in how they process inputs. In the encoder, the input sequence, combined with positional embeddings, first goes through a multi-head attention mechanism. Immediately following this, residual connections and then layer normalization are applied to enhance the flow and stability of information. This process is repeated after the position-wise feed-forward network, where another set of residual connections and layer normalization help prepare the output for subsequent layers, thus enabling the construction of deeper architectures with improved training stability.

Conversely, in the decoder, the inputs consist of embedded tokens generated during the decoding process, along with positional embeddings. These inputs are processed by masked attention components, which are crucial for ensuring that the attention mechanism only considers previously known or decoded tokens, thus preventing the decoder from accessing future information prematurely. This design is fundamental to maintaining the autoregressive property of the decoder, where each prediction is made based only on earlier tokens in the sequence. The output from the masked attention, when combined with the outputs from the encoder, feeds into the encoder-decoder attention mechanism, facilitating interaction between current decoder inputs and all encoder outputs. Subsequently, this integrated output passes through a feed-forward network. Similar to the encoder, both residual connections and layer normalization are applied after each component to promote stable training and enhance the learning process effectively.

The process concludes in the decoder's final layer, where the model generates the logits of the output sequence through a linear transformation of the decoder's outputs. These logits are subsequently processed by the softmax function, which converts them into probabilities. This final transformation aims to predict the next element in the sequence, effectively capturing the sequential nature and dependencies of the input data.

Inspired by Bi et al. [3], the Autoregressive Transformer (AR Transformer) is specifically adapted for weather nowcasting. This streamlined version of the original architecture eliminates the encoder and relies solely on the decoder. It primarily processes a sequence of conditioned tokens to predict subsequent tokens, ensuring each prediction is based only on previously known information.The AR Transformer's decoder features layers of causal self-attention, which prevent attending to future tokens, paired with a position-wise feed-forward network using GELU as the nonlinearity. Each component is followed by residual connections and layer normalization to promote stable training and effective learning. In the final layer of the decoder, the model outputs logits through a linear transformation, which are then converted into probabilities by the softmax function. The decision-making process in our project utilizes these probabilities, selecting the codebook token with the highest likelihood as the model's prediction.

4.2.1. Multi-head self Attention Mechanism

The attention mechanism is a fundamental component of the Transformer model, distinguishing it from traditional neural network architectures through its ability to capture

complex relationships and dependencies among input elements, irrespective of their positions within the sequence. This mechanism based on three critical components: queries (Q), keys (K), and values (V). It evaluates the similarity between 'query' and 'key' to assign weights to the 'value' components, thereby enabling the model to focus variably on different parts of the input sequence according to their relevance.
The core operation of the attention mechanism can be encapsulated by the formula:

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V \tag{4.3}$$

Here, d_k denotes the dimension of the query, and the division by $\sqrt{d_k}$ serves to scale the dot products, preventing the softmax function from entering regions with extremely small gradients. This mechanism, known as self-attention when Q, K, and V are derived from the same input as $Q = XW_Q$, $K = XW_K$ and $V = XW_V$, allows the model to weigh the importance of each part of the input sequence, addressing each element in relation to the rest. The attention is illustrated in 4.5.

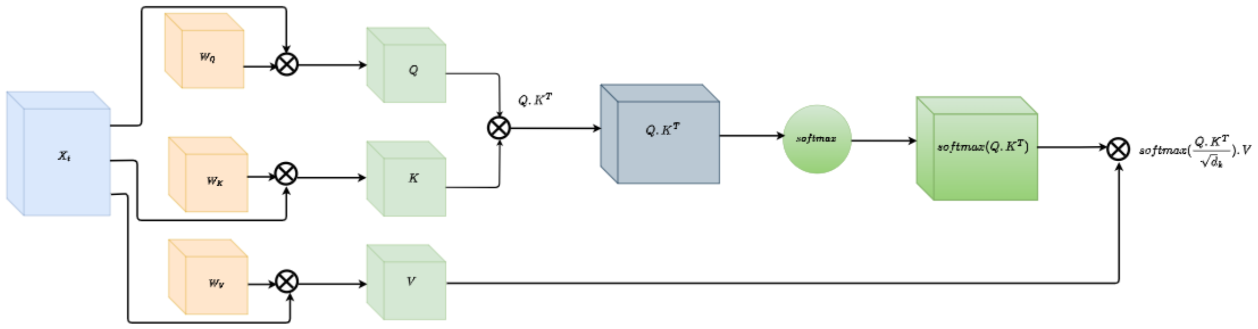
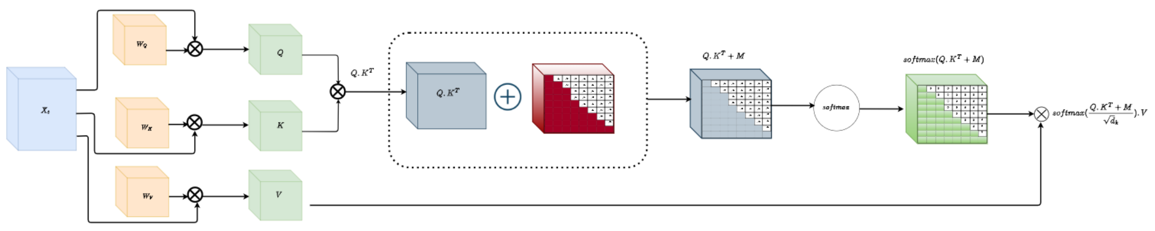


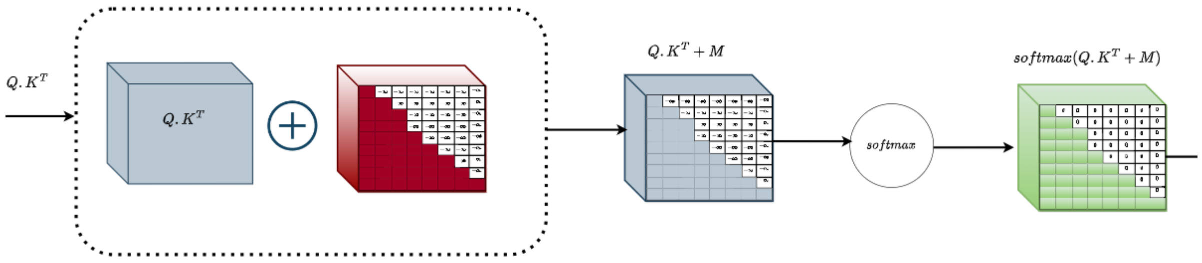
Figure 4.5: Illustration of the attention mechanism.

The causal attention mechanism employs masking to ensure predictions follow a logical sequence, thereby preventing the model from accessing future tokens prematurely. Specifically, within the matrix product QK^T, the entries that correspond to future tokens (non-causal entries) are assigned a value of $-\infty$. This manipulation is crucial when applying the softmax function, where the exponential of $-\infty$ effectively becomes zero. Consequently, these entries contribute nothing to the output of the $softmax$, ensuring no weight is allocated to future positions as shown in figure 4.6. This strategic approach directs the model's attention exclusively to previously seen or current tokens when predicting subsequent elements in the sequence.

The multi-head attention mechanism significantly enhances the Transformer's capability by distributing the attention process across multiple 'heads', each assigned to



(a)



(b)

Figure 4.6: Illustration of the attention mechanism.

analyze different segments of the input sequence. This division facilitates a thorough comprehension of both the positional and contextual nuances within the data, as each head concentrates on distinct aspects of the sequence. By assessing various aspects of the sequence in parallel, each attention head provides a unique perspective on the input's context, enriching the overall representation of the sequence's meaning. The outputs from these heads are subsequently merged and processed through a weight matrix W^O, ensuring consistency with the model's dimensional requirements.
The MultiHead attention operation is succinctly represented as

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{4.4}$$

where each head performs attention on uniquely transformed queries, keys, and values. This approach empowers the Transformer to identify and synthesize a diverse range of data interactions, significantly enhancing its proficiency in managing complex sequence-based challenges.

4.2.2. Training and Generation

Each image is encoded using the VQVAE and represented by a set of discrete tokens denoted as $z_q = q(E(x))$ which belongs to the set $0, \ldots, |Z| - 1$. In the Autoregressive Transformer (AR Transformer), these tokens are transformed into a continuous vector through an embedding process. This is further enhanced by combining them with positional embeddings to integrate the sequence order into the vectors. The transformer then processes these embedded, positional-encoded tokens through all its layers, ultimately generating raw, unnormalized predictions for each token, known as logits.

The logits generated by the transformer are normalized and transformed into probabilities using a softmax layer. The associated loss function, represented by equation 4.5 calculates the conditional probabilities of a subsequent token z_i given all previous tokens $z_{<i}$, formulated as $p(z) = \prod_{i=1}^{N} p(z_i|z_{<i})$. This model framework depicted in figure 4.7 enables the Transformer to effectively predict the likelihood of future indices based on accumulated prior information. The cross-entropy loss guide the back-propagation process, enhancing parameter optimization to improve both predictive accuracy and learning efficiency of the model in capturing data distributions.

$$\mathcal{L}_{\text{Transformer}} = \mathbb{E}_{x\sim p(x)}[-\log p(z)] \tag{4.5}$$

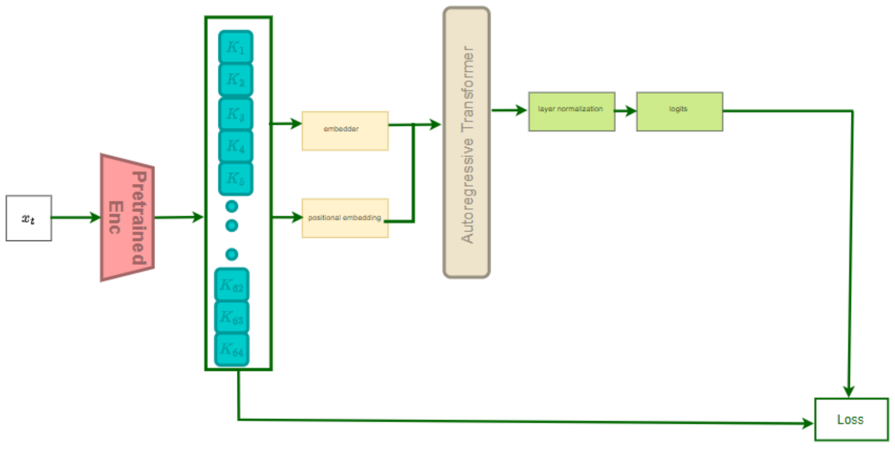
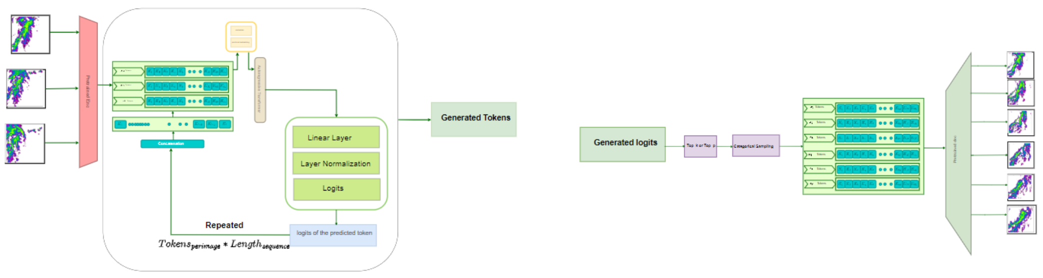


Figure 4.7: The cross entropy loss calculated between the actual distribution of the tokens and the logits.

In the generation phase of an Autoregressive Transformer (AR Transformer) model, the process starts by inputting a conditioned frame as a sequence of tokens , as depicted in figure 4.8a. The model iteratively predicts the logits for the next token and then slices and concatenates these logits with the preceding tokens. This updated sequence serves as the input for the subsequent prediction cycle. This iterative process continues for a number of steps, determined by the product of the number of prediction frames and the total number of tokens required to decode a single radar image.

Once the logits for all prediction frames have been generated, the selection process employs advanced techniques such as top-k and top-p sampling to refine the choices. The final output consists of a series of tokens that are sampled from the predicted probability distribution through categorical sampling. The output tokens are then passed to the decoder side of the Vector VQ-VAE, where the discrete tokens are converted back into continuous vectors retrieved from a pretrained codebook. Subsequently, the predicted maps are generated based on these vectors. The afermentioned process is illustrated in fig: 4.8.



(a) Generation phase of the AR transformer.

(b) Generated logits and their output

Figure 4.8: An example of the generation phase of the AR transformer where 6 output images are generated based on 3 input images.

To expedite the generation process, the model utilizes a Key-Value (K-V) cache that stores the keys and values computed by the attention mechanism, enabling their reuse in subsequent prediction steps. This is particularly advantageous in the AR Transformer, where outputs are generated sequentially, one token at a time. By leveraging the precalculated K-V cache from the previous step, the decoder avoids recomputing the keys and values for the entire sequence during each step. This approach improves efficiency and reduces computational overhead.

4.3. Model Configuration Details

Due to memory constraints encountered during the training of the VQVAE, radar images with a spatial resolution of 256×256 have been downsampled to 128×128. This downsampling preserves the semantic integrity of the images, enabling efficient VQ-VAE training to achieve a reduced latent space representation without overloading the Graphics Processing Unit (GPU). Figure 4.9 displays examples of these downsampled images, illustrating how they maintain their essential semantic content.

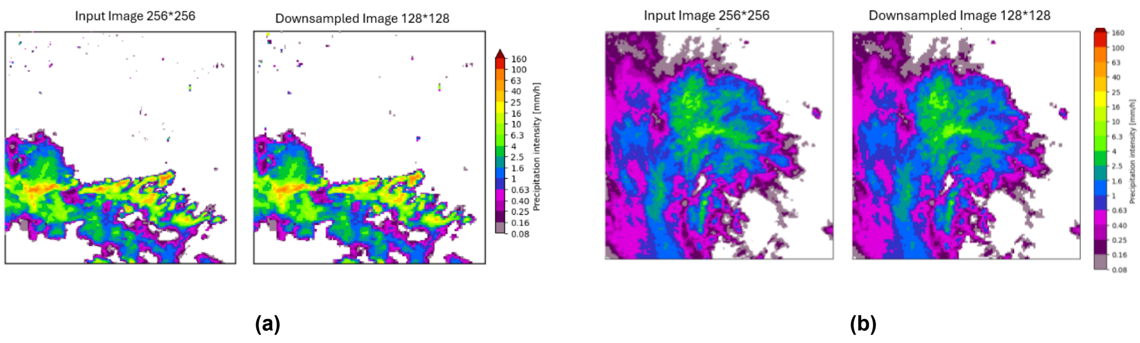


Figure 4.9: Example of downsampled radar images.

The encoder processes downsampled radar images through four successive downsampling layers, each composed of two ResNet blocks. An attention layer follows the final downsampling layer, leading into an intermediate block that includes two additional ResNet blocks with another attention layer placed between them. This architecture (illustrated in figure: 4.10) systematically reduces the image resolution from $128X128$ down to $8X8$ across stages($128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 8$). Concurrently, as the spatial dimensions decrease, the number of channels increases from 1 to 1024. This increase aligns with the embedding dimension ($n_z = 1024$) of the codebook, enhancing the quantization accuracy by facilitating the selection to the nearest token based on distance.

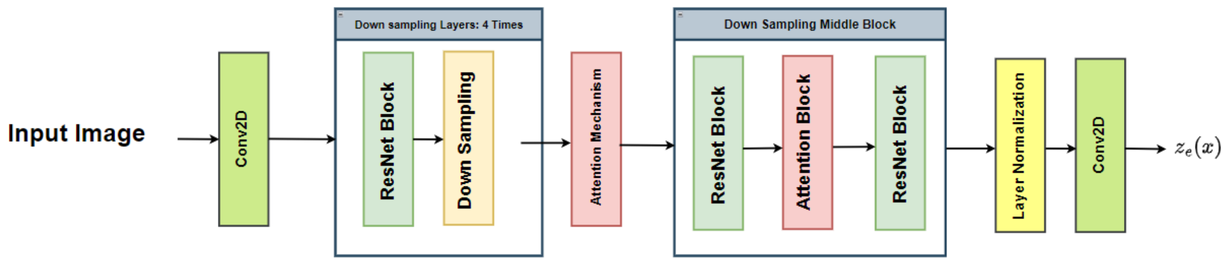


Figure 4.10: The encoder configuration used in Nowcasting GPT.

The codebook consists of 1024 unique codes, enabling each $128x128$ image to be represented by 64 tokens. Each token, a number ranging between 0 and 1023, corresponds to an embedding dimension of 1024.

The quantization process produces 64 tokens that are feed into the transformer architecture, as depicted in figure 4.11. These tokens are converted into continuous vectors through word embedding and then augmented with sequence context via positional embeddings. To prevent overfitting, a dropout layer is applied. These prepared vectors then proceed to the core of the transformer, which consists of N uniform blocks. Each block begins with layer normalization, followed by multi-head attention with a residual connection. After the attention phase, another layer normalization occurs, leading into a multi-layer perceptron (MLP) which introduces nonlinearity, followed by another residual connection. As the vectors progress through the N blocks, they undergo further normalization and ultimately produce logits that represents the prediction values. The optimal architecture configurations are determined through a grid search, explained in detail in the subsequent chapter, 6.

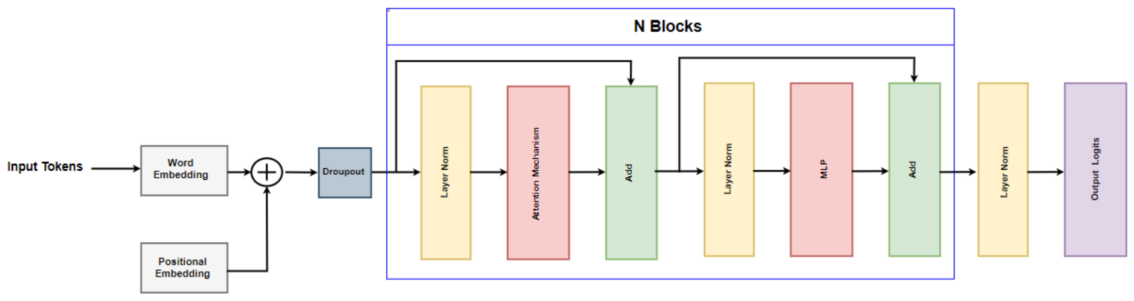


Figure 4.11: The Transformer architecture used in Nowcasting GPT.

The decoder figure 4.12 mirrors the encoder's structure. It begins with two ResNet blocks separated by an attention block, followed by four upsampling layers. This configuration facilitates the reconstruction of the image from its compact latent format, effectively preserving the essential details of the original image.

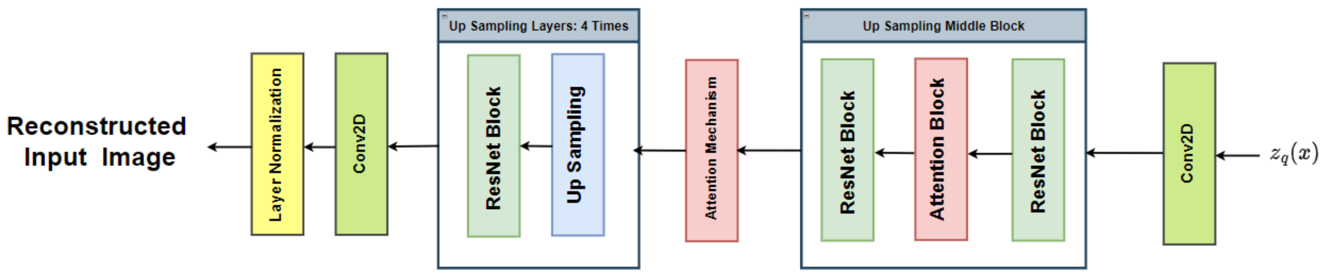


Figure 4.12: The decoder configuration used in Nowcasting GPT.

4.4. Conclusion

This chapter has laid out the methodologies that underpin our approach to sophisticated meteorological data analysis using machine learning. By integrating the VQ-VAE and the AR Transformer, we have established a system capable of efficiently handling large datasets and predicting weather conditions with improved accuracy. The next chapter will detail the implementation of these methods, including the optimization of model parameters through grid search and the evaluation metrics used to assess the performance of the Nowcasting GPT model. This will provide a deeper understanding of the model's effectiveness and identify opportunities for further refinements.

5 Verification

To evaluate the accuracy of weather predictions generated by the Vector Quantized Variational Autoencoder (VQVAE) and the Autoregressive (AR) Transformer models, this study utilizes a comprehensive set of evaluation metrics. The geographical focus of this evaluation is the Netherlands, with a particular emphasis on the entire region and its catchment areas. This analysis offer a robust test for assessing the predictive capabilities of the models under study. The evaluation metrics are organized into three main categories to provide a holistic analysis of the model's performance: Continuous metrics for quantitative accuracy, Categorical metrics for classification precision, and Spatial metrics for geographical accuracy. This structured approach ensures a detailed examination of the models' ability to forecast weather patterns with varying degrees of complexity and across different spatial scales.

5.1. Continuous metrics

Pearson's Correlation Coefficient (PCC):

Pearson's Correlation Coefficient (PCC) is employed to quantify the linear correlation between the predicted and actual observed radar maps, which are represented as images of 256*256 pixels. This pixel-level analysis enables a detailed examination of the model's accuracy in replicating actual precipitation patterns. The PCC, denoted as ρ, is calculated using the following equation:

$$\rho = \frac{1}{\text{row} * \text{col}} \sum_{i=1}^{\text{row}} \sum_{j=1}^{\text{col}} \frac{(F[i][j] - \mu_F)(O[i][j] - \mu_O)}{\sigma_F \sigma_O} \tag{5.1}$$

Here, $F[i][j]$ and $O[i][j]$ represent the predicted and actual precipitation values at the

i^{th} row and j^{th} column of the radar map, respectively. Given that the radar map dimensions are 256x256 pixels, 'row' and 'col' refer to these dimensions. μ_F and μ_O denote the mean values of the predicted and actual maps, respectively. σ_F and σ_O represent their respective standard deviations, which are calculated across all pixels to capture the variability within each map.

For a prediction to be deemed skillful, the PCC value should meet or exceed the threshold of $1/e$, where e is the base of the natural logarithm (approximately 2.718). This criterion ensures that the correlation between predicted and actual observations is sufficiently strong to be considered meaningful and useful. A PCC value below this threshold indicates that the forecast lacks skill and may not provide reliable insights.

To comprehensively assess the predictive skill of our model, PCC will be calculated at multiple lead times: 30, 60, 90, 120, 150, and 180 minutes. This approach enables us to evaluate the model's performance and its ability to predict over varying future intervals, providing a detailed understanding of its effectiveness across different timescales.

Mean Square Error and Mean Absolute Error:
To quantify the average discrepancy between predicted and actual precipitation values, two metrics are utilized: the Mean Square Error (MSE) and the Mean Absolute Error (MAE). The MAE provides a straightforward measure of the average magnitude of errors in the predictions, indicating how close the model's predictions generally are to the ground truth across the dataset. A low MAE signifies that the model delivers reliable results, reflecting its accuracy in forecasting precipitation amounts.

On the other hand, the MSE is particularly sensitive to larger errors in the prediction set, as it squares the differences before averaging them. Therefore, a low MSE not only suggests that the model's predicted intensities are, on average, close to the actual observations but also indicates that the model is less likely to produce significant errors in its predictions.

$$\text{MSE} = \frac{1}{\text{row} \times \text{col}} \sum_{i=1}^{\text{row}} \sum_{j=1}^{\text{col}} (F[i][j] - O[i][j])^2 \tag{5.2}$$

$$\text{MAE} = \frac{1}{\text{row} \times \text{col}} \sum_{i=1}^{\text{row}} \sum_{j=1}^{\text{col}} |F[i][j] - O[i][j]| \tag{5.3}$$

5.2. Categorical metrics

Critical Success Index (CSI) and the False Alarm Ratio (FAR):

In this thesis, the categorical metrics employed to evaluate the model's performance include the Critical Success Index (CSI) and the False Alarm Ratio (FAR). These metrics are computed based on a predefined precipitation threshold, where each pixel $[i, j]$ in both the actual and predicted precipitation maps is classified as '1' (indicating precipitation levels at or above the threshold) or '0' (indicating levels below the threshold).

The CSI measures the ratio of correctly predicted precipitation events relative to the total number of actual events, missed events, and false alarms. This index provides insight into the model's effectiveness in accurately predicting significant precipitation events. The CSI is evaluated for precipitation levels of 1mm, 2mm, and 8mm, as shown in Eq. 5.4, to determine the model's accuracy across various precipitation intensities.

$$\text{CSI} = \frac{TP}{TP + FP + FN} \tag{5.4}$$

where TP (True Positives) indicates instances where the model correctly predicts precipitation events that occur, FP (False Positives) refers to incorrect predictions of precipitation, and FN (False Negatives) refers to instances when the model fails to predict precipitation that does occur.

Conversely, the FAR calculates the ratio of false positive predictions FP to the total number of predicted positive precipitation events.This metric, detailed in Eq. 5.5, is assessed for the same precipitation levels as the CSI. A lower FAR indicates higher precision in the model's precipitation forecasts, characterized by fewer false alarms.

$$\text{FAR} = \frac{FP}{FP + TP} \tag{5.5}$$

Both CSI and FAR are crucial for evaluating the model's ability to categorize precipitation intensity accurately. These metrics provide valuable insights into the model's reliability and precision in forecasting significant weather events based on specified precipitation thresholds.

5.3. Spatial metric

Fractional Skill Score:

The Fractional Skill Score (FSS) is utilized to evaluate the spatial accuracy of our model. This metric measures the model's performance relative to specific precipitation thresholds and selected spatial dimensions. For any given rainfall threshold, pixels within the radar map are categorized as '1' if the precipitation level exceeds the threshold, and '0' otherwise. A window of size $n \times n$ is then applied across the radar map to compute the average of pixels that meet or exceed the threshold, yielding new Observed (O_n) and Forecasted (F_n) matrices. The FSS is computed using Equation 5.6, while the reference Mean Square Error (MSE_{ref}) is calculated by Equation 5.7.

$$\text{FSS} = 1 - \frac{\text{MSE}(n)}{\text{MSE}_{\text{ref}}(n)} \tag{5.6}$$

$$\text{MSEref}(n) = \frac{1}{\text{row} \times \text{col}} \left(\sum_{i=1}^{\text{row}} \sum_{j=1}^{\text{col}} O_n[i][j]^2 + \sum_{i=1}^{\text{row}} \sum_{j=1}^{\text{col}} F_n[i][j]^2 \right) \tag{5.7}$$

To assess the model's performance, the Fractional Skill Score (FSS) is calculated at spatial scales of 1km, 10km, 20km, and 30km. A model at a specific lead time t is deemed skillful if its forecast accuracy surpasses that of a random forecast, defined as $0.5 + \frac{f_0}{2}$, where f_0 represents the fraction of the observed area relative to the total area of the radar map.

6
Results

In enhancing the prediction accuracy of a deep generative model for rainfall prediction, the synergy between the Vector Quantized Variational Autoencoder (VQVAE) and the Auto-regressive (AR) Transformer is pivotal. The VQVAE efficiently compresses radar imagery into a manageable number of discrete tokens, capturing essential spatial information relevant to rainfall patterns. This compression is not just a matter of efficiency; it ensures that the subsequent AR Transformer can focus on the most relevant features for rainfall prediction. The AR Transformer's core strength lies in its ability to capture temporal dependencies among the compressed tokens. By analyzing the sequence of these tokens, it can make more accurate predictions about future rainfall events. This capability is crucial because rainfall is often influenced by weather patterns that develop over long periods. The AR Transformer is efficient at handling these long range dependencies within sequences, enabling it to identify and incorporate patterns that significantly affect rainfall into its predictions. Moreover, the auto-regressive nature of the AR Transformer allows it to generate forecasts based on past observations. This feature means that as new data becomes available, the transformer can refine its predictions adaptably, improving accuracy over time. By leveraging both the spatial compression of the VQVAE and the temporal modelling capability of the AR Transformer, the deep generative model represents a sophisticated approach to predicting rainfall, capable of understanding and anticipating the complex dynamics of weather pattern.

6.1. Comprehensive Analysis of Optimal Transformer Configuration

To further refine the prediction accuracy of the AR Transformer, optimizing its configuration is crucial. The balance between the number of attention heads, the depth of layers, and the embedding dimension is crucial for addressing the complex spatial and temporal dependencies inherent in rainfall data. A rigorous hyper-parameter tuning process is undertaken, employing a comprehensive grid search across various combinations of attention heads (4, 8, 16), layer depths (8, 12, 16, 20, 24), and embedding dimensions (64, 128, 256, 512). This optimization aims to ascertain the most effective setup, ensuring the transformer's capability to accurately capture and forecast the nuanced dynamics of rainfall patterns, thereby significantly enhancing its forecasting accuracy. The selection of the optimal model configuration is determined through an exhaustive analysis of performance metrics, aiming to strike an optimal balance between model complexity and predictive accuracy. The evaluation framework encompasses a variety of metrics, including the Pearson Correlation Coefficient (PCC), Mean Absolute Error (MAE), and Critical Success Index (CSI) at rainfall thresholds of 1mm, 2mm, and 8mm. Additionally, the False Alarm Rate (FAR) at these thresholds, along with the Fractions Skill Score (FSS) at spatial scales of 1km, 10km, 20km, and 30km, are assessed. These metrics collectively provide a comprehensive view of the model's performance, focusing on accuracy, reliability, and spatial precision, thereby guiding the selection of the most effective model configuration.

6.1.1. Configuration Analysis: 4 Heads

To check the model performance along different configurations, our comparative analysis is initialized by a transformer model with 4 heads and varying embedding dimensions. For each embedding dimension, we tested different number of layers.

6.1.1.1. Embedding Dimension Analysis: 8 Heads with 64

Initially, we fixed the number of heads (4 heads) and the embedding dimension (64). We tested 5 different number of layers: 8, 12, 16, 20, and 24 (shown in 6.1). The 8-layer model emerges as the most balanced, excelling in PCC, CSI across all thresholds, and FSS across spatial scales, indicating its superior ability in capturing correlation, detection, and spatial accuracy. However, it shows potential for improvement in MAE and FAR at the 2mm and 8mm threshold, pointing to a need for refinement in predicting moderate rainfall more accurately.

Further investigation into configurations with increased layer counts, specifically 16 and 20 layers, yields mixed results. The 16-layer model demonstrates a notable capability in reducing average errors (MAE) and improving FAR performance at the 2mm threshold, highlighting its effectiveness in quantitative predictions and in minimizing false alarms for light to moderate rainfall. However, it exhibits challenges in consistently achieving high scores in PCC and CSI, pointing to variability in trend consistency and event detection. Conversely, the 20-layer model shows promise in FAR at the 8mm threshold and matches the 8-layer model's FSS up to 20km, yet it does not consistently excel across all metrics, particularly in PCC, CSI at lower thresholds, and FSS at the 30km scale. This observation suggests that the addition of layers does not necessarily correlate with universally enhanced model performance, emphasizing the overall robustness and effectiveness of the 8-layer con-figuration in capturing the nuances of rainfall patterns, albeit with room for minor improvements.

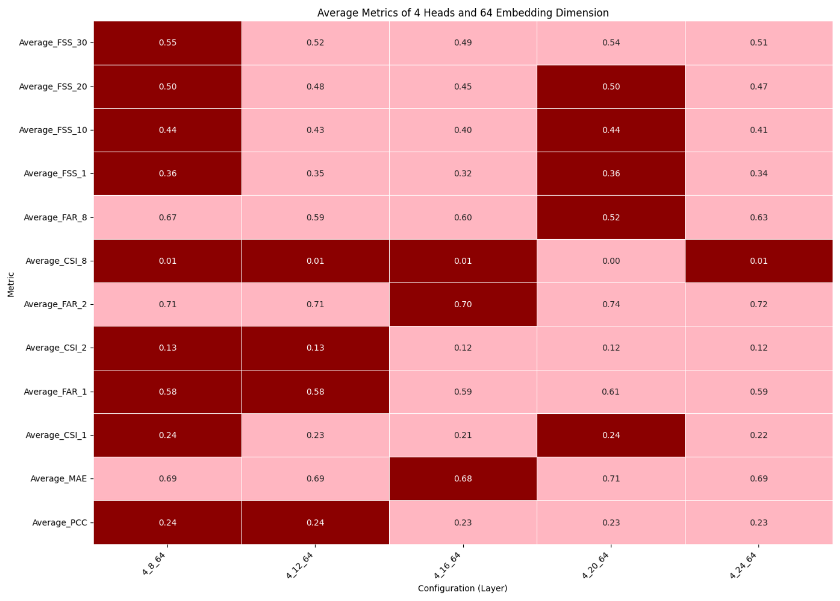


Figure 6.1: Performance of different layers with 4 heads and 64 embedding dimensions.

6.1.1.2. Embedding Dimension Analysis: 4 Heads with 128

As the analysis progresses to models with 128 embedding dimensions, maintaining the 4-head architecture, it uncovers that the 16-layer configuration is particularly adept at spatial pattern recognition and predicting significant rainfall events (6.2). This is shown by its high FSS and CSI scores. However, this configuration does not achieve optimal performance in PCC and MAE, suggesting a compromise in accurately predicting exact rainfall quantities and trend correlation. In comparison, the 8-layer model outperforms in PCC, FAR (at the 1mm and 2mm thresholds) and MAE. This affirms that the 8 layers model is superior in quantitative predictions and trend tracking, especially for light rainfall events. Despite this, it falls short in CSI, presenting a discernible trade-off between CSI and FAR.

The selection of the optimal layer depth ultimately depends on the application needs in balancing quantitative accuracy and spatial forecasting capabilities. For tasks that prioritize spatial detail and event detection, the 16-layer model (head:4, layer:16, embedding dimension:128) stands out, whereas the 8-layer model (head:4, layer:8, embedding dimension:128) offers a well rounded solution suitable for general forecasting, with an advantage in accuracy and reduced false alarm rates for lighter rains. Consequently, both the 8 and 16 layers are considered in the final optimal model analysis for this embedding dimension.

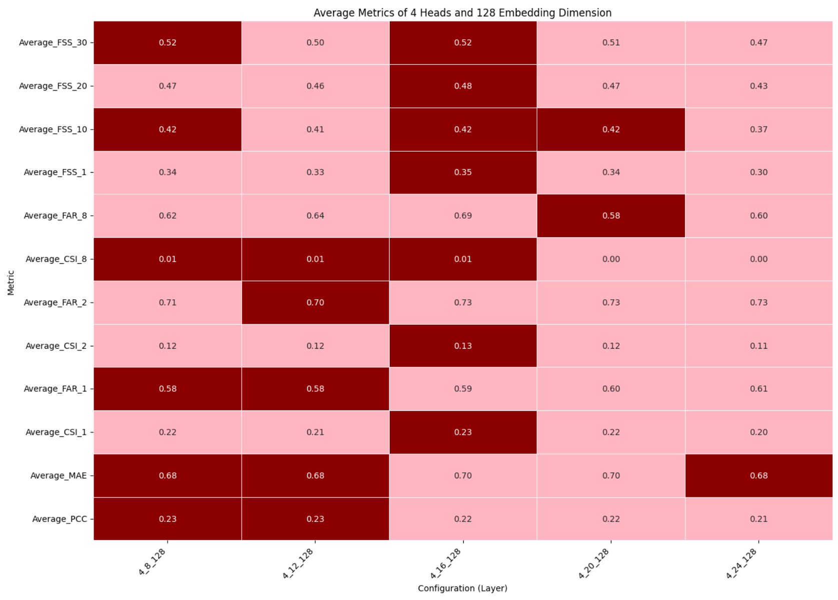


Figure 6.2: Performance of different layers with 4 heads and 128 embedding dimensions.

6.1.1.3. Embedding Dimension Analysis: 4 Heads with 256

We explored a 256 embedding dimension model (heads:4, embedding dimensions:256) along different layers. Among the tested layers (6.3), the 24 layers configuration stands as a superior model in overall correlation with observed rainfall (PCC), accuracy in predicting significant rainfall events at various thresholds (CSI), and accurately capturing the spatial distribution of rainfall (FSS). This configuration is identified as the leading choice within its embedding class for applications demanding high precision in both spatial and intensity aspects of rainfall prediction. In contrast, the 20 layers configuration matches the 24-layer model in PCC, indicating a strong linear relationship with observed data, and shows improvements in MAE and reduced FAR. This suggests a potential trade-off between precision and general predictive reliability. Nonetheless, due to the 24-layer model's exceptional precision in PCC, CSI, and FSS, its capability in accurately determining the timing, occurrence, and spatial distribution of rainfall guarantees its inclusion in the conclusive analysis for the optimal model configuration.

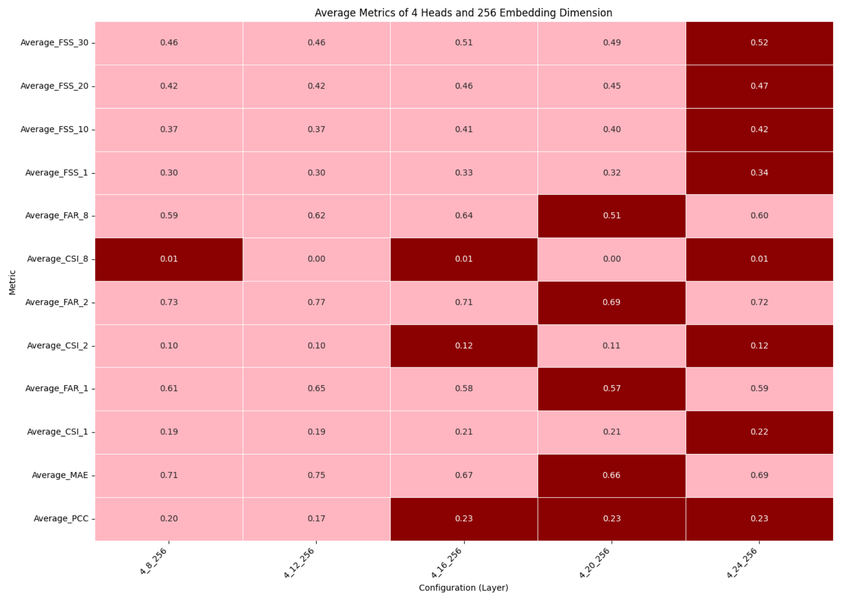


Figure 6.3: Performance of different layers with 4 heads and 256 embedding dimensions.

6.1.1.4. Embedding Dimension Analysis: 4 Heads with 512

We tested the model with fixed 4 heads and fixed 512 embedding dimensions along 5 different layers. The evaluation loss (6.4) shows an increasing trend which indicates that even with a small number of layers, this configuration is over fitting. This indicates that the model is too complex for the data at hand. Thus, no further analysis is needed on this embedding dimension.

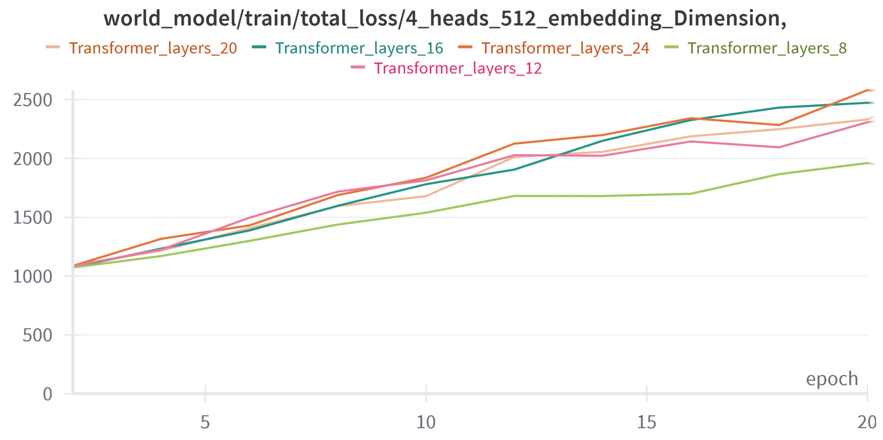


Figure 6.4: Evaluation loss of 4 heads and 512 embedding dimensions with different layers.

6.1.1.5. Selection of the optimal model at 4 heads

After analyzing the performance of the model with 4 heads across varying numbers of layers and embedding dimensions, we identified the four best-performing models for each embedding dimension (64, 128, 256). These models are 4-8-64, 4-8-128, 4-16-128, and 4-24-256 (6.5). The next step is to compare how these models perform using different metrics to identify the optimal model configuration with 4 heads.

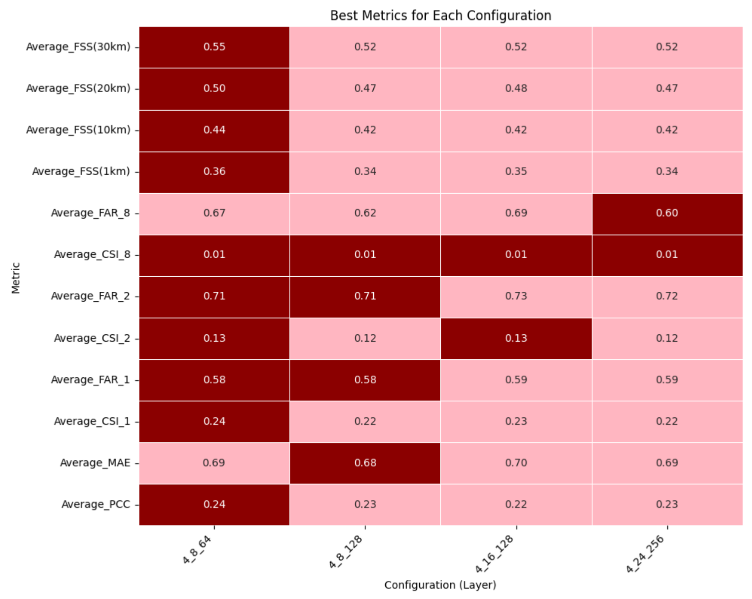


Figure 6.5: Performance of different layers with best selected models at 4 heads.

Upon comparing the models, the heat-map analysis reveals that models with lower complexity generally excel across all evaluated metrics, ranging from the correlation between predicted and real observations to prediction accuracy and spatial skill scores. Specifically, the model with a configuration of 4-8-64 stands out for its performance. It achieves the lowest False Alarm Ratio (FAR) at moderate and low thresholds, although its FAR increases at higher thresholds, particularly in comparison to the 4-24-256 configuration. Despite this, the latter configuration results in distortion across other metrics, underscoring a trade-off.
Interestingly, the model with a 128 embedding dimension and 8 layers showcases the lowest mean absolute error (MAE), indicating a marginal improvement. However, this comes at the expense of Critical Success Index (CSI) and Fraction Skill Score (FSS) performance, high-lighting a distortion in these metrics.
The balanced performance of the 4-8-64 model, combined with its computational efficiency, solidifies its position as the optimal choice among all transformers with 4 heads. This model's ability to deliver high accuracy and reliability with a streamlined set of parameters makes it a particularly appealing choice in operational environments where computational resources are constrained. This efficiency and effectiveness underscore the significance of selecting a model that not only meets accuracy and reliability benchmarks but does so in a resource-efficient manner.

6.1.2. Configuration Analysis: 8 Heads

Exploring the potential of an 8-head configuration represents a logical progression in our quest for the most effective and efficient model for rainfall prediction. Guided by careful consideration of the performance trade-offs and computational implications of varying the number of heads, our analysis dives into the nuanced interplay between model complexity and predictive accuracy across different embedding dimensions.

6.1.2.1. Embedding Dimension Analysis: 8 Heads with 64

Initially, we fixed the number of heads (8 heads) and the embedding dimension (64). We tested 5 different number of layers: 8,12,16,20, and 24 (6.6). The 12-layer model (heads:8, layers:12, embedding:64) consistently outperforms others across all performance metrics, except for the FAR at the 8mm threshold. A deeper exploration into configurations with more layers reveals that the 20-layer model (heads:8, layers:20, embedding:64)) offers a lower FAR at this threshold, at the expense of diminished performance in other critical areas such as correlation and event detection. The comparison between the models emphasizes the 12-layer model's suitability for a broad spectrum of applications, especially those necessitating reliable estimations of both the quantity and location of rainfall events. Its equilibrium between trend accuracy, quantitative precision, and spatial detail makes it a comprehensive choice for operational deployment.

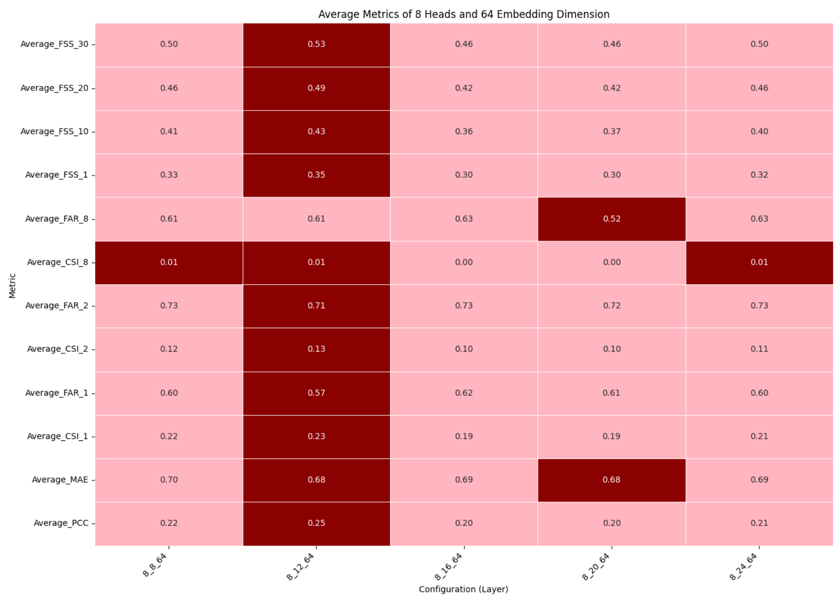


Figure 6.6: Performance of different layers with 8 heads and 64 embedding dimensions.

6.1.2.2. Embedding Dimension Analysis: 8 Heads with 128

Progressing to an embedding dimension of 128 while retaining the 8-head architecture (6.7), we find the 16-layer configuration (heads:8, layers:16, embedding:128) to be supremely advantageous. This model configuration not only establishes a strong correlation with observed rainfall data, but also excels in accurate event detection at lower thresholds. It effectively pinpoint significant rainfall events with a reduced False Alarm Rate (FAR), except in comparison to the 8-layer model at the 8mm threshold, where the heads:8, layers:8, embedding:128 model demonstrates superior FAR. These findings highlight the 16-layer model's precision and efficiency, making it a pivotal consideration in our determination of the optimal model configuration.

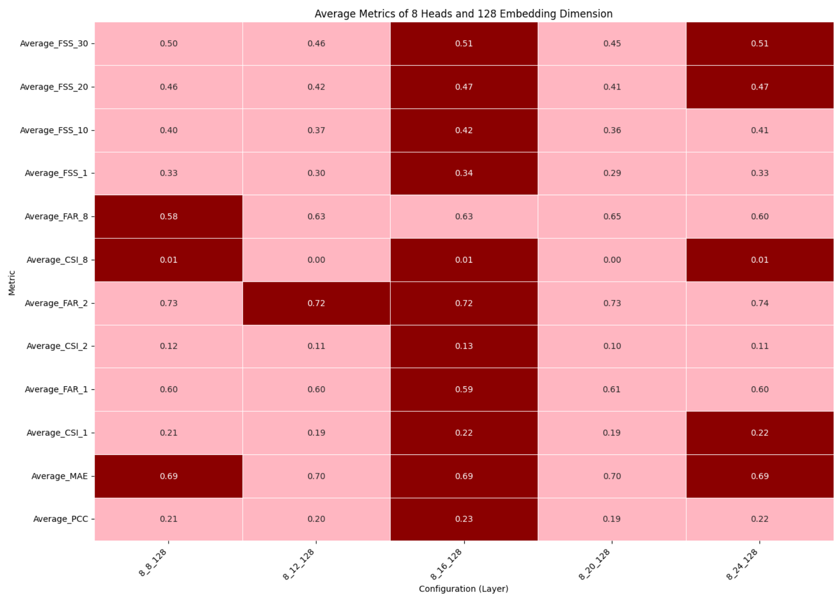


Figure 6.7: Performance of different layers with 8 heads and 128 embedding dimensions.

6.1.2.3. Embedding Dimension Analysis: 8 Heads with 256

Expanding the embedding dimension to 256 while maintaining 8 heads (6.8), the 24 layer model (heads:8, layers:24, embedding:256) ascends as the outstanding performer. This model was able to achieve the highest PCC and CSI across all thresholds. This configuration showcases unequal ability in detecting significant rainfall events. Moreover, its FSS across a range of spatial scales affirms its adeptness in accurately predicting rainfall's spatial distribution. However, when refining MAE and minimizing FAR, the 20-layer model (heads:8, layers:20, embedding:256) presents an improvement, with a notable trade-off in detection capability. This complexity suggests the (heads:8, layers:24, embedding:256) model as the prime candidate for the 256 embedding dimension, pending a comparative analysis with other distinguished models from varying embedding dimensions to establish the universally optimal configuration.

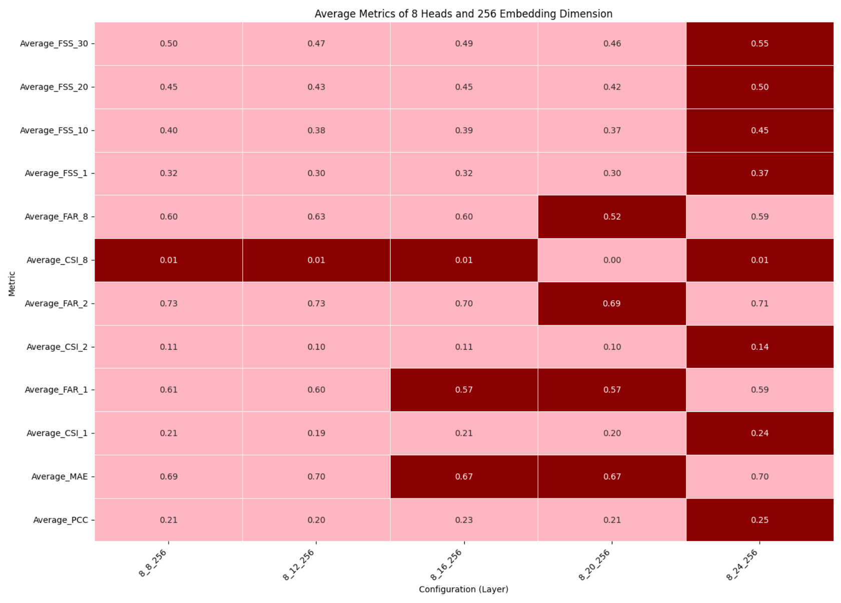


Figure 6.8: Performance of different layers with 8 heads and 256 embedding dimensions.

6.1.2.4. Embedding Dimension Analysis: 8 Heads with 512

Since the model was too complex with 4 heads as shown before (section), increasing the number of heads to 8 will result in a more complex model. Hence, the model will over-fit. Therefore, these configurations (heads:8, layers: 8,12,16,20,24, embedding: 512) are excluded from the analysis.

6.1.2.5. Selection of the optimal model at 8 heads

In the pursuit of optimizing a neural network architecture with 8 heads for the task of predictive modelling, an empirical investigation was conducted to evaluate model performance across various combinations of layers and embedding dimensions. This rigorous analysis led to the identification of three candidate models: 8-12-64, 8-16-128, and 8-24-256, each distinguished by its embedding dimension. The forthcoming comparative study aims to choose the optimal model configuration when employing 8 heads, by systematically assessing these models against a spectrum of performance indicators.

The comparative analysis of the models reveals that the configuration comprising 24 layers and an embedding dimension of 256 (8-24-256) emerges as the superior choice. This model configuration excels in several critical performance metrics, including PCC, CSI across all rainfall thresholds, and FSS across different spatial scales. These outcomes indicate a robust capability in accurately predicting rainfall events and effectively capturing their spatial distribution.

Despite the lower MAE and FAR at lower thresholds, which is achieved by the model with the lowest complexity (heads: 8, layers:12, embedding:64), this advantage comes at the cost of a decrease in CSI at lower and medium rainfall thresholds. Addition-

ally, it exhibits a higher FAR at high intensity thresholds and experiences distortion in spatial scores across all scales when compared to the (heads:8, layers: 24, embedding:256)configuration. This trade-off highlights the limitations of the simpler model in maintaining accuracy and reliability across a broader range of conditions.
The medium complexity model (heads:8, layers: 24, embedding:256)stands out as the optimal choice when prioritizing a configuration with 8 heads. This conclusion is drawn from its balanced performance, demonstrating both high accuracy in rainfall event prediction and efficient spatial distribution mapping.

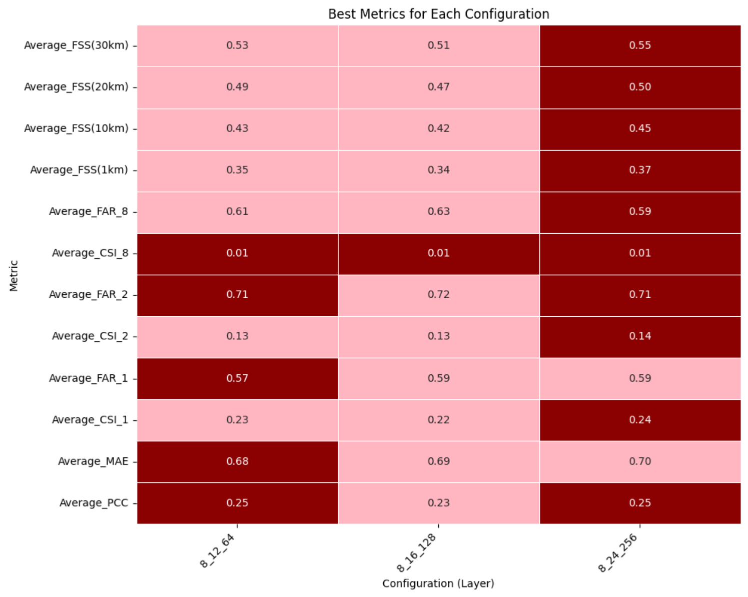


Figure 6.9: Performance of different layers with best selected models at 8 heads.

6.1.3. Configuration Analysis: 16 Heads

Exploring the potential of a 16-head configuration represents a logical progression in our quest for the most effective and efficient model for rainfall prediction. Guided by careful consideration of the performance trade-offs and computational implications of varying the number of heads, our analysis dives into the nuanced interplay between model complexity and predictive accuracy across different embedding dimensions.

6.1.3.1. Embedding Dimension Analysis: 16 Heads with 64

In our quest to refine a transformer model for rainfall prediction, equipped with 16 heads and a 64-dimensional embedding, our comprehensive grid search has identified that a configuration with 12 layers stands superior across most of the performance metrics (6.10). This setup notably excels in PCC and MAE, showcasing a model that correctly mirrors observed data trends and forecasts rainfall amounts with minimized average errors. Additionally, this configuration shows an improvement in CSI at all evaluated thresholds, and FSS across various spatial scale. This model underscores its efficiency in event detection and precise mapping of rainfall's spatial distribution. However, it is crucial to acknowledge that while the 12-layer model exhibits proficiency in many aspects, transitioning to a 24-layer configuration yields a slight improvement in the FAR at the higher rainfall thresholds of 2mm and 8mm. This suggests that the integration of additional layers may contribute to reducing the over-prediction of significant rainfall events. Nevertheless, this marginal benefit is counterbalanced by a decline in other essential metrics, indicating that increasing the complexity of the models does not consistently translate to enhanced model performance. In particular, the 24-layer model demonstrates a decrease in the accuracy of rainfall event detection, spatial distribution precision, and overall correlation with observed data, as reflected by diminished scores in PCC, CSI, and FSS. The 12-layers model, which provides a well balanced and high performing solution, emerges as an important configuration for subsequent comparison to ascertain the most effective model.

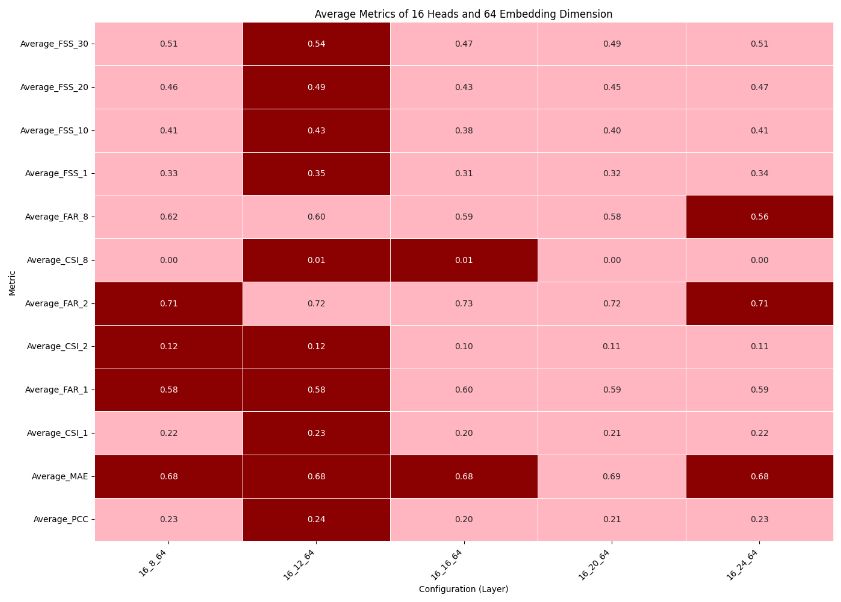


Figure 6.10: Performance of different layers with 16 heads and 64 embedding dimensions.

6.1.3.2. Embedding Dimension Analysis: 16 Heads with 128

Increasing the embedding dimension to 128 within the same 16-head transformer model framework (6.11), the 12-layers setup continues to appear as the best configuration . It upholds superior capability in aligning actual and predicted observations and excels in the CSI across different thresholds. Yet, an increment to 24 layers leads to a reduction in MAE and FAR, unveiling a substantial trade-off between CSI and FAR. Consequently, despite the reduced FAR, the concurrent decline in CSI (which correlates to prediction accuracy) also becomes evident.

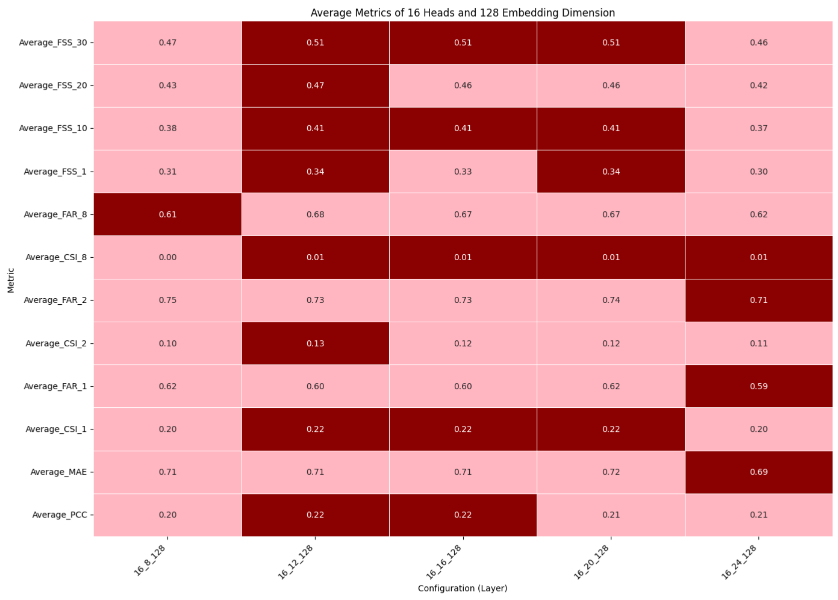


Figure 6.11: Performance of different layers with 16 heads and 128 embedding dimensions.

6.1.3.3. Embedding Dimension Analysis: 16 Heads with 256

Moreover, with an embedding dimension of 256, the model unveils significant findings regarding performance variation among different layer configurations (6.12). The 24-layer model asserts dominance by registering the top performance in CSI at all thresholds and FSS across spatial extents, alongside PCC, marking pronounced predictive strengths. Interestingly, the 20-layer configuration surpasses the 24-layer model in MAE and FAR but at the cost of diminished capabilities in CSI. Therefore, the model (heads: 16, layers: 24, embedding: 256) is one of the candidates that will be used later to select the optimal model.

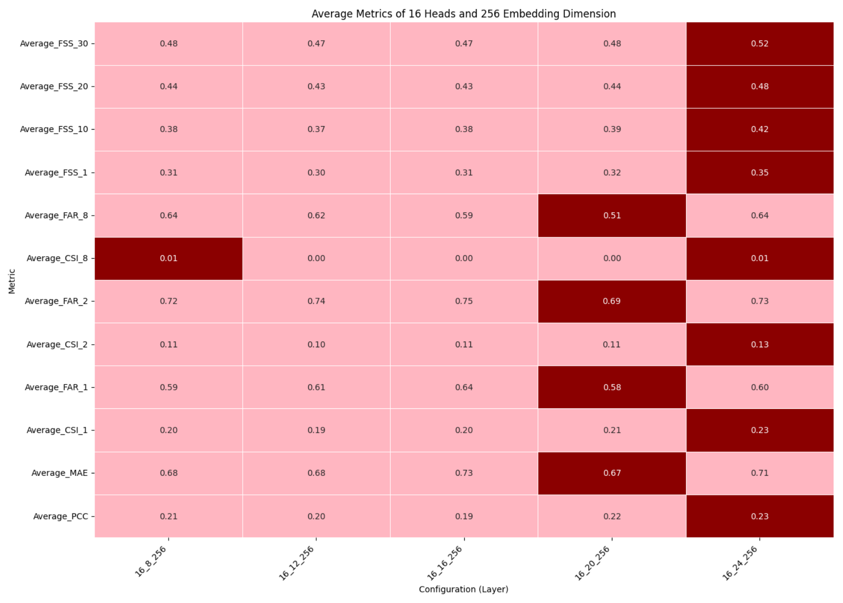


Figure 6.12: Performance of different layers with 16 heads and 256 embedding dimensions.

6.1.3.4. Embedding Dimension Analysis: 16 Heads with 512

As mentioned before (section), since the model was too complex with 4 heads, increasing the number of heads to 16 will result in a more complex model. Hence, the model will over-fit. Therefore, these configurations (heads:16, layers: 8,12,16,20,24, embedding: 512) are excluded from the analysis.

6.1.3.5. Selection of the optimal model at 16 heads

To identify the most effective model configuration for a transformer architecture featuring 16 heads, an array of configurations has demonstrated notable potential across various em-bedding dimensions, specifically (heads: 16, layers: 12, embedding: 64), (heads:16, layers: 12, embedding: 128), and (heads: 16, layers: 24, embedding:256). A comprehensive evaluation of these configurations is essential to discern the optimal model (6.13).

In the context of utilizing a higher head count, the analysis reveals that the configuration with the smallest embedding dimension, (heads: 16, layers: 12, embedding: 64), surpasses its counterparts in performance. This model stands out for its exceptional PCC, MAE, FSS, FAR, and CSI across both low and high rainfalls intensities.

Nonetheless, it is important to note that, despite its superior overall performance, the (heads: 16, layers: 12, embedding: 64) model's CSI at medium rainfall intensities is better in configurations with different em-bedding dimensions. This advantage, however, comes with trade-offs, including distortions in spatial scale accuracy, a reduction in PCC, and an elevation in FAR. This observation highlights the intricate balance required between maintaining model accuracy and navigating the complexities inherent in accurately predicting rainfall distribution.

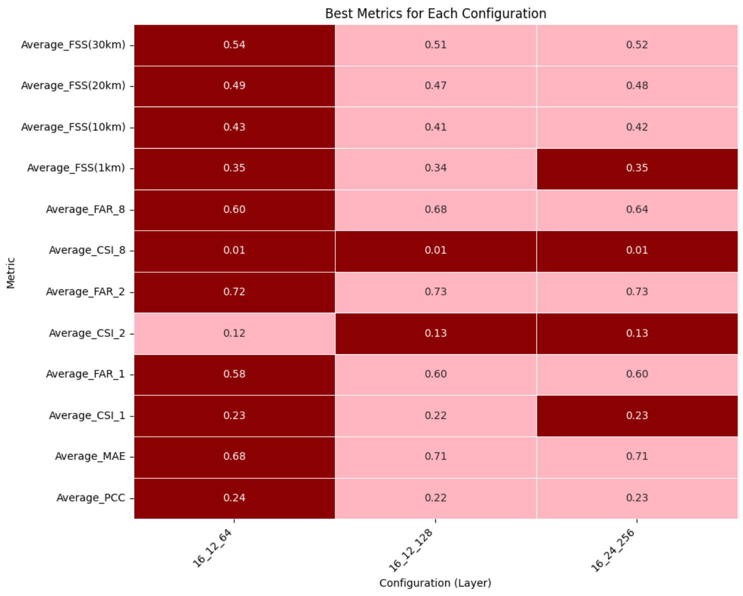


Figure 6.13: Performance of different layers with best selected models at 16 heads.

6.1.4. Preliminary Results

After meticulous analysis to determine the optimal model configuration by varying the number of heads, layers, and embedding dimensions, we've identified the best configuration for each head count: (Heads: 4, layers: 8, embedding: 64), (Heads: 8, layers: 24, embedding: 256), and (Heads: 16, layers: 12, embedding: 64). The subsequent comparison of these configurations is pivotal in identifying the superior model.

The analysis, guided by a heat-map comparison (6.14), indicates a close competition primarily between the (Heads: 4, layers: 8, embedding: 64), (Heads: 8, layers: 24, embedding: 256) models. Even though the (Heads: 16, layers: 12, embedding: 64) model excelled in MAE compared to its counterparts, it did not lead in all other critical metrics. Hence, this model falls short and is not the optimal model.

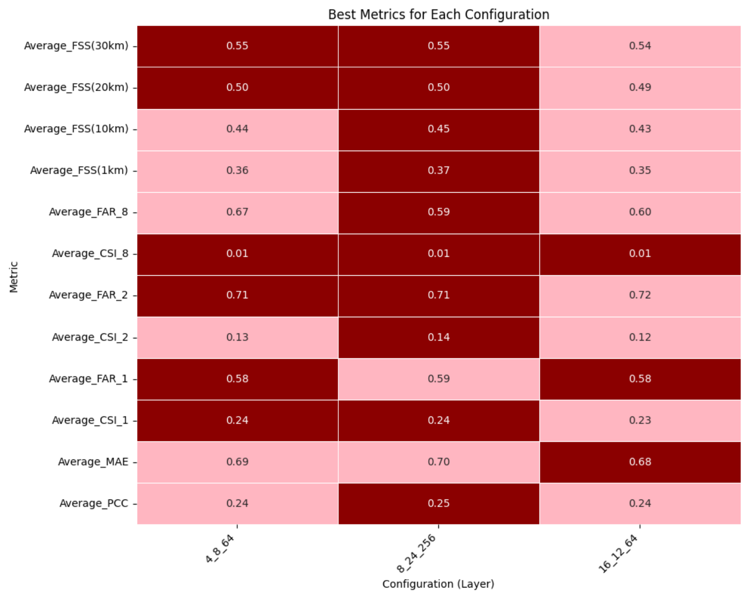


Figure 6.14: Performance of the best selected models.

The model configured with 8 heads and an embedding dimension of 256 (Heads: 8, layers: 24, embedding: 256) demonstrates superior performance, especially across different FSS scales, CSI, and FAR at all thresholds. Notably, this model shows the most significant advantage at higher rainfall thresholds when compared with the (Heads: 4, layers: 8, embedding: 64) configuration.
However, it's crucial to acknowledge that despite the superior performance of the (Heads: 8, layers: 24, embedding: 256) model over others, the marginal difference between the (Heads: 4, layers: 8, embedding: 64) and the (Heads: 8, layers: 24, embedding: 256) configuration remains modest across most metrics. This suggests that while the (Heads: 8, layers: 24, embedding: 256) model stands out in terms of accuracy and reliability, the improvement it offers over the (Heads: 4, layers: 8, embedding: 64) model is not overwhelmingly large considering the computational requirements and the generation time. Based on this analysis, the (Heads: 4, layers: 8, embedding: 64) is considered the preliminary optimal model. Nonetheless, further analysis will be done to assess the best models found in each head: 4-8-64, 4-8-128, 4-16-128 (section: 6.1.1.5), 8-12-64, 8-16-128, 8-24-256 (section: 6.1.2.5), 16-12-64, 16-12-128 and 16-24-256 (section: 6.1.3.5).

6.2. Analysis of Optimal Transformer Configuration at Catchment Level

The analysis discussed in the previous section encompasses the entire area of the Netherlands. The findings indicate that certain model configurations outperform others when evaluated against specific metrics. To acquire a more detailed understanding of the capabilities of the chosen model configuration—4-8-64—an additional experiment has been conducted. This experiment involves isolating selected sections from precipitation maps predicted by various models to focus specifically on relevant catchment areas. Each area has a defined extreme rainfall threshold set at the top 1% of historical data, equating to 5 mm of rainfall per 3-hour period, recorded from 2008 to 2014. An extreme rainfall event is defined as any instance where the average rainfall over a 3-hour period exceeds this threshold. To enhance the assessment of the models, varying thresholds ($thr_{varying}$) are applied across all predictions for the catchment areas while maintaining the ground truth threshold ($thr_{ground-truth}$) constant. This methodology evaluates the models' effectiveness under diverse conditions. The predicted precipitation, X_{pre}, and the observed precipitation, X_{obs}, are calculated as follows:

$$\begin{aligned} X_{\text{pre}} &= \left(X^{\text{pre}}_{T+30} + X^{\text{pre}}_{T+60} + \ldots + X^{\text{pre}}_{T+180}\right) \times \frac{1}{6} \times 3 \\ X_{\text{obs}} &= \left(X^{\text{obs}}_{T+30} + X^{\text{obs}}_{T+60} + \ldots + X^{\text{obs}}_{T+180}\right) \times \frac{1}{6} \times 3 \end{aligned} \tag{6.1}$$

The evaluation of model performance is categorized into four outcomes, with each variation of the threshold ranging from 10 mm/3hrs to 0.5 mm/3hrs in 0.5 mm/3hrs decrements (20 data points). This stepwise adjustment of the threshold yields a series of results that are classified as follows:

True Positive: $Obs_{precipitation} \geq thr_{grdtruth}$ and $Pred_{precipitation} \geq thr_{varying}$
Observed precipitation exceeds the ground truth threshold, and the predicted precipitation also surpasses the varying threshold.

False Negative: $Obs_{precipitation} \geq thr_{grdtruth}$ and $Pred_{precipitation} < thr_{varying}$
Observed precipitation exceeds the ground truth threshold, but the predicted precipitation falls below the varying threshold.

True Negative: $Obs_{precipitation} \leq thr_{grdtruth}$ and $Pred_{precipitation} < thr_{varying}$
Both the observed and predicted precipitation are below their respective thresholds.

False Positive: $Obs_{precipitation} \leq thr_{grdtruth}$ and $Pred_{precipitation} \geq thr_{varying}$
Observed precipitation is below the ground truth threshold, but the predicted precipitation exceeds the varying threshold.

These outcomes are recorded and subsequently used to plot the Receiver Operating Characteristic (ROC) curve, which illustrates the relationship between the False Alarm

Rate (FAR) and the Hit Rate (HR), and the Precision Recall curve, which shows the relationship between Recall (or Hit Rate) and Precision. These curves enhance our understanding of the precision and reliability of the rainfall prediction models across varying thresholds, thereby aiding in the improvement of forecasting and management strategies for extreme weather events. The matrices used in plotting the ROC and the PRC.

False Alarm Rate (FAR)	:	$FAR = \dfrac{FP}{FP + TN}$
Recall or Hit Rate (HR)	:	Recall $= \dfrac{TP}{TP + FN}$
Precision	:	Precision $= \dfrac{TP}{TP + FP}$

This structured analysis assesses the effectiveness of models in predicting rainfall events. The results of these experiments are illustrated in fig. 6.15. Both figures show that the configurations 4-8-64 and 8-12-64 outperform all other configurations when comparing the Area Under Curve (AUC) . These observations corroborate the findings in the previous section regarding the selection of the optimal model.

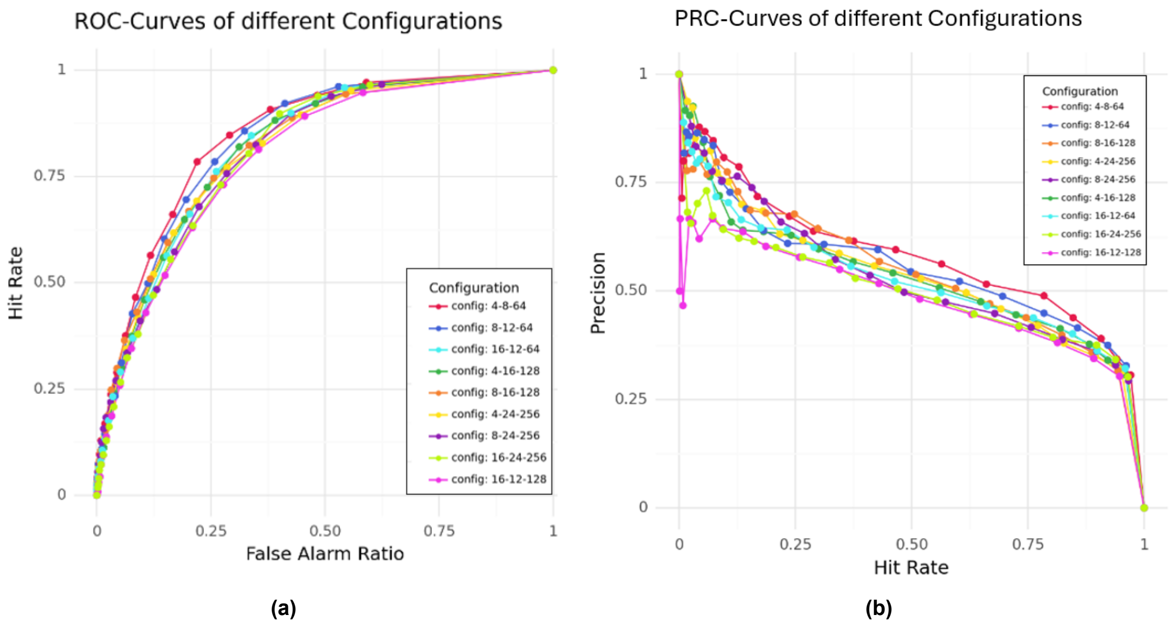


Figure 6.15: Figure shows the ROC-PRC curve of different configurations, the 4-8-64 and 8-12-64 performs the best among all used configurations.

6.3. Analysis of the Attention Mechanism

Understanding the attention weight distribution within a Transformer model is crucial for optimizing its performance. By investigating how the model assigns attention across different parts of a sequence, valuable insights into its operation can be gained, leading to adjustments in its configurations. Our investigation aimed to discern why certain model configurations, such as those with 4 heads, 8 layers, and an embedding dimension of 64, outperform others. We hypothesized that the effectiveness of a Transformer model depends not only on its architectural parameters but also on how it allocates attention across the input sequence.

Diversity in attention allocation across heads and layers within a Transformer model is essential for capturing the full spectrum of information present in the input sequence. Each attention head and layer may focus on different aspects or features of the input, enriching the overall interpretation of the data. While some level of consistency in attention patterns may be expected, variations across layers and heads are necessary for comprehensive feature capture. These deviations from uniformity are crucial for ensuring the model's efficiency in capturing all relevant features within the input sequence. Insufficient attention to particular details may compromise the model's capacity to fully grasp the input, potentially resulting in the neglect of vital elements within the sequence.

Cosine similarity analysis provides a detailed method for evaluating the degree of similarity between the attention mechanisms of different heads or layers within a transformer model. This analysis begins by extracting attention scores from the model, reflecting where the model focuses within the input data. Each set of scores from the heads or layers is converted into flat vectors, and the cosine similarity between these vectors is calculated using the formula:

$$CosineSimilarity = S_C(u, v) = \frac{u \cdot v}{\|u\|\|v\|} \tag{6.2}$$

where u and v are the attention vectors from two different heads or layers.

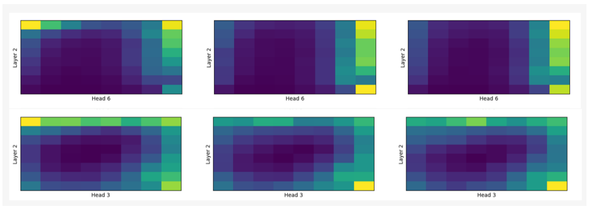
To assess the performance consistency and diversity of attention mechanisms, we focused initially on a specific layer, such as layer 1, and calculated the cosine similarity across all heads within this layer. This step helped identify which heads behave similarly, showing consistency in their attention patterns at this specific level. Subsequently, we analyzed a single head across different layers to determine the degree of similarity in its attention mechanism through various stages of the model. The compiled similarities create an overview of how differently parts of the model process information, visualized to facilitate deeper analysis.

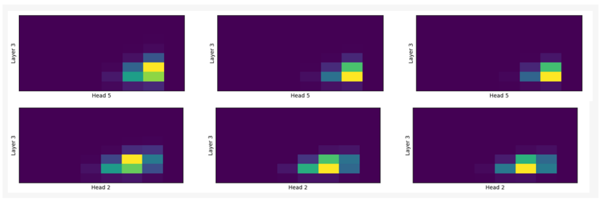
Notably, a high mean cosine similarity coupled with low variance among these values often indicates redundancy, suggesting that several heads or layers are paying atten-

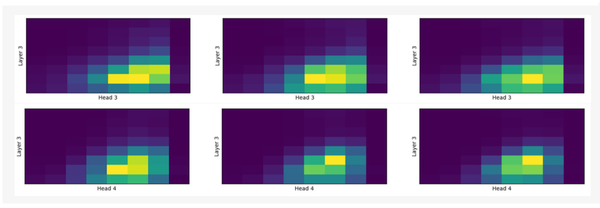
tion to the same features without adding unique insights or value. Conversely, a low mean cosine similarity accompanied by high variance suggests that the model distributes its attention across various features, indicating diversity in attention allocation and helping to avoid redundancy. This dual approach – examining both across heads within a single layer and across layers within a single head – provides crucial insights that can be used to refine and enhance the model's interpret-ability and efficiency.
To better understand the distribution of attention weights and to justify the selection of our model, we initiated our analysis by applying a cosine similarity matrix to a transformer featuring 8-24-256. This configuration will be compared with 16-12-64 as well as the best-performing configurations in the catchment level analysis, featuring 4-8-64 and 8-12-64. By computing and comparing the cosine similarity matrices across each layer and examining each head, we observed significant redundancy and repetition in the focus areas of the transformer. This was evident from the repetitive patterns in the attention mechanisms.

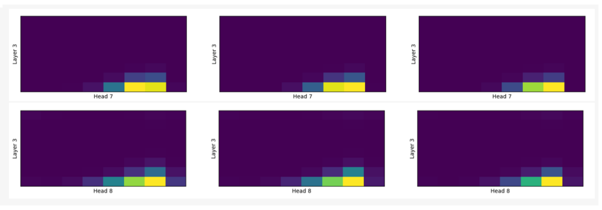
6.3.1. Attention Mechanism Analysis: 8-24-256

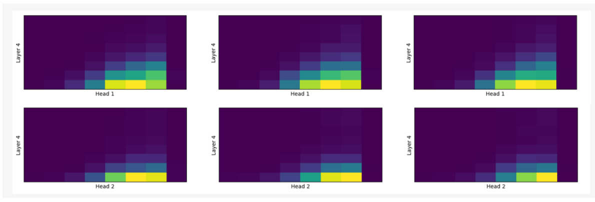
An analysis of attention head redundancy across different layers reveals distinct patterns of focus overlap among various heads. In the second layer, attention heads 3 and 6 highlight shared features in their focus areas. In the third layer, heads 2 and 5, heads 3 and 4, and heads 7 and 8 demonstrate shared focus points. The fourth layer presents an overlap between heads 1 and 2. Moving to the twenty-fourth layer, overlaps are also seen between heads 1 and 2. These similarities are illustrated in Fig. 6.16.


(a) Layer 2: Head 3 vs. Head 6


(b) Layer 3: Head 2 vs. Head 5


(c) Layer 3: Head 3 vs. Head 4


(d) Layer 3: Head 7 vs. Head 8


(e) Layer 4: Head 1 vs. Head 2

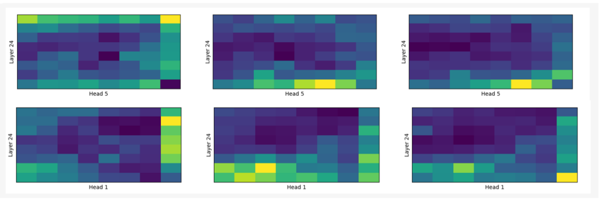
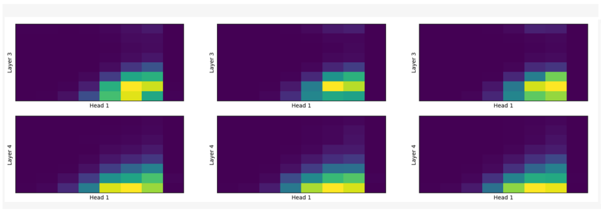
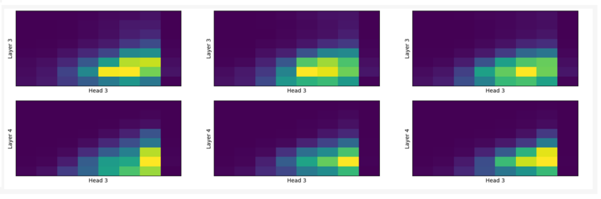

(f) Layer 24: Head 1 vs. Head 5

Figure 6.16: Exploring Redundancy Across 8 Attention Heads in Different Layers of configuration 8-24-256.

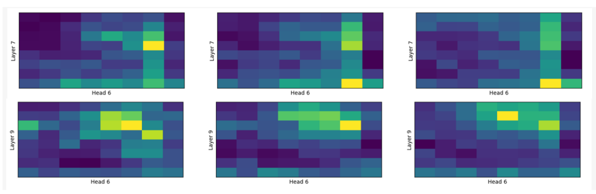
Similarities also appear in the same heads across different layers. In the first head, attention layers 3 and 4 exhibit similarities in their focus areas. In the sixth head, attention layers 7 and 9 display similar patterns. The third head presents an overlap between layers 3 and 4. Moving to the eighth head, overlaps are observed between layers 22 and 23, and further shared focus points are demonstrated between layers 23 and 24. These similarities are illustrated in Fig. 6.17.



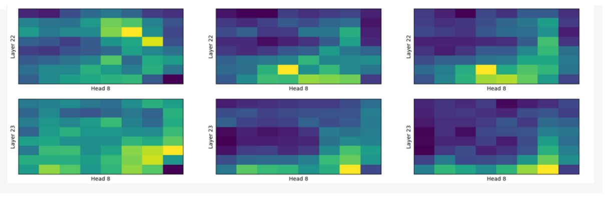
(a) Head 1: Layer 3 vs. Layer 4



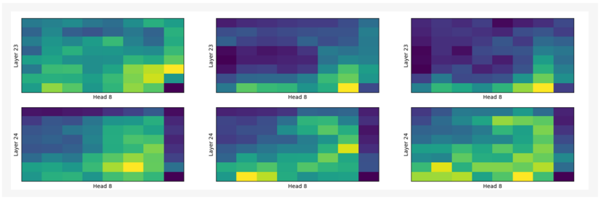
(b) Head 3: Layer 3 vs. Layer 4



(c) Head 6: Layer 7 vs. Layer 9



(d) Head 8: Layer 22 vs. Layer 23



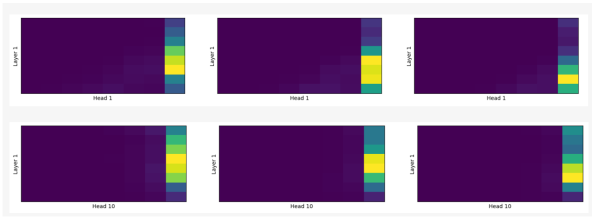
(e) Head 8: Layer 23 vs. Layer 24

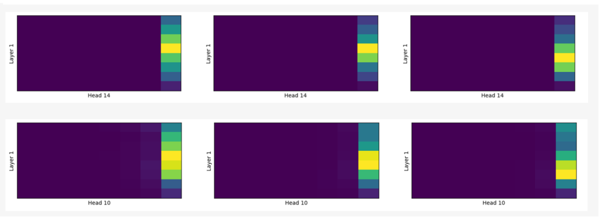
Figure 6.17: Exploring Redundancy Across 24 Attention Layers in Different Heads of configuration 8-24-256.

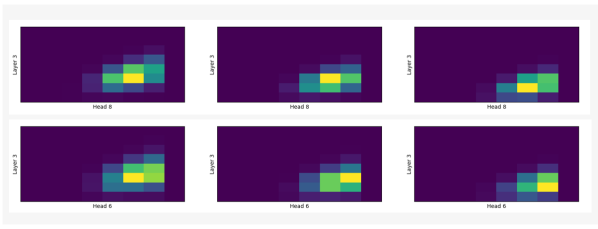
6.3.2. Attention Mechanism Analysis: 16-12-64

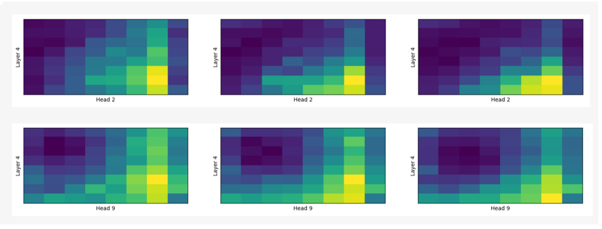
Analyzing attention head redundancy across different layers reveals distinct focus overlap patterns. In the first layer, heads 1 and 10, heads 1 and 14, and heads 10 and 14 exhibit similarities in their focus areas, with heads 10 and 14 notably sharing overlapping patterns. In the third layer, heads 5 and 15, heads 13 and 14, and heads 6 and 8 show shared focus points. The fourth layer features overlap between heads 2 and 9.

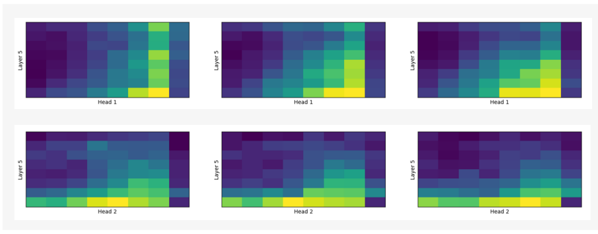
Moving to the fifth layer, overlaps occur between heads 1 and 2, and heads 2 and 16. The seventh layer highlights similarities between heads 4 and 13. In the eighth layer, heads 6 and 10 exhibit similar focus patterns. By the tenth layer, overlaps between heads 5 and 13, and heads 5 and 16, indicate a recurring theme of attention redundancy. These findings are illustrated in Fig. 6.18.

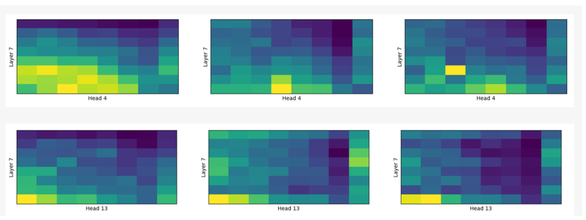

(a) Layer 1: Head 1 vs. Head 10

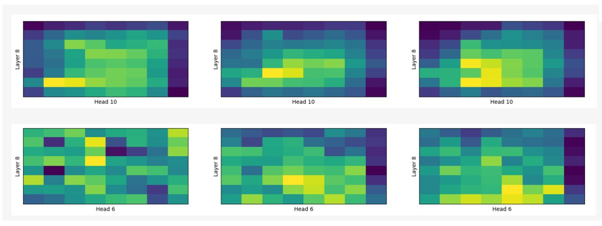

(b) Layer 1: Head 14 vs. Head 10


(c) Layer 3: Head 8 vs. Head 6


(d) Layer 4: Head 2 vs. Head 9


(e) Layer 5: Head 1 vs. Head 2


(f) Layer 7: Head 4 vs. Head 13


(g) Layer 8: Head 10 vs. Head 6

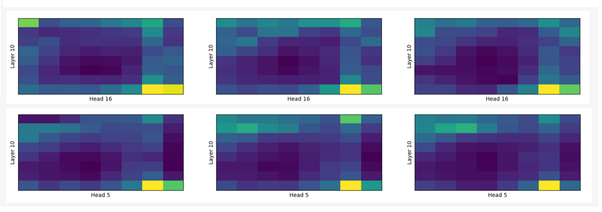
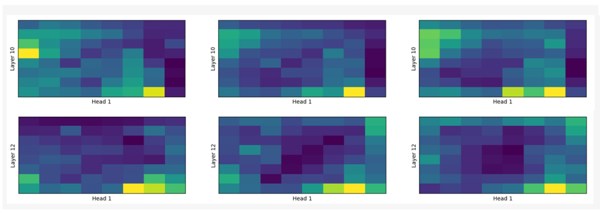
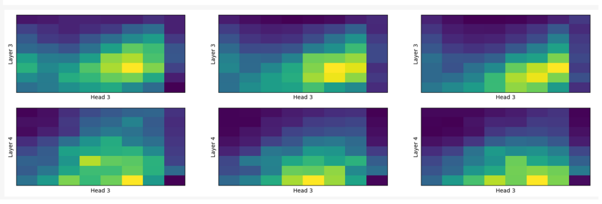

(h) Layer 10: Head 16 vs. Head 5

Figure 6.18: Exploring Redundancy Across 16 Attention Heads in Different Layers of configuration 16-12-64.

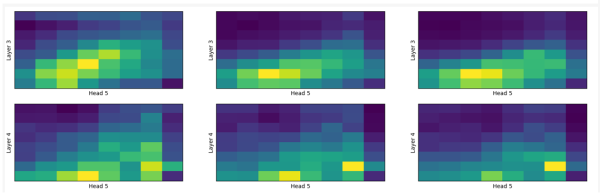
Similar patterns are observed across heads. The first head shows consistent attention between layers 10 and 12. The third head displays similar focus in layers 10 and 12. The ninth head reveals overlaps between layers 2 and 3. For head 12, similarities occur in layers 2 and 4, while head 13 shows matching patterns in layers 5 and 6. Head 15 highlights attention between layers 4 and 8, and head 16 exhibits redundancy in layers 1 and 3, as well as layers 5 and 12. Fig. 6.19 illustrates the redundancy of the 12 layers across different heads.



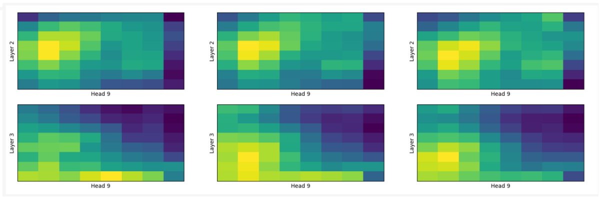
(a) Head 1: Layer 10 vs. Layer 12



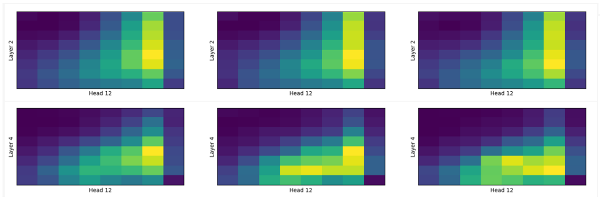
(b) Head 3: Layer 3 vs. Layer 4



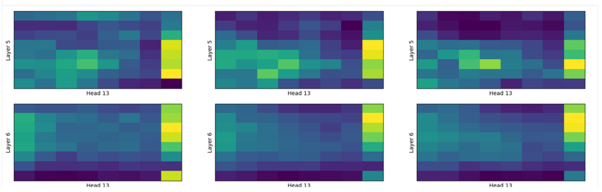
(c) Head 5: Layer 3 vs. Layer 4



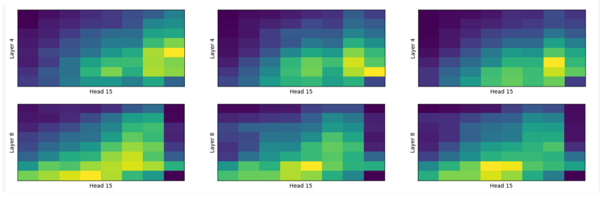
(d) Head 9: Layer 2 vs. Layer 3



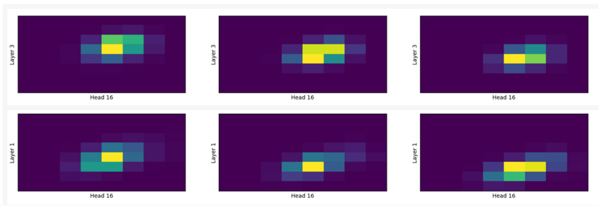
(e) Head 12: Layer 2 vs. Layer 4



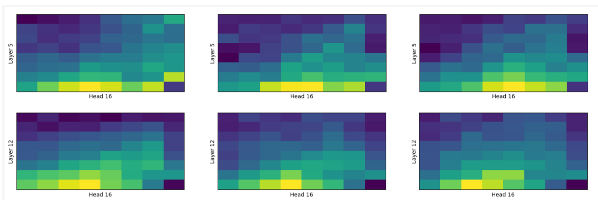
(f) Head 13: Layer 5 vs. Layer 6



(g) Head 15: Layer 4 vs. Layer 8



(h) Head 16: Layer 1 vs. Layer 3

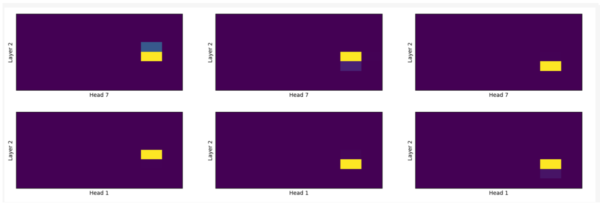


(i) Head 16: Layer 5 vs. Layer 12

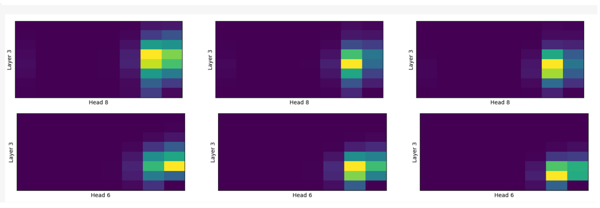
Figure 6.19: Exploring Redundancy Across 12 Attention Layers in Different Heads of configuration 16-12-64.

6.3.3. Attention Mechanism Analysis: 8-12-64

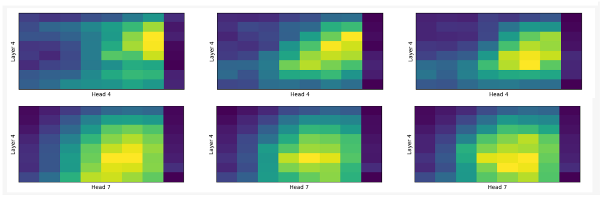
Further analysis is done with 8-12-64 configuration, as shown in fig. 6.21. The figure illustrates how different heads exhibit similar behavior when examining a specific layer. Heads 1 and 7 indicate similar focus at layer 2. Shifting the focus to layer 3, head 6 and 8 reveals parallel attention pattern while heads 4 and 7 demonstrate similar focus at layer 4. Head 1 and 5 highlighted shared attention areas, and head 2 and 3 present comparable attention at layer 7.



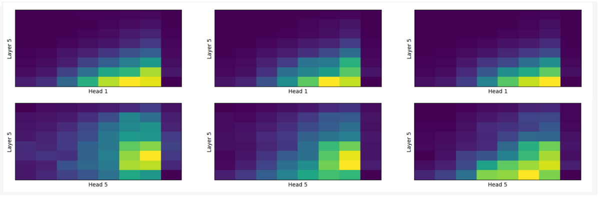
(a) Layer 2: Head 1 vs. Head 7



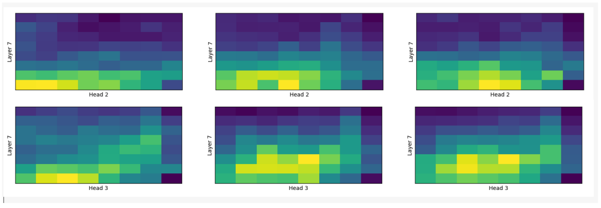
(b) Layer 3: Head 6 vs. Head 8



(c) Layer 4: Head 4 vs. Head 7



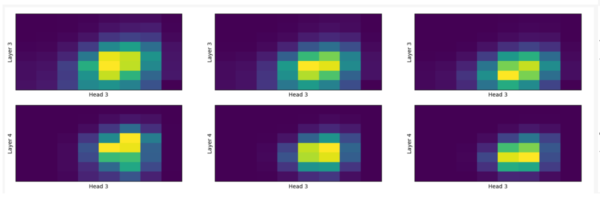
(d) Layer 5: Head 1 vs. Head 5



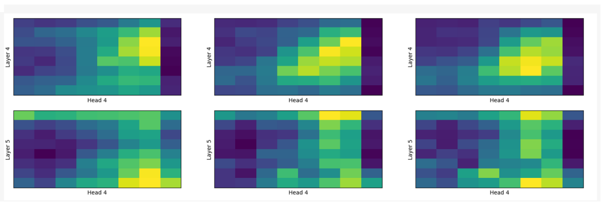
(e) Layer 7: Head 2 vs. Head 3

Figure 6.20: Exploring Redundancy Across 8 Attention Heads in Different Layers of configuration 8-12-64.

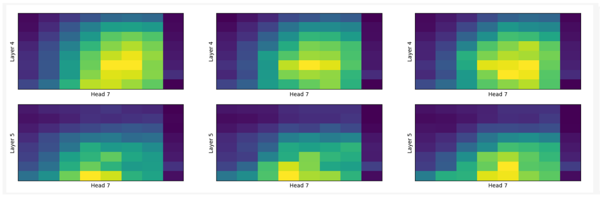
Examining the redundant information across layers within specific heads, Figure 6.21 illustrates the redundancy patterns of different heads. Layers 10 and 11 show redundancy in head 1, while layers 6 and 7 display commonalities in head 2. In head 3, layers 3 and 4 reveal similar patterns. Heads 4 and 7 indicate similar focus between layers 4 and 5. Comparable attention is found in head 5 across layers 4 and 7, and head 8 shows similar patterns in layers 8 and 9.



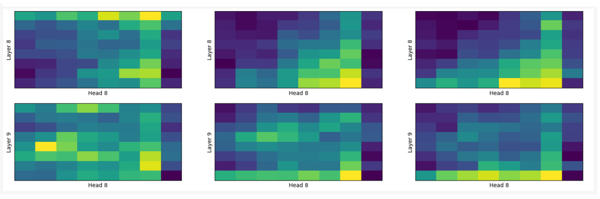
(a) Head 3: Layer 3 vs. Layer 4



(b) Head 4: Layer 4 vs. Layer 5



(c) Head 7: Layer 4 vs. Layer 5

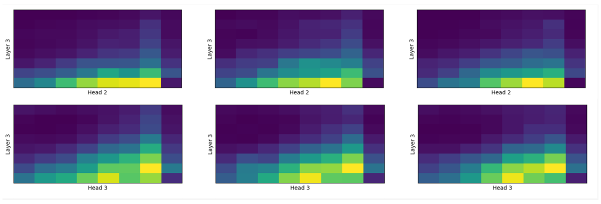


(d) Head 8: Layer 8 vs. Layer 9

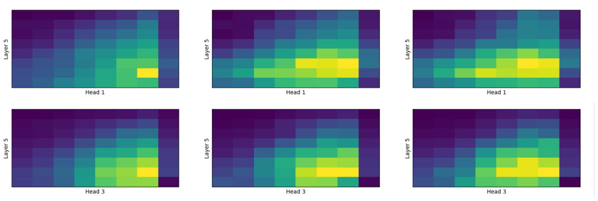
Figure 6.21: Exploring Redundancy Across 12 Attention Layers in Different Heads of configuration 8-12-64.

6.3.4. Attention Mechanism Analysis: 4-8-64

The final analysis involves the best-performing model configuration, 4-8-64, across both the entire Netherlands and the catchment level. This analysis begins by examining the behavior of layers across different heads. Heads 2 and 3 exhibit similar patterns at layer 3, while heads 1 and 3 present comparable attention at layer 5. This pattern is displayed in figure 6.22.



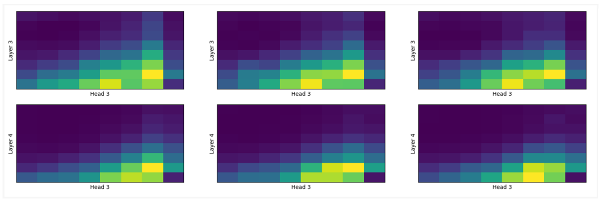
(a) Layer 3: Head 2 vs. Head 3



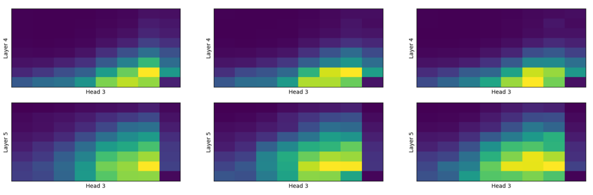
(b) Layer 5: Head 1 vs. Head 3

Figure 6.22: Exploring Redundancy Across 4 Attention Heads in Different Layers of configuration 4-8-64.

Exploring redundancy across the same heads but different layers reveals that layers 3 and 4, as well as layers 4 and 5, exhibit similar behavior in head 3 as shown in 6.23.



(a) Head 3: Layer 3 vs. Layer 4



(b) Head 3: Layer 4 vs. Layer 5

Figure 6.23: Exploring Redundancy Across 8 Attention Layers in Different Heads of configuration 4-8-64.

6.4. Results

After analyzing the attention mechanisms of different configurations, it is evident that 8-24-256 and 16-12-64 exhibit clear attention patterns across different layers and heads. However, the 8-12-64 configuration shows lower redundancy across layers and heads compared to the previously mentioned configurations, and the 4-8-64 configuration exhibits very little redundancy. This illustrates that increasing the complexity of the model does not always lead to better performance. Large models tend to have redundant parameters that do not contribute significantly to the model's effectiveness. In contrast, the 4-8-64 configuration, with fewer parameters, minimizes redundancy and allows the model to focus on learning more salient features. This improves the model's ability to generalize from training to testing data. Additionally, having fewer parameters smooths the training process, enabling the model to converge faster and operate more efficiently.

7 Comparison Between Models

After optimizing the transformer, the best configuration—with 4 heads, 8 layers, and a 64-dimensional embedding—is used to evaluate the Nowcasting-GPT model against PySTEPS and Nuwa-Pytorch (ref Haoran). These models are selected to assess the performance of Nowcasting-GPT in comparison to another deep generative model and a state-of-the-art model. Nuwa-Pytorch, the deep generative model, combines a Vector-Quantized Generative Adversarial Network (VQGAN) with an auto-regressive transformer and includes an additional extreme value loss function to detect extreme events. PySTEPS, on the other hand, serves as a benchmark for evaluating the performance of state-of-the-art models against the deep generative models, Nowcasting-GPT and Nuwa-Pytorch.

In this chapter, Nowcasting-GPT is evaluated against both models. The first section assesses the predictions of Nowcasting-GPT for the entire Netherlands using both continuous and categorical metrics such as PCC, MAE, FSS, CSI, and FAR, as discussed in chapter 5.

The second section focuses on catchment-level analysis, which is crucial for understanding the model's performance in areas where precipitation accumulates over time. In this analysis, the average precipitation accumulation over a 3-hour period in specific catchment areas is calculated using data from the preceding forecasting outcomes. This calculation is then compared with the established threshold for extreme events, following the procedure discussed in the previous chapter, section 6.2. The model's performance in detecting weather events at specific thresholds is assessed using categorical metrics for binary classification, including HR, FA, CSI, FAR, the AUC of the ROC curve, and the Precision-Recall curve.

7.1. Nowcasting-GPT Performance Across the Netherlands

Continuous Metrics Analysis: PCC and MAE

The Pearson Correlation Coefficient (PCC) averaged over time for each model is shown in figure 7.1. Our Nowcasting-GPT model demonstrates consistently higher PCC than the other two models, indicating that its predictions align more closely with the actual data throughout the forecast period. The threshold of $1/e$ is considered the benchmark for a nowcast to be deemed skilful. Despite the integration of an extreme value loss function which is designed to better capture extreme weather conditions in the Nuwa-Pytorch model, it under-performed compared to the Nowcasting-GPT model, which does not specifically account for extreme events. The Nowcasting-GPT model exhibits skilful predictions up to 40 minutes into the future and maintains superior performance at subsequent lead times relative to the other models.

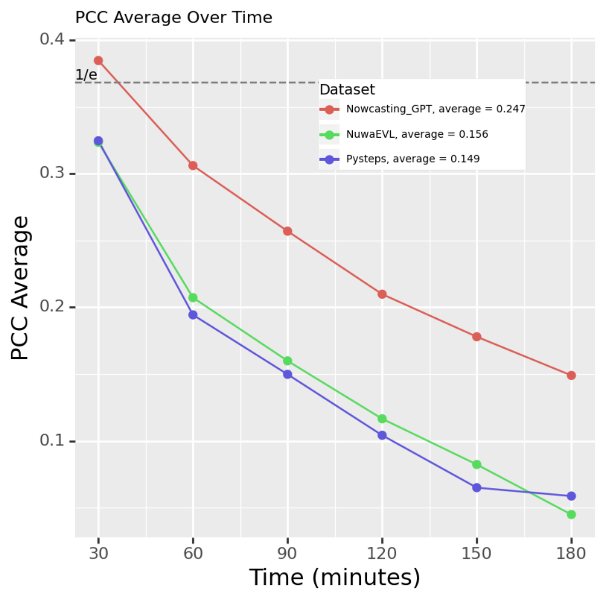


Figure 7.1: PCC metric evaluation over the 6 lead times. Each point represents the average value for a specific lead time over the whole dataset. Higher values represent better performance. Nowcasting-GPT outperformed other models.

Furthermore, Nowcasting-GPT demonstrates a significant reduction in Mean Absolute Error (MAE) compared to the other two models, both on average and at each lead time [7.2]. This indicates that the precipitation amounts forecasted by the Nowcasting-GPT method are, on average, closer to the actual observed amounts than those forecasted by Nuwa-Pytorch and PySTEPS. This improvement could be attributed to our model's approach of treating all tokens equally; however, in Nuwa-Pytorch, the model assigns more weight to extreme tokens, which may lead to a higher MAE.

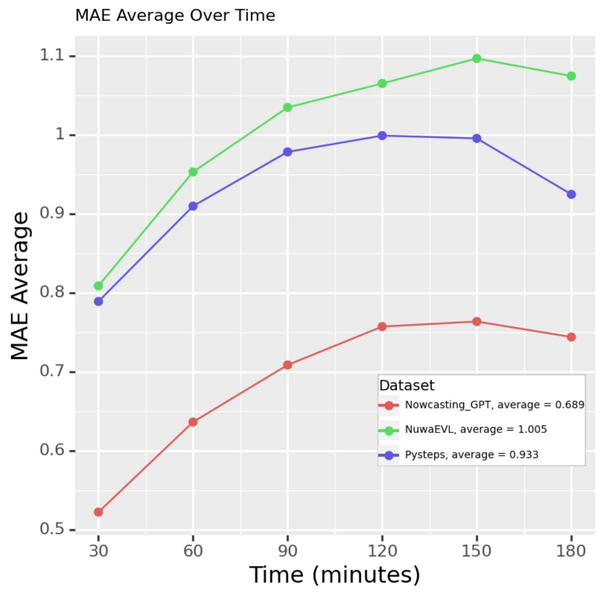


Figure 7.2: MAE metric evaluation over the 6 lead times. Each point represents the average value for a specific lead time over the whole dataset. Lower values represent better performance. Nowcasting-GPT outperformed other models.

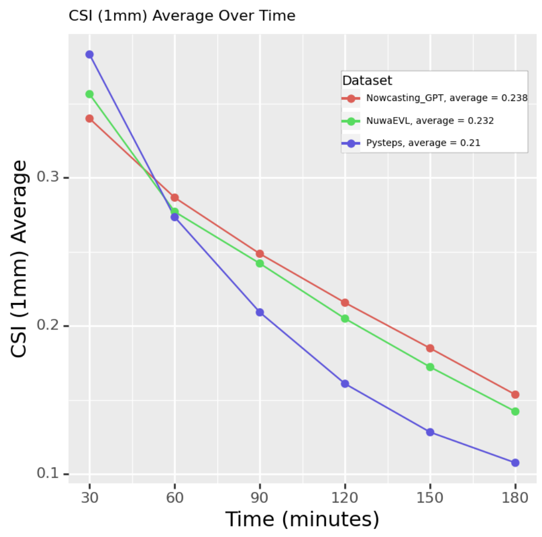
Critical Success Index and False Alarm Ratio at Different Thresholds:

To assess the model's performance accurately, we utilized the Critical Success Index (CSI) and the False Alarm Ratio (FAR) as key metrics. The CSI measures the model's ability to accurately forecast rainfall events at predetermined intensity thresholds, highlighting its precision. On the other hand, the FAR quantifies the rate at which the model incorrectly forecasts non-occurring events. A lower FAR signifies enhanced model reliability, suggesting a higher probability that forecasted events will indeed occur.

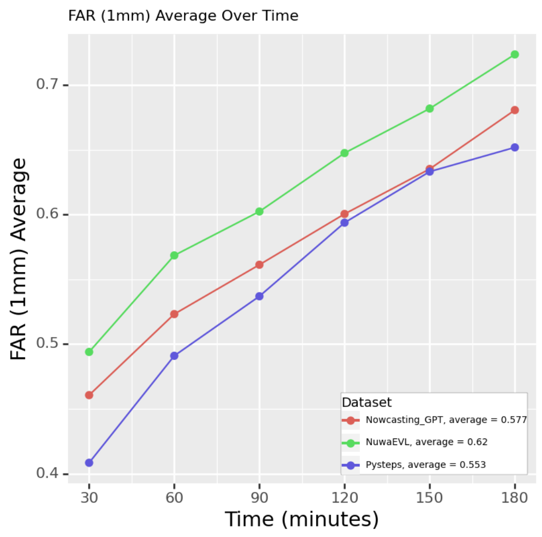
Our evaluation included examining CSI values at 1mm, 2mm, and 8mm thresholds, which allowed us to assess the model's capability in detecting light, moderate, and heavy rainfall events, respectively. The analysis shows that, on average, the Nowcasting-GPT model outperformed PySTEPS and Nuwa-Pytorch in identifying light and moderate rainfall events. However, at a 30-minute lead time, it falls short against these models, potentially due to challenges in quickly adjusting to sudden shifts in rainfall intensity or in dealing with the initial conditions' temporal and spatial dependencies. Yet, at longer lead times, Nowcasting-GPT demonstrates its strength, accurately forecasting the progression of weather patterns over time.

Moreover, our FAR analysis reveals that at the 1mm (7.3b) and 2mm (7.4b) thresholds, Nowcasting-GPT's forecasts are more reliable than Nuwa-Pytorch's, consistently across various lead times. While PySTEPS presents a lower FAR at the 1mm threshold, indicating fewer false alarms, it also shows a reduced CSI, pointing to a decrease in accurately identifying actual rainfall events. At the 2mm threshold, although PyS-

TEPS and Nowcasting-GPT exhibit similar FAR values, Nowcasting-GPT achieves a higher CSI (7.3a), emphasizing its superior accuracy in detecting rainfall events, particularly for moderate rainfall scenarios.

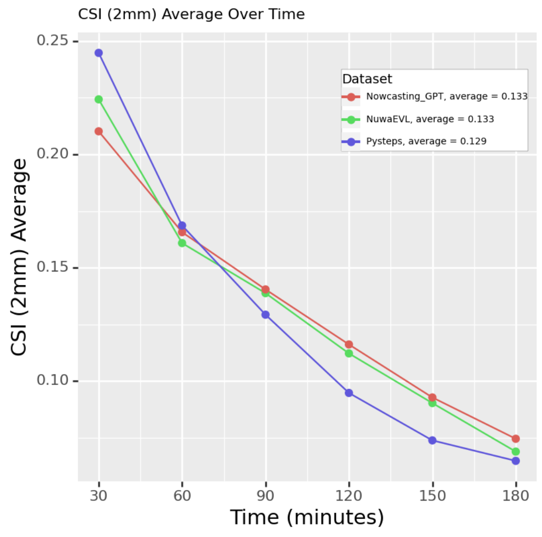


(a) CSI(1mm): Higher values represent better performance.

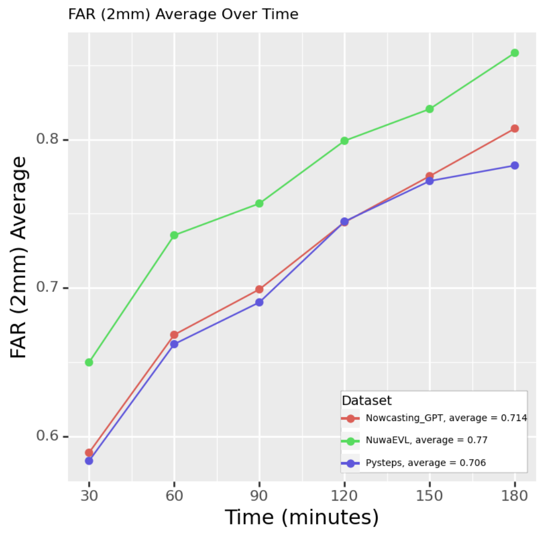


(b) FAR(1mm): Lower values represent better performance.

Figure 7.3: CSI(1mm) and FAR(1mm) metric evaluation over the 6 lead times. Each point represents the average value for a specific lead time over the whole dataset.



(a) CSI(2mm): Higher values represent better performance.

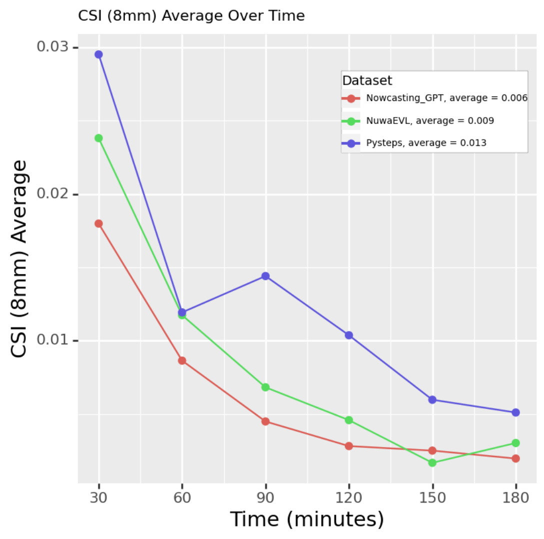


(b) FAR(2mm): Lower values represent better performance.

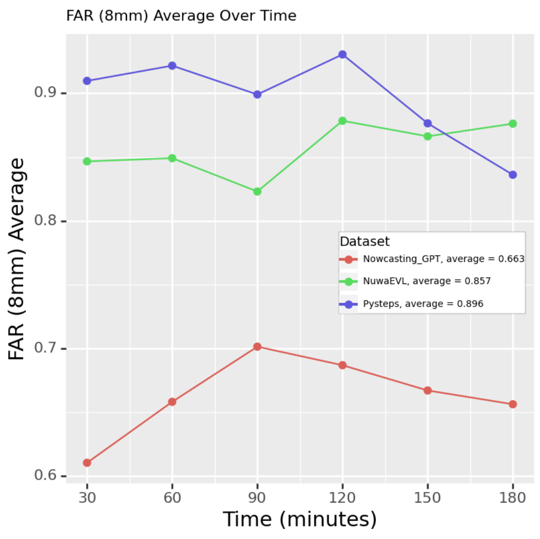
Figure 7.4: CSI(2mm) and FAR(2mm) metric evaluation over the 6 lead times. Each point represents the average value for a specific lead time over the whole dataset.

At the 8mm threshold, an interesting pattern emerges where both PySTEPS and

Nuwa-Pytorch outperform Nowcasting-GPT in the accuracy of rainfall detection (7.5). Despite this, they also register a higher FAR compared to Nowcasting-GPT. This indicates that while PySTEPS and Nuwa-Pytorch may be more proficient in detecting heavy rainfall events, they also tend to predict more non-occurring events, reducing their overall reliability. Even though Nowcasting-GPT is not as effective in capturing heavy rainfall events, its predictions show a lower FAR. The difference in the outcomes highlights the trade-offs between achieving high accuracy in event detection and maintaining low rates of false alarms in weather forecasting models.



(a) CSI(8mm): Higher values represent better performance.



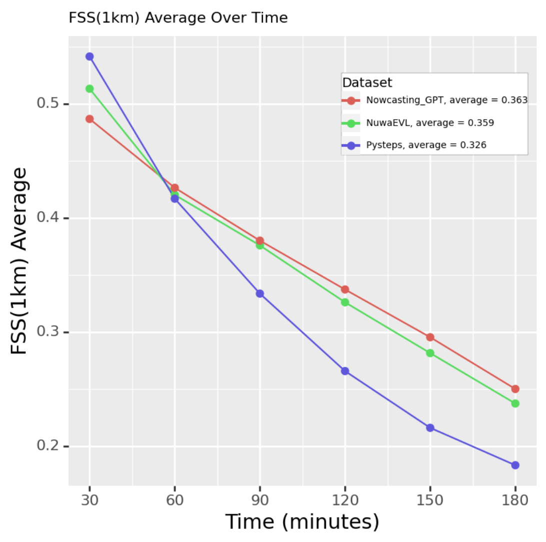
(b) FAR(8mm): Lower values represent better performance.

Figure 7.5: CSI(8mm) and FAR(8mm) metric evaluation over the 6 lead times. Each point represents the average value for a specific lead time over the whole dataset.

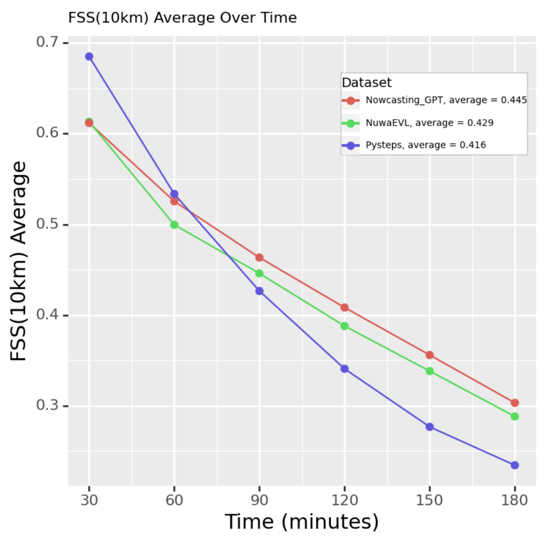
Fractional Skill Score

The Fractional Skill Score (FSS) is a metric for assessing the accuracy of spatial forecasts, which is useful for evaluating the performance of model predictions over a given area. A model is deemed skillful if its FSS exceeds the threshold of $0.5 + 1/f_0$, where f_0 represents the ratio of the window size to the overall image size. This criterion emphasizes the intuitive notion that forecasting over smaller areas is inherently more challenging than over larger areas. Therefore, higher FSS values (with large window sizes, e.g., 30 km) indicate better model performance.
PySTEPS consistently exhibits skillfulness in FSS at a 30-minute lead time across various spatial scales. In contrast, Nowcasting-GPT demonstrates superior performance among all models for lead times beyond 60 minutes, even though it doesn't always meet the $0.5+1/f_0$ criterion. Nowcasting-GPT meets this criterion up to approximately 70 minutes at a 1 km scale, up to 100 minutes at a 10 km scale, and up to 115 minutes

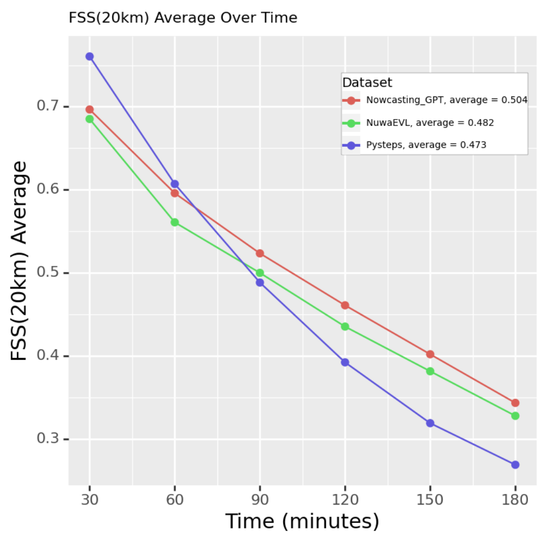
at a 20 km scale, marking the highest duration of skillfulness across the examined lead times for any model (7.6).



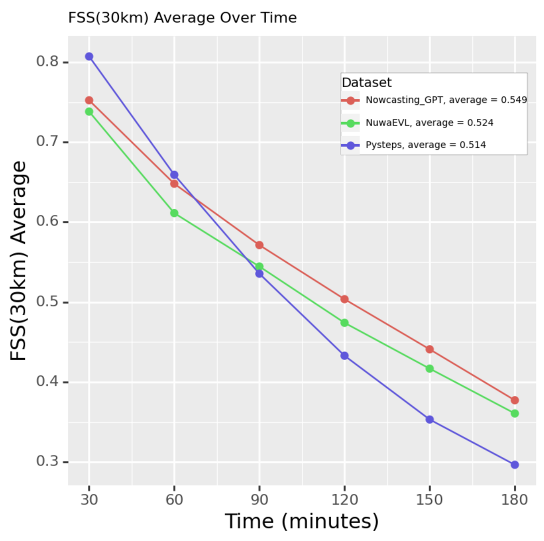
(a) FSS(1km)



(b) FSS(10km)



(c) FSS(20km)



(d) FSS(30km)

Figure 7.6: FSS metric evaluation over the 6 lead times at different spatial resolution. Each point represents the average value for a specific lead time over the whole dataset. Higher value indicates better performance.

The variation in performance across all models, thresholds, and scales can be attributed to several factors, including the models' ability to integrate and analyze spatial and temporal data, their sensitivity to initial conditions, and the underlying algorithms used to predict the movement and development of weather patterns. The superior performance of PySTEPS at a 30-minute lead time suggests its effective utilization of short-term data and initial conditions. Meanwhile, Nowcasting-GPT's strength at longer lead times across various scales highlights its robustness in capturing and predicting the evolution of weather patterns over extended periods.

Conclusion
In this section, we evaluated Nowcasting-GPT against PySTEPS and Nuwa-Pytorch using various metrics across the Netherlands. Nowcasting-GPT showed superior performance with higher PCC values and lower MAE, indicating closer alignment with actual data and more accurate precipitation forecasts. It outperformed the other models in detecting light and moderate rainfall events, especially at longer lead times. FAR analysis revealed that Nowcasting-GPT provided more reliable forecasts at 1mm and 2mm thresholds with fewer false alarms, although PySTEPS had a lower FAR at 1mm but a reduced CSI. FSS analysis showed PySTEPS's skillfulness at a 30-minute lead time, while Nowcasting-GPT excelled at longer lead times, demonstrating robustness in extended weather pattern prediction. Overall, Nowcasting-GPT offers accurate and reliable forecasts, particularly for longer lead times and moderate rainfall events, making it a valuable tool for weather forecasting in the Netherlands.

7.2. Nowcasting-GPT Performance at the Catchment Level

7.2.1. Effect of Post-processing and Ensemble

Chen et al. [23] developed a post-processing technique to enhance the visibility of high-intensity precipitation pixels on radar maps from the RT-dataset, which typically do not display such pixels prominently. This method modifies the initial predictions (RP) to increase their accuracy, particularly for intense precipitation events, applying the formula:

$$TP[i][j] = \left(1 + a\left(\frac{RP[i][j]}{\max(RP)}\right)^{b}\right) \times RP[i][j] \tag{7.1}$$

where TP and RP represent the post-processed and the original unprocessed predictions, respectively. The indices (i, j) pinpoint the pixel locations on the prediction maps. The parameters a and b, fine-tuned to maximize the Gilbert Skills Score (GSS) on a validation dataset comprising 357 extreme events, were set to 0.66 and 0.81, respectively, following the approach of Bi et al [3].
This enhancement not only increases the detection rate of high-intensity precipitation but also raises the false alarm rate. To mitigate this, it is typically used in conjunction with an ensemble technique to balance sensitivity and accuracy.
To further refine our analysis, we explored a range of ensemble sizes at the catchment level, specifically 3, 5, 7, 10, 12, and 14. We utilized Receiver Operating Characteristic (ROC) and Precision-Recall (PRC) curves to assess how these sizes impact model performance, with predictions generated through the optimal configuration of

the transformer for each size. Analysis of the ROC curves (shown in fig: 7.7a) revealed a significant convergence among the ensemble sizes of 10, 12, and 14, with minimal performance distinctions evident among them. This observation was clarified by cropping the ROC curve (fig: 7.7b), which emphasized the negligible differences particularly between sizes 12 and 14.

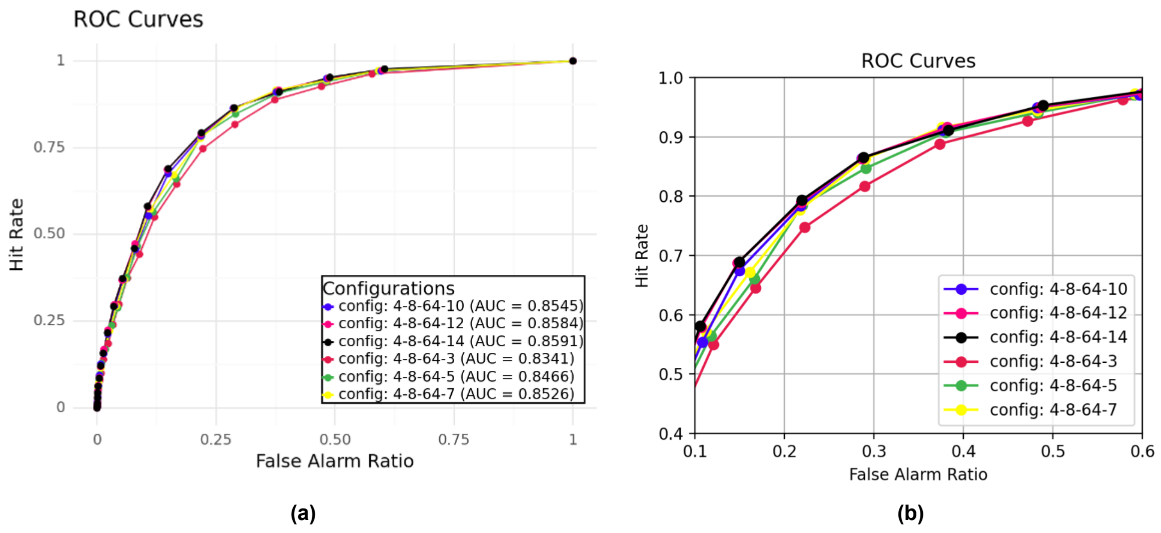


Figure 7.7: ROC Curve Analysis for 3-Hour Extreme Event Detection Across Varying Precipitation Thresholds at Different Ensemble Sizes (a) Complete ROC Curve Showing Thresholds from 10mm/3hrs to 0.5mm/3hrs. (b) Cropped ROC Curve with HR between 0.1 and 0.6 and FAR between 0.4 and 1.

The PRC analysis (fig: 7.8a) mirrored this pattern, showing overlapping performances for ensemble sizes 12 and 14. A focused examination (fig: 7.8b of the PRC curves between hit rates of 0.5 and 0.8 showed that although performance was comparable, ensemble size 12 consistently edged out size 14 in terms of effectiveness. Consequently, ensemble size 12 was selected for Nowcasting GPT in subsequent analyses. For PySTEPS [6], the ensemble size was set at 20, while for Nuwa-Pytorch, it was set at 5, following the specifications established by Bi et al. [3].

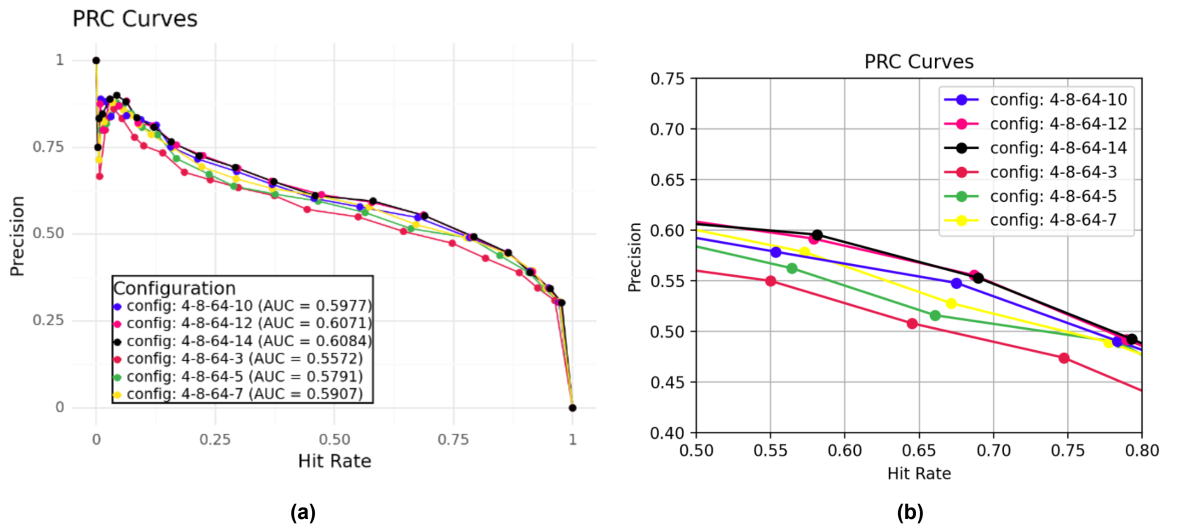


Figure 7.8: PRC Curve Analysis for 3-Hour Extreme Event Detection Across Varying Precipitation Thresholds at Different Ensemble Sizes (a) Complete PRC Curve Showing Thresholds from 10mm/3hrs to 0.5mm/3hrs. (b) Cropped PRC Curve with HR between 0.5 and 0.8 and FAR between 0.4 and 0.75.

This methodical selection of ensemble sizes ensures a balanced approach to enhancing both sensitivity and specificity, optimizing the predictive accuracy of the models for practical applications.

7.2.2. Performance Comparison of Predictive Models in Detecting Extreme Events

To evaluate the ability of the Nowcasting-GPT model to predict rainfall precipitation at catchment levels, we compared it with Nuwa-Pytorch and PySTEPS. Selected sections from predicted precipitation maps were isolated to focus on specific catchment areas. The average predicted precipitation for each catchment was calculated based on six predicted radar maps $(X_{\text{pre+30}}, \ldots, X_{\text{pre+180}})$, averaged over 3 hours. The ground truth threshold was set at $5mm/3hrs$, an extreme rainfall threshold based on the top 1% of historical data. Predicted precipitation thresholds varied from $10mm/3hrs$ to $0.5mm/3hrs$, yielding 20 data points.

Model performance was evaluated based on four outcomes: True Positive (TP), False Negative (FN), False Positive (FP), and True Negative (TN). For a detailed methodology explanation, see section 6.2 in Chapter 6. These outcomes facilitated the plotting of Receiver Operating Characteristic (ROC) curves and Precision-Recall curves, comparing the performance of Nowcasting-GPT, PySTEPS, and Nuwa-Pytorch.

ROC CURVE

The ROC curve is instrumental in assessing predictive models in binary classification tasks such as extreme event detection. This curve plots the true positive rate (Hit Rate) against the false positive rate (False Alarm Ratio), emphasizing the trade-offs between correctly identifying positive events and the consequences of false alarms.In figure 7.9, Nowcasting-GPT, with an AUC of 0.8584, significantly outperforms Nuwa-Pytorch and PySTEPS.

Notably, Nowcasting-GPT maintains a hit rate above 0.75 for five out of twenty thresholds, as detailed in Figure 7.9b. This consistent performance indicates an effective balance of sensitivity and specificity, essential in settings where precise detection of extreme events is crucial. In contrast, both Nuwa-Pytorch and PySTEPS show lower hit rates at these thresholds, indicating a weaker ability to distinguish between true events and false alarms.

As the false alarm ratio approaches 0.5, Nowcasting-GPT's superior capability becomes increasingly apparent. The graphical representation in Figure 7.9a illustrates that its curve remains consistently higher and closer to the top-left corner, indicating optimal performance in predicting true events without increasing false alarms. This robust capability underscores its reliability for critical applications requiring precise event detection.

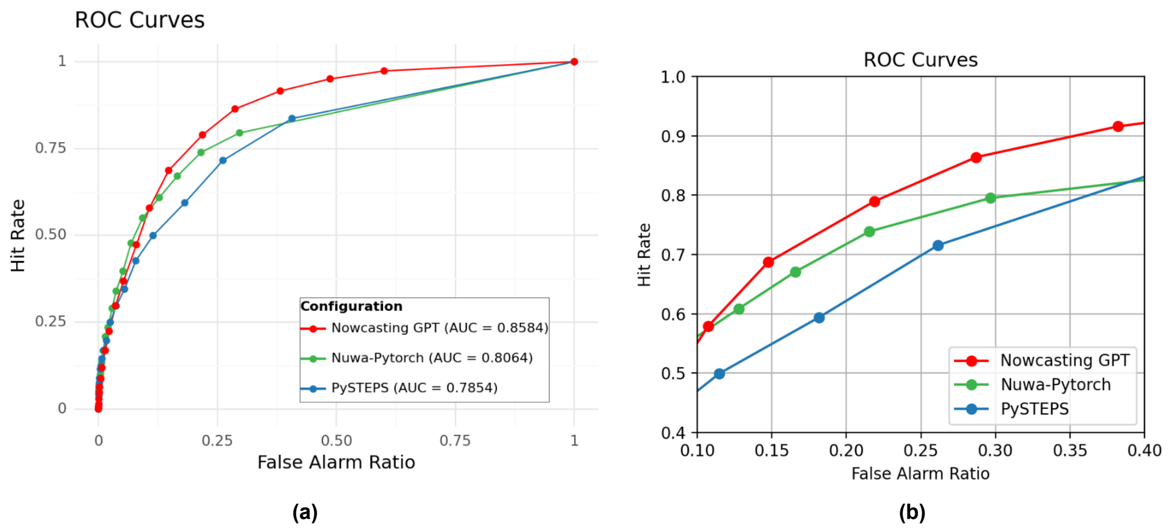


Figure 7.9: (a) The complete ROC curve for 3-hour extreme event detection, the points on the curve (from left to right) represent thresholds from 10mm to 0.5mm. (b) Cropping of the ROC curve by limiting the hit rate to be higher than 0.4 and false alarm rate lower than 0.4.

PRC CURVE

The Precision-Recall Curve (PRC) is crucial for models operating in scenarios with imbalanced classes, as it focuses on the balance between precision and recall. This graph evaluates how effectively a model can detect relevant events (high recall) while maintaining accuracy (high precision). In figure 7.10, Nowcasting-GPT achieves the highest AUC of 0.6071, distinctly outperforming both Nuwa-Pytorch and PySTEPS,

underscoring its superior capability in managing the precision-recall trade-off.
At a detailed level, fig: 7.10b shows a clear decline in precision as the hit rate increases. This decreasing trend in precision with increasing hit rates is typical, indicating a trade-off between recall and precision. Despite this trend, the Nowcasting GPT model maintains the highest precision across various hit rates compared to the other models. Specifically, Nowcasting GPT maintains a precision approximately 18-22% higher than Pysteps and about 9-11% higher than Nuwa-Pytorch across various hit rates.
The gradual decline in precision for Nowcasting-GPT as recall increases further demonstrates its robustness. Unlike the steeper declines observed for Nuwa-Pytorch and PySTEPS, Nowcasting-GPT's precision decreases more slowly, ensuring reliable event detection across a broader range of operational conditions (fig: 7.10a). This consistent performance across varying thresholds makes Nowcasting-GPT exceptionally valuable in practical applications, particularly in settings where maintaining the accuracy of detected events is as crucial as increasing the number of detections.

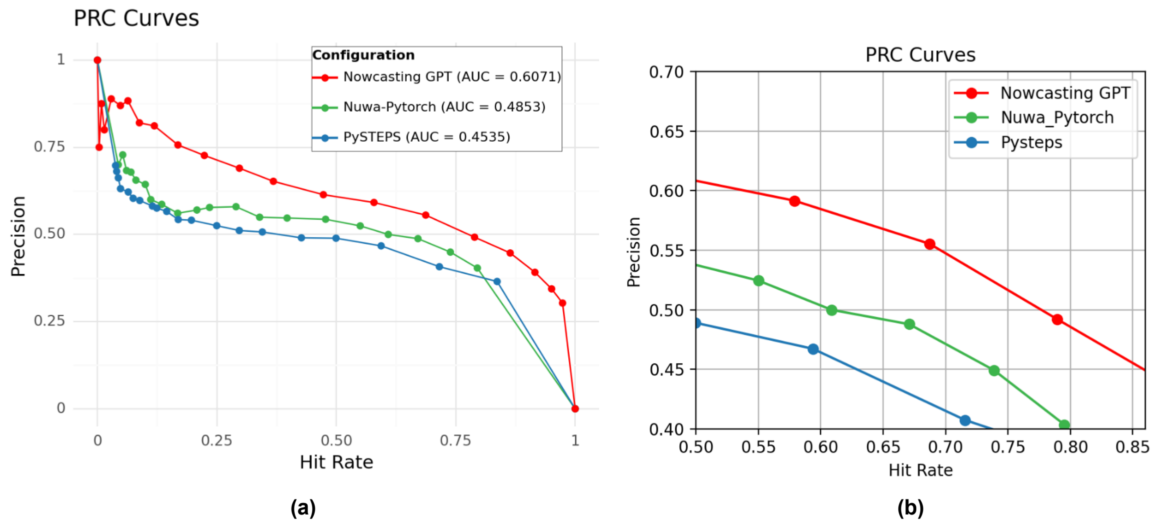


Figure 7.10: (a) The complete PRC curve for 3-hour extreme event detection, the points on the curve (from left to right) represent thresholds from 10mm to 0.5mm. (b) Cropping the PRC curve by limiting the hit rate to be lower than 0.25 and precision higher than 0.5.

7.2.3. Performance Comparison of Predictive Models in Detecting Moderate Events

The analysis of Receiver Operating Characteristic (ROC) and Precision-Recall (PRC) curves for moderate rainfall thresholds ($2mm/3hrs$) employs methodologies similar to those used for extreme event detection, albeit with a shift in contextual focus. This analysis transitions from the realm of extreme rainfall events to moderate ones, which occur more frequently and necessitate a distinct approach to predictive accuracy. This adjustment emphasizes not only the models' detection capabilities but also their consistency and precision in forecasting more common, moderate rainfall events, ensuring that the methodologies are adapted to reflect the unique challenges posed by different levels of rainfall severity.

Nowcasting-GPT distinguishes itself by starting with a higher true positive rate at lower false alarm ratios and consistently maintaining this lead across the full spectrum of false alarm ratios depicted in the ROC curve depicted in fig: 7.11a. This performance consistency, evidenced by a high AUC of 0.8862, underscores its superior effectiveness in event detection at various thresholds. Such a profile suggests that Nowcasting-GPT is exceptionally adept at identifying true events across a wide range of operational conditions, providing reliable outputs that are crucial for effective decision-making.
In contrast, Nuwa-Pytorch and PySTEPS exhibit capabilities that are comparable at lower false alarm ratios, yet neither model achieves as high true positive rates as Nowcasting-GPT at similar false alarm levels (fig: 7.11b). This relative under performance may reflect a less robust system in scenarios demanding a precise balance between detecting true events and minimizing false alarms. Additionally, the ideal point on an ROC curve, located at the top-left corner representing a perfect true positive rate with no false positives, is approached more closely by Nowcasting-GPT than by Nuwa-Pytorch and PySTEPS. This proximity to the ideal point further demonstrates that Nowcasting-GPT more closely aligns with the optimal performance in event detection.

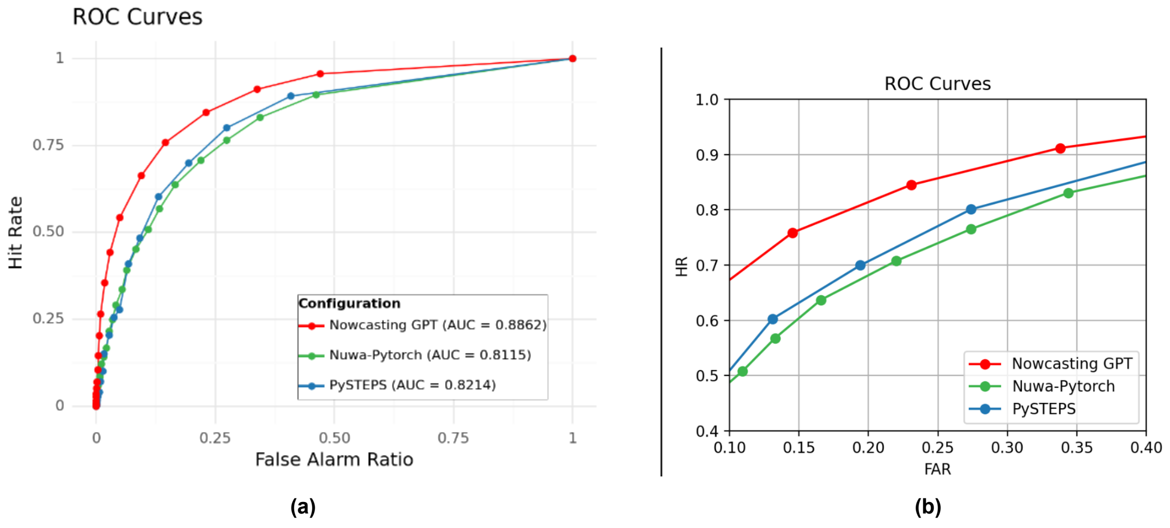
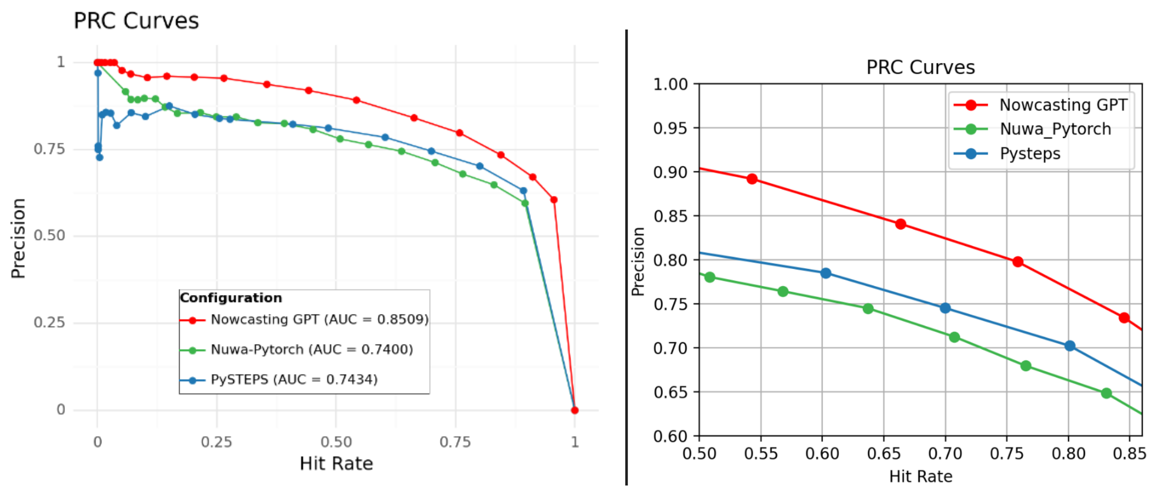


Figure 7.11: (a) The complete ROC curve for 3-hour moderate rainfall event detection, the points on the curve (from left to right) represent thresholds from 10mm to 0.5mm. (b) Cropping of the ROC curve by limiting the False Alarm to be between 0.1 and 0.4 and Hit Rate higher between 0.4 and 1.

From the comprehensive analysis of the PRC data, it is evident that Nowcasting-GPT significantly surpass its competitors with an Area Under the Curve (AUC) of 0.8509, which is markedly higher than those of Nuwa-Pytorch and PySTEPS (fig: 7.12a). This superior AUC reflects Nowcasting-GPT's enhanced ability to effectively discriminate between positive and negative classifications across various levels of recall. Notably, Nowcasting GPT maintains a precision approximately 7-10% higher than Pysteps and about 14-16% higher than Nuwa-Pytorch across hit rates ranging between 0.5 and 0.85.

Furthermore, Nowcasting-GPT exhibits remarkable stability in its precision, maintaining levels at 0.80 even as recall increases to 0.75. In contrast, both Nuwa-Pytorch and PySTEPS experience steeper declines in precision as recall increases. Those results are illustrated in fig: 7.12b. The consistent precision of Nowcasting-GPT, particularly in scenarios involving moderate rainfall, underscores its robust performance and affirms the model's reliability in accurately forecasting moderate rainfall events—crucial for applications that rely on consistent and dependable weather predictions.



(a) **(b)**

Figure 7.12: (a) The complete PRC curve for 3-hour moderate rainfall event detection, the points on the curve (from left to right) represent thresholds from 10mm to 0.5mm. (b) Cropping of the PRC curve by limiting the hit rate to be higher than 0.5 and precision higher than 0.6.

Conclusion

The detailed analyses of the ROC and PRC curves distinctly affirm the superiority of Nowcasting-GPT in extreme event detection, notably in predicting rainfall precipitation at catchment levels. Although Nuwa-Pytorch is specifically biased towards detecting extreme tokens through the integration of EVL with the transformer, Nowcasting-GPT consistently maintains high precision, even as recall increases, far surpassing both Nuwa-Pytorch and PySTEPS. This exemplary balance of precision and recall establishes Nowcasting-GPT as the most reliable model for applications that demand minimal false positives while ensuring no true positive events are missed.

8 Conclusion and Future Work

This thesis has introduced and scrutinized the Nowcasting-GPT model, an innovative approach to precipitation nowcasting that integrates a Vector Quantized Variational Autoencoder (VQ-VAE) with an Auto-regressive (AR) Transformer. The research has demonstrated significant advancements in forecasting accuracy and computational efficiency, providing a robust framework for predicting both moderate and extreme rainfall events.

The AR Transformer component of Nowcasting-GPT has shown marked superiority over traditional models such as ConvLSTM and RNN. Its proficiency in capturing long-range dependencies within weather data allows for accurate forecasts over extended periods, a crucial advantage in meteorological applications. However, this capability comes at the cost of increased computational demand due to the quadratic complexity of the attention mechanism. This necessitates optimized configurations of the model to maintain a balance between computational resources and forecasting efficacy.

Efficiency in training was significantly enhanced through the use of VQ-VAE, inspired by the work of Esser et al. [18] and Bi et al.[3]. By encoding high-resolution radar images into a more manageable lower-dimensional latent space and reducing the spatial resolution of input images from 128x128 to 8x8, the training of the AR Transformer was rendered feasible. This configuration facilitated an effective transformation of compressed latent representations back into detailed precipitation maps using a pre-trained decoder, showcasing an efficient use of computational resources.

Furthermore, the thesis explored various configurations of the transformer to determine the most effective setup. This exploration revealed that the optimal configuration significantly impacts the model's performance, with the best settings focusing on pertinent features rather than redundant ones, thus preventing computational waste and enhancing the accuracy of weather feature capture.

At a national level, the comprehensive performance analysis across the Netherlands demonstrated that Nowcasting-GPT consistently outperformed other models in terms of Pearson Correlation Coefficient (PCC) and Mean Absolute Error (MAE), particularly over longer forecast horizons. This highlights its robustness in operational settings, where it effectively detected both moderate and extreme rainfall events with remarkable precision. The application of the Fractional Skill Score (FSS) further illustrated its capability to capture the spatial distribution of rainfall accurately.
Catchment-level performance evaluation revealed that Nowcasting-GPT also excelled beyond other advanced models, such as Nuwa-Pytorch, which includes an Extreme Value Loss (EVL) to enhance detection of extreme weather events. Surprisingly, despite lacking a similar specialization, Nowcasting-GPT's performance in detecting rainfall events was superior, suggesting that its architectural integration of VQ-VAE and AR Transformer inherently balances the detection of typical and extreme weather patterns effectively.
These findings not only affirm the efficacy of the Nowcasting-GPT model in weather forecasting but also highlight the potential for further improvements and broader applications. Future research could focus on integrating additional data sources, applying more advanced machine learning techniques, and expanding the model's application to other geographical regions and climatic conditions. The continued development and adaptation of this model are poised to significantly advance our capabilities in weather prediction and analysis, with broad implications for environmental science, disaster management, agriculture, and beyond.

Future Work
A significant challenge highlighted by this research is the highly imbalanced nature of the dataset, notably concerning the infrequency of extreme precipitation events which are critical for detecting hazardous weather conditions like floods. This scarcity leads to an under-representation of extreme weather tokens, impacting the model's ability to forecast severe weather conditions effectively. Future research could address this by exploring advanced methods to manage data imbalance, such as adaptive sampling techniques or specialized loss functions designed to emphasize learning from these rare but significant events.
Further enhancing the model's capabilities, the application of Extreme Value Loss (EVL) as demonstrated by Haoran et al. [3]. shows promise in refining predictions of precipitation maps that include extreme events. By integrating EVL as an additional loss function alongside the transformer loss in Nowcasting-GPT, the model gains a robust regularization mechanism tailored to handle data imbalance. This approach emphasizes extreme events in the loss calculation, improving the model's sensitivity

and accuracy in forecasting severe weather conditions and contributing to a more balanced learning process.
Additionally, the potential integration of a spatial discriminator within the VQ-VAE framework, as discussed by researchers like Esser et al.[12], represents a transformative enhancement. This would involve adapting the adversarial training component to provide dynamic feedback to the VQ-VAE, promoting iterative improvements in the generated outputs. Integrating a spatial discriminator could significantly refine the model's capability to process and reconstruct complex spatial patterns in precipitation data, leading to more detailed and accurate weather forecasts.
An exciting direction for future development is the potential adaptation of the model for video analysis, as suggested by advancements in VideoGPT by Yan et al.[24]. This modification would involve integrating 3D convolutions within the VQ-GAN encoder to capture both spatial features from individual frames and temporal dynamics across sequences. This capability is ideally suited for analyzing radar map data, which inherently forms a time series akin to video sequences. Enhancing the model to better process these temporal relationships could lead to a more nuanced understanding and prediction of precipitation patterns over time.
Lastly, embedding physical constraints into the modeling process could profoundly increase the predictive accuracy of the model. By incorporating established physical laws and atmospheric dynamics into the training process, the model could yield forecasts that are not only more precise but also align with theoretical meteorological principles [25]. This hybrid approach would merge empirical data-driven techniques with traditional physically-based forecasting methods, potentially revolutionizing weather prediction methodologies.

References

[1] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, *Convolutional lstm network: A machine learning approach for precipitation nowcasting*, 2015. arXiv: `1506.04214 [cs.CV]`.

[2] S. Ravuri, K. Lenc, M. Willson, *et al.*, “Skilful precipitation nowcasting using deep generative models of radar,” *Nature*, vol. 597, no. 7878, pp. 672–677, Sep. 2021, ISSN: 1476-4687. DOI: `10.1038/s41586-021-03854-z`. [Online]. Available: `http://dx.doi.org/10.1038/s41586-021-03854-z`.

[3] H. Bi, M. Kyryliuk, Z. Wang, *et al.*, “Nowcasting of extreme precipitation using deep generative models,” in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5. DOI: `10.1109/ICASSP49357.2023.10094988`.

[4] R. Prudden, S. Adams, D. Kangin, *et al.*, *A review of radar-based nowcasting of precipitation and applicable machine learning techniques*, 2020. arXiv: `2005.04988 [physics.ao-ph]`.

[5] U. Germann and I. Zawadzki, “Scale-dependence of the predictability of precipitation from continental radar images. part i: Description of the methodology,” *Monthly Weather Review - MON WEATHER REV*, vol. 130, Dec. 2002. DOI: `10.1175/1520-0493(2002)130<2859:SDOTPO>2.0.CO;2`.

[6] S. Pulkkinen, D. Nerini, A. A. Pérez Hortal, *et al.*, “Pysteps: An open-source python library for probabilistic precipitation nowcasting (v1.0),” *Geoscientific Model Development*, vol. 12, no. 10, pp. 4185–4219, 2019. DOI: `10.5194/gmd-12-4185-2019`. [Online]. Available: `https://gmd.copernicus.org/articles/12/4185/2019/`.

[7] X. Shi, Z. Gao, L. Lausen, *et al.*, *Deep learning for precipitation nowcasting: A benchmark and a new model*, 2017. arXiv: `1706.03458 [cs.CV]`.

[8] K. Trebing, T. Stanczyk, and S. Mehrkanoon, *Smaat-unet: Precipitation nowcasting using a small attention-unet architecture*, 2021. arXiv: `2007.04417 [cs.LG]`.

[9] A. Bojesomo, H. A. Marzouqi, and P. Liatsis, *A novel transformer network with shifted window cross-attention for spatiotemporal weather forecasting*, 2022. arXiv: `2208.01252 [cs.CV]`.

[10] X. Shi and D.-Y. Yeung, *Machine learning for spatiotemporal sequence forecasting: A survey*, 2018. arXiv: 1808.06865 [cs.LG].

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, *Generative adversarial networks*, 2014. arXiv: 1406.2661 [stat.ML].

[12] L. Tian, X. Li, Y. Ye, P. Xie, and Y. Li, "A generative adversarial gated recurrent unit model for precipitation nowcasting," *IEEE Geoscience and Remote Sensing Letters*, vol. 17, pp. 601–605, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:201248197.

[13] J. R. Jing, Q. Li, X. Y. Ding, N. L. Sun, R. Tang, and Y. L. Cai, "Aenn: A generative adversarial neural network for weather radar echo extrapolation," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-3/W9, pp. 89–94, 2019. DOI: 10.5194/isprs-archives-XLII-3-W9-89-2019. [Online]. Available: https://isprs-archives.copernicus.org/articles/XLII-3-W9/89/2019/.

[14] T. Nguyen, J. Brandstetter, A. Kapoor, J. K. Gupta, and A. Grover, *Climax: A foundation model for weather and climate*, 2023. arXiv: 2301.10343 [cs.LG].

[15] H. Beekhuis and I. Holleman, "From pulse to product: Highlights of the digital-if upgrade of the dutch national radar network," in *Proceedings of ERAD 5*, Helsinki, 2008, pp. 0–0.

[16] A. Best, "The size distribution of raindrops," *Quarterly Journal of the Royal Meteorological Society*, vol. 76, pp. 16–36, 327 1950. DOI: 10.1002/qj.49707632704.

[17] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, *Neural discrete representation learning*, 2018. arXiv: 1711.00937 [cs.LG].

[18] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," *CoRR*, vol. abs/2012.09841, 2020. arXiv: 2012.09841. [Online]. Available: https://arxiv.org/abs/2012.09841.

[19] V. Micheli, E. Alonso, and F. Fleuret, *Transformers are sample-efficient world models*, 2023. arXiv: 2209.00588 [cs.LG].

[20] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," *CoRR*, vol. abs/1603.08155, 2016. arXiv: 1603.08155. [Online]. Available: http://arxiv.org/abs/1603.08155.

[21] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2023. arXiv: 1706.03762 [cs.CL].

[22] K. Cho, B. van Merrienboer, C. Gulcehre, *et al.*, *Learning phrase representations using rnn encoder-decoder for statistical machine translation*, 2014. arXiv: 1406.1078 [cs.CL].

[23] G. Chen and W. Wang, "Short-term precipitation prediction using deep learning," *CoRR*, vol. abs/2110.01843, 2021. arXiv: 2110.01843. [Online]. Available: https://arxiv.org/abs/2110.01843.

[24] W. Yan, Y. Zhang, P. Abbeel, and A. Srinivas, "Videogpt: Video generation using VQ-VAE and transformers," *CoRR*, vol. abs/2104.10157, 2021. arXiv: 2104.10157. [Online]. Available: https://arxiv.org/abs/2104.10157.

[25] K. Kashinath, M. Mustafa, A. Albert, *et al.*, "Physics-informed machine learning: Case studies for weather and climate modelling," *Philosophical Transactions of the Royal Society A*, vol. 379, p. 20 200 093, 2021. DOI: 10.1098/rsta.2020.0093. [Online]. Available: http://doi.org/10.1098/rsta.2020.0093.

A

Reconstructions Examples

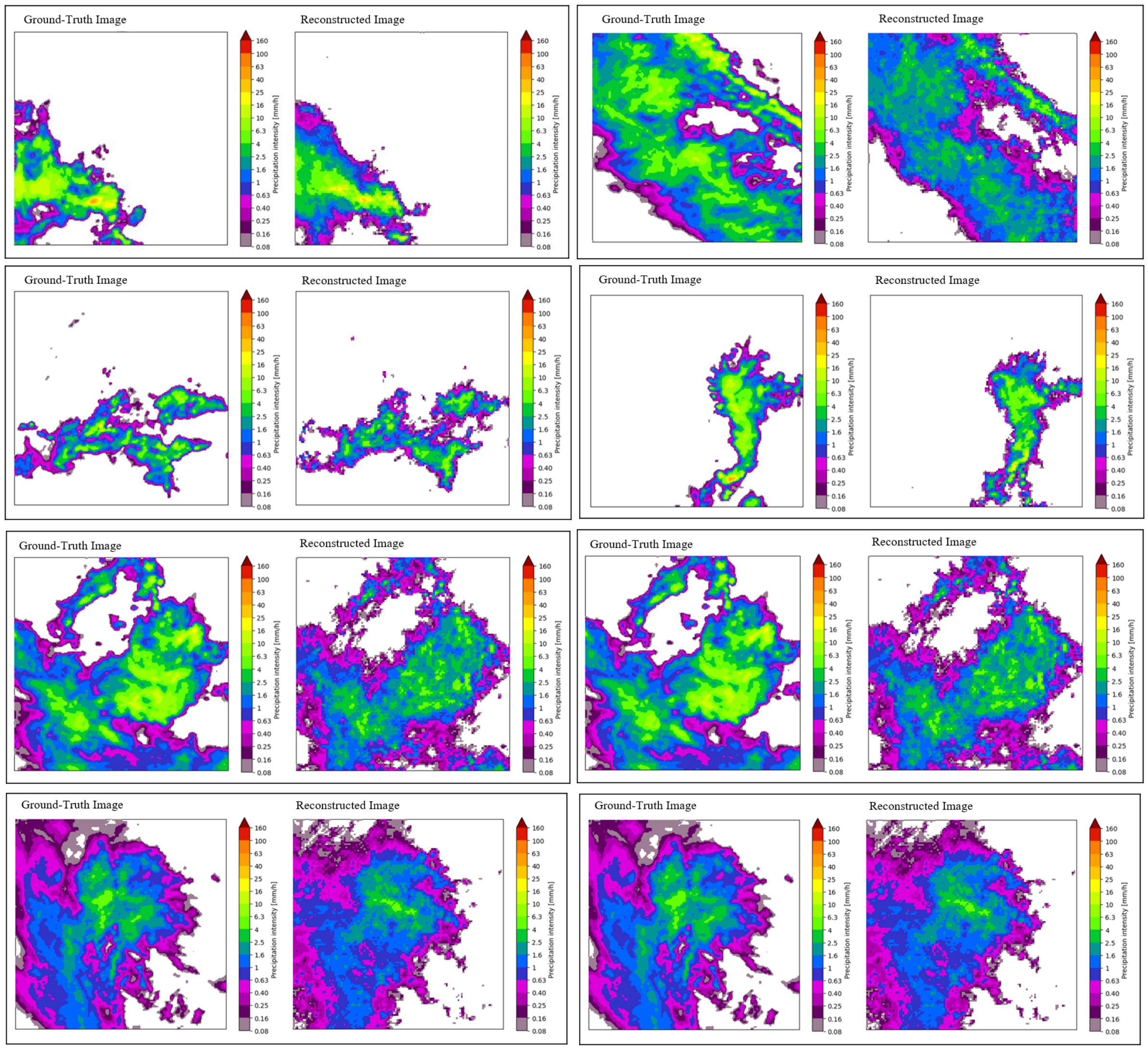


Figure A.1: Example of reconstruction of precipitation fields by the VQ-VAE. The left column shows the original radar input images, and the right column shows the reconstructed precipitation fields.

B

Comparison between Nowcasting Predictions of different models

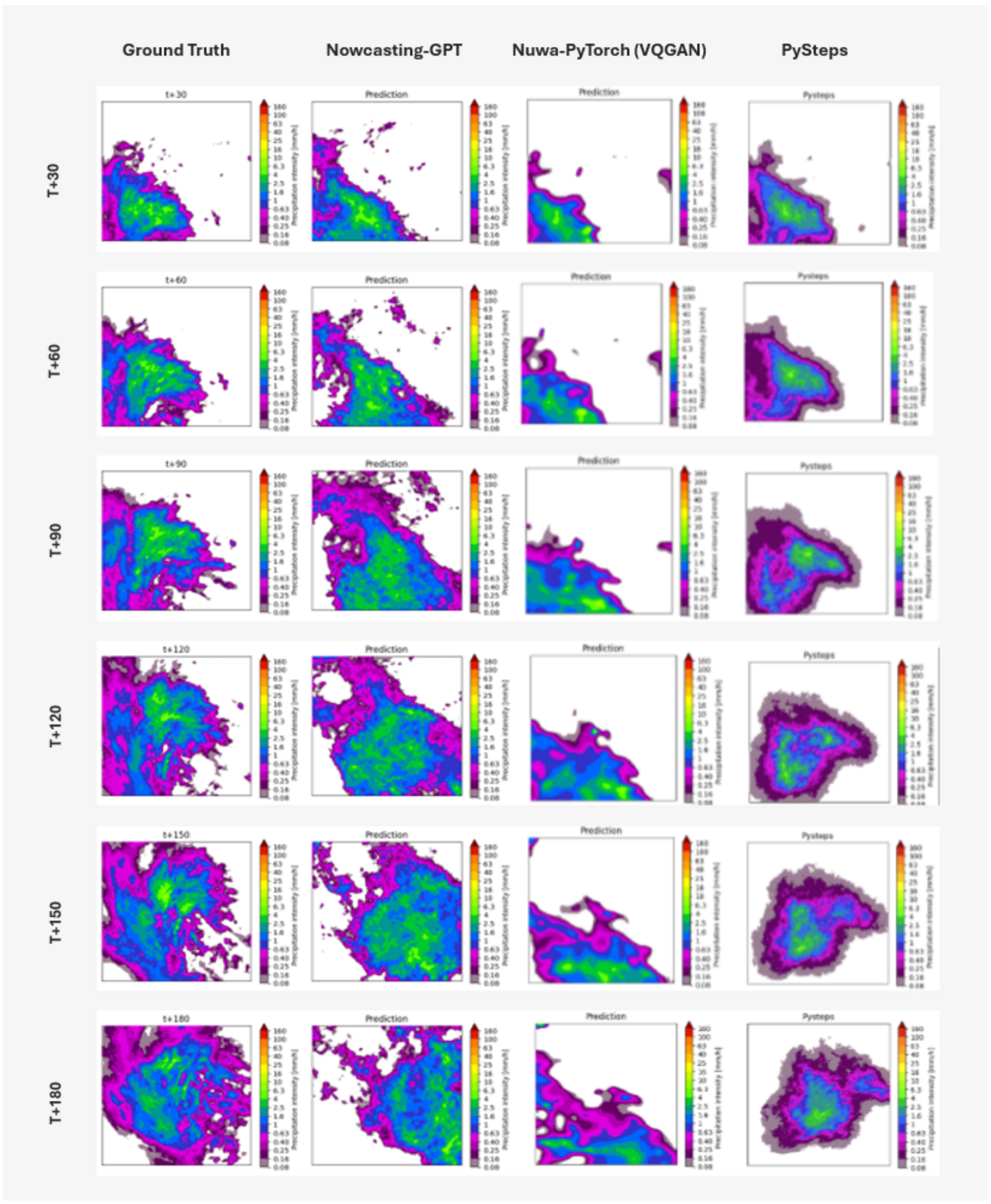


Figure B.1: Comparison of Nowcasting results over the entire Netherlands region by different models (t = 03:00 2019/03/07)

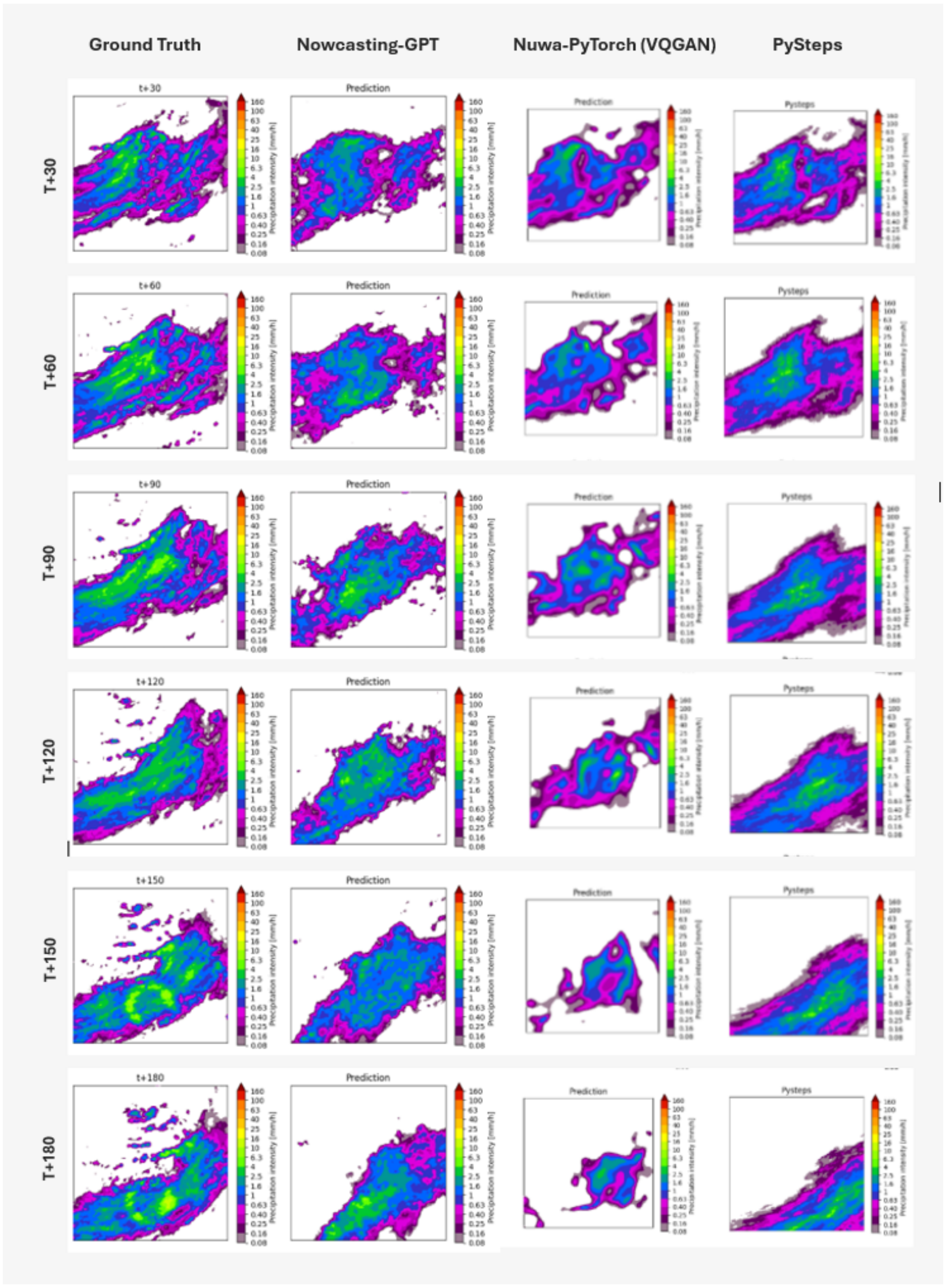


Figure B.2: Comparison of Nowcasting results over the entire Netherlands region by different models (t = 05:45 2019/11/28).

C
Comparison of Number of Parameters, Training and Generation time

	Nuwa-PyTorch (VQGAN)	Nowcasting-GPT	PySTEPS
Number of parameters	772,832 M	704,259 M	–
Training time	672 h	**210 h**	–
Generation time	322.86 s	38.90 s	**5.5 s**

Table C.1: Comparison of training time, generation time, and the number of parameters for different models.

In table C.1, it is evident that Nowcasting-GPT has the fastest generation time, followed by PySTEPS. In contrast, the generation time for Nuwa-PyTorch (VQGAN) is significantly higher than the others. One key reason for Nowcasting-GPT's superior performance is the implementation of KV-caching, which speeds up the generation of tokens in the autoregressive transformer, thereby enhancing the overall generation time for prediction frames.

Additionally, to facilitate efficient training of the deep learning model, input radar maps for all datasets have been converted into NumPy arrays. This optimization, along with the best transformer configuration, significantly contributes to the improved training time compared to Nuwa-PyTorch (VQGAN), as highlighted in the table.