Forecasting Height of High-Frequency Sea Level Oscillations in Bakar Bay Using Deep Learning

Metličić, Nikola

University of Split
Faculty of Science

Forecasting Height of High-Frequency Sea Level Oscillations in Bakar Bay Using Deep Learning

Master thesis

Nikola Metličić

Split, September 2024

Temeljna dokumentacijska kartica

Sveučilište u Splitu
Prirodoslovno–matematički fakultet
Odjel za fiziku
Ruđera Boškovića 33, 21000 Split, Hrvatska

Diplomski rad

Prognoziranje visine visoko-frekventnih oscilacija razine mora u Bakarskom zaljevu primjenom dubokog učenja

Nikola Metličić

Sveučilišni diplomski studij Fizika; smjer: Fizika okoliša

Sažetak:

Značajne oscilacije razine mora mogu predstavljati ozbiljnu prijetnju obalnim ekosustavima i zajednicama. Visokofrekventna komponenta ovih oscilacija, koja se javlja na periodima od nekoliko minuta do nekoliko sati i koja je vođena atmosferskim forsiranjima, značajno doprinosi ekstremnim razinama mora. Ovaj rad istražuje primjenu dubokog učenja na prognoziranje visina visokofrekventnih oscilacija razine mora ($T < 1$ h) u Bakarskom zaljevu. Koristeći podatke mareografa i ERA5 reanalize razvijena je konvolucijska neuronska mreža za prognoziranje dnevnih maksimuma ovih oscilacija do 72 sata unaprijed. Model je pokazao visoku točnost za neekstremne događaje, ali smanjenu učinkovitost za ekstremne događaje, posebno za duže intervale prognoza. Točno prognoziranje ekstremnih događaja ostaje izazov. U radu se ispituje i važnost komponenti modela kroz studije ablacije i alternativnih konfiguracija treniranja. Ovaj rad doprinosi razvoju točnijih i pouzdanijih modela prognoziranja, koji su ključni za smanjenje rizika od obalnih poplava i povećanje otpornosti ugroženih zajednica. Buduća poboljšanja mogla bi uključivati integraciju atmosferskih podataka finije rezolucije i proširenje vremenskog raspona ulaznih podataka kako bi se poboljšala učinkovitost modela za prognoziranje rijetkih ekstremnih događaja.

Ključne riječi: visoko-frekventne oscilacije razine mora, Bakarski zaljev, duboko učenje, prognoziranje razine mora, obalno plavljenje, ERA5 reanaliza, seši

Rad sadrži: 52 stranice, 23 slike, 5 tablica, 68 literaturnih navoda. Izvornik je na engleskom jeziku.

Mentor: izv. prof. dr. sc. Jadranka Šepić

Komentor: Dr. Matjaž Ličer

Ocjenjivači: izv. prof. dr. sc. Jadranka Šepić,
Dr. Matjaž Ličer,
Krešimir Ruić, mag. phys.

Rad prihvaćen: 30. 8. 2024.

Rad je pohranjen u Knjižnici Prirodoslovno–matematičkog fakulteta, Sveučilišta u Splitu.

Basic documentation card

University of Split
Faculty of Science
Department of Physics
Ruđera Boškovića 33, 21000 Split, Croatia

Master thesis

Forecasting Height of High-Frequency Sea Level Oscillations in Bakar Bay Using Deep Learning

Nikola Metličić

University graduate study Physics, specialization in Environmental Physics

Abstract:
Significant sea level oscillations can pose severe threats to coastal ecosystems and communities. The high-frequency component of these oscillations, occurring over minutes to hours and driven by atmospheric forces, contributes notably to extreme sea levels. This thesis explores the use of deep learning for forecasting the heights of high-frequency ($T < 1$ h) sea level oscillations in Bakar Bay. Utilizing tide gauge records and ERA5 reanalysis data, a convolutional neural network was developed to forecast the daily maxima of these oscillations up to 72 hours in advance. The model demonstrated high accuracy for non-extreme events but showed decreased performance for extreme events, particularly with longer forecast lead times. Forecasting extreme events accurately remains a challenge. The thesis also examines the importance of model components through ablation studies and alternative training configurations. This work contributes to the development of more accurate and reliable forecasting models, essential for mitigating coastal flooding risks and supporting the resilience of vulnerable communities. Future improvements could involve integrating finer-resolution atmospheric data and extending the input data time range to enhance the model's performance for rare extreme events.

Keywords: high-frequency sea level oscillations, Bakar Bay, deep learning, sea level forecasting, coastal flooding, ERA5 reanalysis, seiches

Thesis consists of: 52 pages, 23 figures, 5 tables, 68 references. Original language: English.

Co-mentor: Assoc. Prof. Dr. Jadranka Šepić

Leader: Dr. Matjaž Ličer

Reviewers: Assoc. Prof. Dr. Jadranka Šepić,
Dr. Matjaž Ličer,
Krešimir Ruić, MSc. Phys.

Thesis accepted: August 30th, 2024

Thesis is deposited in the library of the Faculty of Science, University of Split.

I want to thank my supervisor, Dr. Matjaž Ličer, for his warm welcome at the Slovenian Environment Agency, as well as for his valuable mentorship and guidance throughout the development of this thesis.

I especially thank my academic mentor Assoc. Prof. Dr. Jadranka Šepić for her continuous support and excellent advice throughout my studies and during the preparation of this thesis.

I am grateful to Marko Rus for his assistance with the model's architecture and for sharing his knowledge of deep learning.

Thank you to Anja Fettich for providing the atmospheric data necessary for this study, and to Iva Međugorac for sharing the tide gauge data from Bakar on behalf of the Department of Geophysics, Faculty of Science, Zagreb.

I also appreciate the support and company of my colleagues at the Slovenian Environment Agency during my Erasmus internship.

I thank my colleagues and professors at the Faculty of Science, University of Split for their help and assistance whenever I needed it.

Finally, I would like to thank my family for always being my support.

Contents

1 Introduction

The rise in global mean sea level, primarily driven by anthropogenic climate change [1], has led to an increase in the frequency and intensity of floods and extreme sea level events, posing significant risks to coastal ecosystems and communities [2][3]. These extremes, which have become more common and severe in recent decades [4][5], can cause hazardous conditions, including damage to infrastructure and loss of life, especially when changes occur rapidly and unexpectedly [3].

Particularly vulnerable are shallow, semi-enclosed basins such as the northern Adriatic Sea, where sea level rise exacerbates the risk of coastal inundation and erosion [6]. In these areas, extreme sea level events, are driven by a variety of atmospheric, oceanic, hydrological, and geologic processes. These processes occur over a wide range of time scales - from minutes to millennia - and across spatial scales that range from localized areas to global extents [7][8].

High-frequency sea level oscillations, occurring on time scales from minutes to hours, also contribute significantly to extreme sea levels. These oscillations include phenomena such as tsunamis [9], atmospherically generated seiches [10], meteotsunamis [11][12], edge waves [13][14], infragravity waves [15][16], and wind-driven waves [17]. Such high-frequency oscillations are particularly relevant in low-tidal basins such as the Mediterranean Sea, where they can reach heights comparable to extreme storm surges, making them a critical component of coastal risk [18]. Consequently, the need for accurate sea level monitoring and forecasting, crucial for mitigating impacts on coastal communities and improving resilience, serves as the motivation for this thesis, which focuses on forecasting the height of high-frequency sea level oscillations.

High-frequency sea level oscillations are primarily driven by intense atmospheric pressure oscillations and wind bursts. Sudden changes in atmospheric pressure over a body of water can generate waves that propagate and amplify, resulting in significant sea level oscillations. Wind, especially when intense, can push water toward the shore, causing temporary sea level rises.

One of the most notable manifestations of high-frequency oscillations are meteotsunamis, long ocean waves generated by atmospheric disturbances. These disturbances are associated with atmospheric gravity waves, convective pressure jumps, frontal lines, and wind gusts [12][19][20][21]. These waves can reach hazardous heights [11][22] at the open coast or within the bays, and are classified as tsunamis based on their spectral properties, heights, and impact [12]. The Adriatic Sea is a recognized meteotsunami „hot spot", where these events are particularly strong [23]. Vilibić and Šepić [18] found a significant positive correlation between high-frequency sea level oscillations and mid-troposphere wind speed, which often fosters the formation of convective cells and internal gravity waves, leading to meteotsunamigenic air pressure jumps [24]. These atmospheric disturbances can generate and significantly amplify long ocean waves through processes such as Proudman and Greenspan resonance, which occur

when the speed of the atmospheric disturbance matches the longwave speed of the ocean waves or the speed of edge waves, respectively [25][26]. Meteotsunamis are further amplified by coastal topography (shoaling [27] and harbour resonance [12]), and in extreme cases, they can reach destructive heights [28]. Harbour resonance occurs when the period of incoming long ocean waves coincides with the period of eigen oscillations of a bay.

In the Mediterranean region, synoptic atmospheric conditions favorable for meteotsunamis are linked to a distinct Rossby wave pattern at approximately 500 hPa. These conditions typically include a surface pressure low located west of the affected area, an inflow of warm, dry air from Africa around 850 hPa, a strong south-westerly wind around 500 hPa, and the presence of unstable atmospheric layers characterized by a Richardson number of less than 0.25 [23].

Richardson number [29][30] is a measure used to evaluate the stability of fluid flows, particularly in the atmosphere or oceans. It indicates the balance between two opposing forces: buoyancy, which tends to suppress turbulence, and wind shear, which tends to generate turbulence. When the Richardson number is low (typically below 0.25), it suggests that the shear forces are strong enough to overcome the stabilizing effect of buoyancy, leading to turbulence and mixing within the fluid. Conversely, a high Richardson number indicates that the buoyant forces are dominant, which suppresses turbulence and results in a more stable, stratified flow. This measure is important in meteorology and oceanography because it helps predict whether air or water will remain stratified or become turbulent.

Local seiches, driven by long ocean waves entering bays, are standing oscillations that can vary significantly depending on the bay's shape, size, and depth [31]. Seiches, characterized by periodic water level rises and falls, can be triggered by atmospheric disturbances, wind, or long ocean waves [12][21]. Their periods, ranging from minutes to several hours, are influenced by the basin's geomorphology, including its length, width, and depth. Seiches can also result from tsunami-generating processes such as earthquakes, landslides, volcanic eruptions, or meteorite impacts [32][33].

Numerous studies, particularly focused on the Mediterranean region, have consistently demonstrated that the most intense atmospherically induced short-period sea level oscillations typically occur under specific synoptic conditions. The pioneering work in this area was conducted by Ramis and Jansà [34] for the Balearic Islands. Their research laid the groundwork for understanding the meteorological factors contributing to these events, leading to the early development of a probabilistic „rissaga" (local name for meteotsunamis) warning system for Ciutadella by 1985 [35]. This system was based on the principle that the probability of a strong „rissaga" event increases with the similarity between forecasted synoptic fields and characteristic synoptic situations associated with past events. These conditions included strong south-westerly winds in the middle and upper atmospheric levels, which played a crucial role in the rapid propagation of mesoscale meteorological disturbances. The phenomenon of Proudman resonance was also recognized as essential for the amplification of marine

responses, leading to pronounced sea level oscillations [35]. Satellite imagery provided further insights, often revealing a recurring pattern, such as a large comma-shaped cloud over North Africa and the Western Mediterranean, associated with mid-level ascents and south-westerly flows. This pattern, along with phenomena such as gravity waves and convective nuclei, has become essential to a conceptual model that helps forecasters predict potential „rissaga" events [35]. An evaluation of this forecasting service in the late 1980s yielded promising results, although variations in terminology and the lack of permanent mareographs in Ciutadella before 2007 complicated comprehensive verification. Early studies indicated that most „rissaga" events were predicted successfully, despite occasional discrepancies in amplitude predictions (both under- and over-predictions) [35].

The application of neural networks, particularly by researchers at the University of the Balearic Islands, has further advanced the prediction of „rissaga" amplitudes. By training these networks on historical data, they have improved the ability to predict future events based on observed or forecasted atmospheric wind-temperature profiles [36].

Šepić et al. [37] investigated the prediction of meteotsunamis through linear correlation techniques, analyzing historical data to identify predictive patterns. They proposed a meteotsunami synoptic index linking short-period sea level oscillations in Ciutadella to specific synoptic variables such as wind speed, direction, temperature and mean sea level pressure gradients, and relative humidity. Although this approach showed potential, it was limited in accurately forecasting the magnitude of meteotsunamis, which requires high-resolution modeling and real-time observations to account for mesoscale atmospheric disturbances.

Bakar Bay in Croatia (location shown in Figure 1) is a pivotal case study for this research, known for its strong seiches. Numerous studies have focused on the seiches in Bakar Bay [38][39][40][41]. Sea level measurements in Bakar, ongoing since 1929, represent the longest oceanographic time series in Croatia. The bay is an elongated basin, 4.5 km long and up to 38 m deep, connected to the Adriatic Sea through a narrow strait (~400 m) [42].

To advance the forecasting capabilities for high-frequency sea level oscillations, this study proposes the use of deep learning techniques. By applying deep learning to high-frequency sea level oscillations, I aim to develop a robust model that can accurately forecast these events, which could provide timely warnings to coastal communities. Other deep learning models have already been developed, such as HIDRA2 [43], for sea level and storm tide forecasting. HIDRA2 is an advanced convolutional neural network specifically designed to predict total sea surface height by integrating atmospheric, tidal, and sea surface height data. This model outperforms previous models in Koper, including its predecessor HIDRA1, and even state-of-the-art numerical ocean models such as NEMO and SCHISM, from June 1st, 2019 to December 31st, 2020 [43].

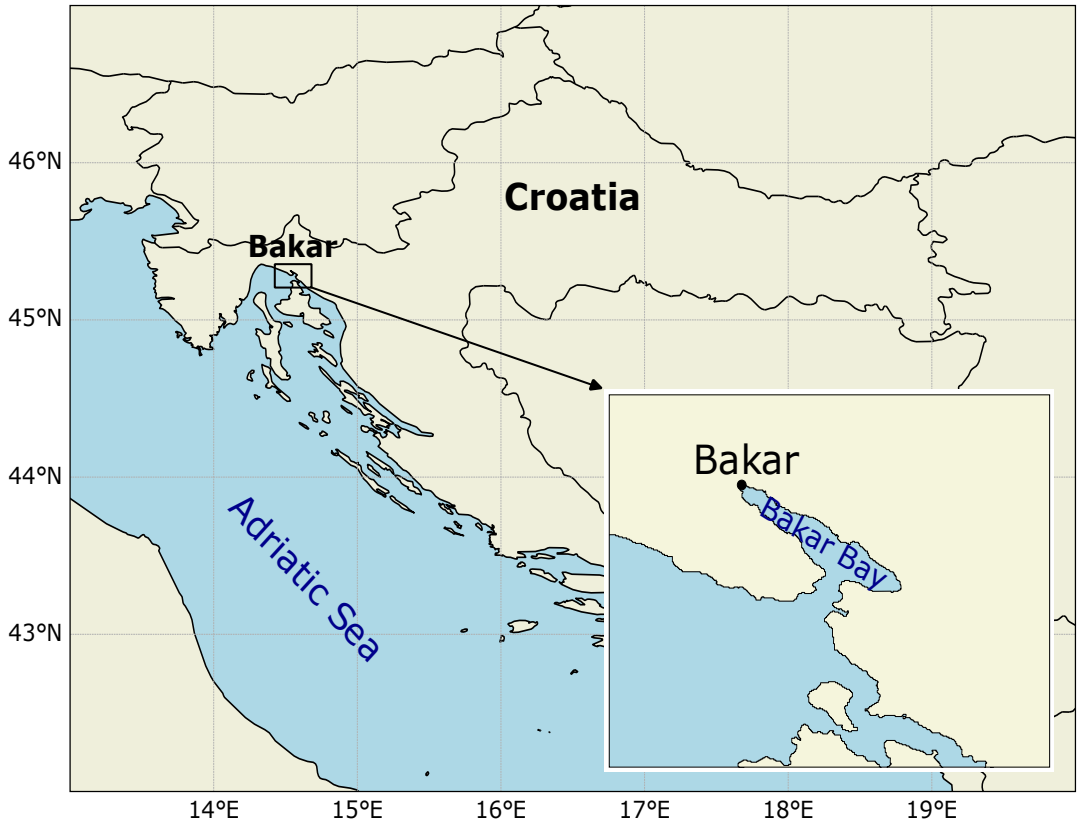

Figure 1: *Map showing the location of Bakar Bay in Croatia.*

This thesis is organized into the following chapters:

- Chapter 2 provides a foundational understanding of deep learning, covering essential concepts such as neural networks, convolutional neural networks (CNNs), forward and backward propagation, and hyperparameters. It sets the stage for the application of these concepts to the problem of sea level forecasting.
- Chapter 3 details the data collection, preprocessing, and modeling techniques used in the study. It includes a description of the sea level and atmospheric data sources, the deep learning model architecture, and the training process. Additionally, it covers the various configurations and experiments conducted to optimize the model's performance.
- Chapter 4 presents the findings of the study, including the performance of the best-performing deep learning model across different forecast lead times, as well as its comparison with other model variants. The results are analyzed in terms of key metrics such as RMSE, MAE, bias, and accuracy, with a particular focus on the model's ability to predict extreme high-frequency sea level oscillations.
- Chapter 5 discusses the implications of the results, the effectiveness of the model components, and the impact of various training configurations. It concludes with recommendations for future research, including potential improvements to the model.

2 Deep Learning: An Overview of Key Concepts

Machine learning (ML) is a branch of artificial intelligence (AI), focusing on developing algorithms that allow machines to learn from data and enhance their performance over time without being explicitly programmed for every task. ML stands out for its ability to learn and improve through experience, offering a more flexible and powerful approach to problem-solving [44].

To address a problem computationally, we typically require an algorithm. An algorithm is a defined set of instructions that transforms input into output. However, explicit algorithms are difficult to derive for many complex tasks. This is where ML becomes valuable - by leveraging data, machines can automatically generate the necessary algorithms through learning, even when explicit solutions are unknown. This algorithm can later be applied to solve problems with new data [45]. This approach is particularly useful in scenarios where human understanding of the process is incomplete, but sufficient data exists to model the underlying patterns [46].

Machine learning algorithms can be broadly categorized as supervised or unsupervised based on the type of experience they are exposed to during the learning process [46]:

- In **supervised learning**, models are trained on a labeled dataset, where the input data is paired with the correct output. The goal is for the model to learn the mapping from inputs to outputs and make predictions on new, unseen data. Common algorithms include regression algorithms, decision trees, support vector machines, and neural networks.
- **Unsupervised learning** involves training models on data that is not labeled, meaning the algorithm must detect patterns and relationships within the data and categorize them accordingly. This approach is often used for clustering, where the goal is to group similar data points together.

In traditional machine learning, the techniques were limited in their ability to process raw data. It must first be transformed into a format that the model can understand. This typically involves designing a feature extractor that converts raw data, such as pixel values in images, into a suitable feature vector. This preprocessing step is crucial for enabling the learning subsystem to detect patterns and make predictions. However, deep learning simplifies this process by automatically learning to extract relevant features from raw data, eliminating the need for manual feature engineering [47].

In recent years, deep learning has achieved remarkable success across various domains, revolutionizing fields such as image or speech recognition and natural language processing [47]. Deep learning is a subfield of machine learning that focuses on neural networks with many layers (hence the term „deep”) to model complex patterns in data. These networks are designed to learn hierarchical representations of data automatically [48], making them particularly powerful for tasks such as, in the context of this thesis, forecasting the height of

high-frequency sea level oscillations. Deep learning models are typically trained using a large amount of labeled data (in the case of supervised learning) and require substantial computational resources.

In this section, I will explore the fundamental concepts of deep learning, including the structure and function of neural networks, the specifics of convolutional neural networks (CNNs), the processes of forward and backward propagation, and key hyperparameters that influence model training and performance.

2.1 Neural Networks

Neural networks are the fundamental building blocks of deep learning, drawing inspiration from the human brain's structure. Neural networks aim to imitate biological neurons, using artificial neurons that operate in a similar way to their biological counterparts. A neural network is composed of layers of nodes (neurons), where each node represents a mathematical operation. The network takes input data, processes it through various layers, and produces an output. The connections between neurons have weights, which are adjusted during the training process to minimize the difference between the predicted and actual output [49].

A neuron is the fundamental unit of a neural network. Each neuron receives one or more inputs, processes them, and produces an output. These inputs are typically real numbers, each associated with a weight that indicates the strength of the connection between neurons. The neuron calculates a weighted sum of its inputs, adds a bias term, and then applies an activation function to produce the final output [49]. This process is illustrated in Figure 2.

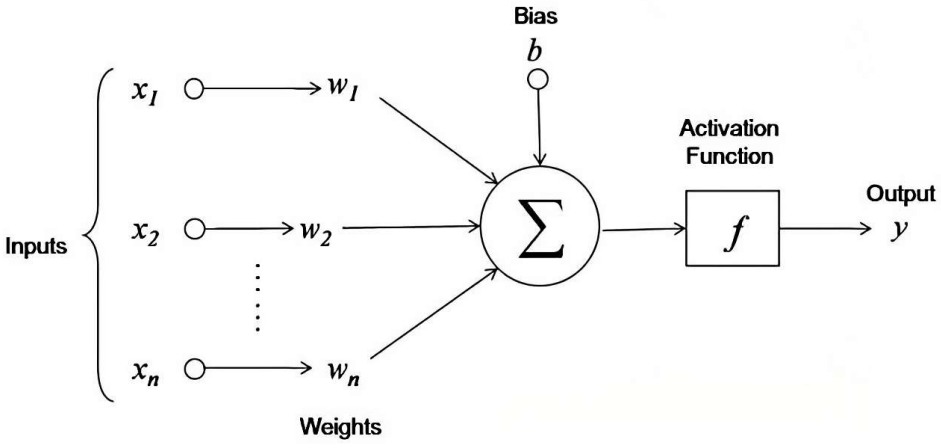

Figure 2: *Graphical representation of a neuron in a neural network. Neuron receives multiple inputs, multiplies them by associated weights, adds a bias, and processes the result through an activation function to produce an output. Source: [50].*

Mathematically, the output of a neuron can be expressed as [49]

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right), \tag{2.1}$$

where n is the number of features being fed into the neuron, x_i represents the input features, w_i are the weights, b is the bias, f is the activation function, and y is the output of the neuron.

Activation functions introduce non-linearity into the neural network and determine the degree to which a neuron should be activated, thereby influencing its contribution to the network's output. Higher values after the activation function indicate a stronger influence of the neuron on the model. The bias is a constant value added to the weighted sum of inputs before applying the activation function. This ensures that a neuron can activate even when inputs are zero, providing the network with greater flexibility to learn complex patterns during training [48]. Common activation functions include the following [51], with others depicted in Figure 3:

- **ReLU** (Rectified Linear Unit) - one of the most widely used activation functions, which outputs the input directly if it is positive, otherwise, it outputs zero

$$\text{ReLU}(x) = \max(0, x), \tag{2.2}$$

- **Sigmoid** - outputs a value between 0 and 1

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{2.3}$$

- **Tanh** - outputs a value between -1 and 1, providing a zero-centred output

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{2.4}$$

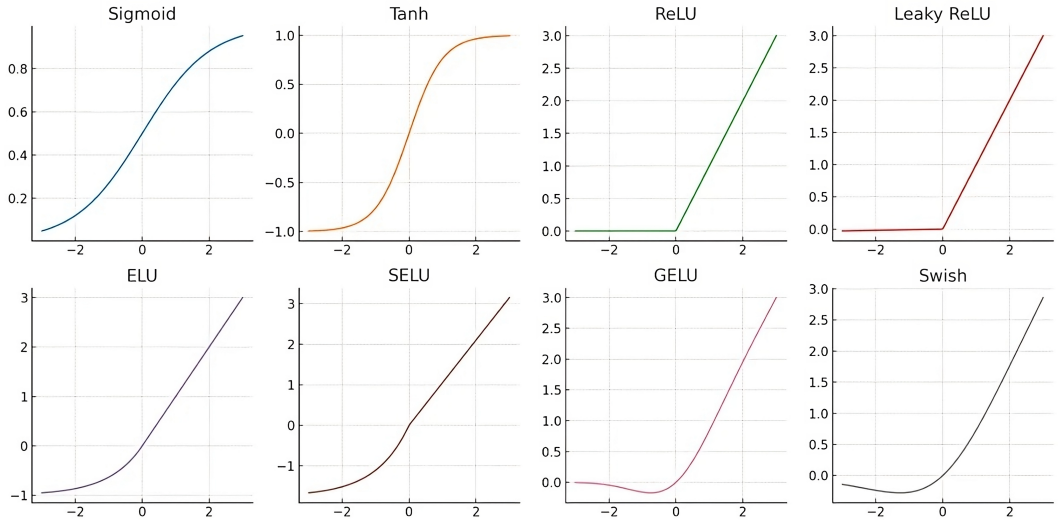

Figure 3: *Common activation functions used in deep learning. Source: [51].*

Neurons are organized into layers, forming the structure of deep neural networks, which typically contain more than two layers. Each layer can have a varying number of neurons, and all neurons in a layer are fully connected to those in the adjacent layers (Figure 4). The output of one layer serves as the input for the next. There are three main types of layers in a neural network [49]:

- **Input layer**: The first layer of the network, which directly receives the input data. The number of neurons in this layer corresponds to the number of features in the input data.
- **Hidden layers**: Positioned between the input and output layers, these layers are the core computational components. Each hidden layer consists of neurons that apply weights and activation functions to the inputs, creating intermediate representations of the data. The depth (number of hidden layers) and width (number of neurons per layer) are key factors in the network's learning capacity, enabling the modeling of nonlinear functions.
- **Output layer**: The final layer of the network, where neurons produce the output predictions. The number of neurons in this layer is determined by the specific task at hand.

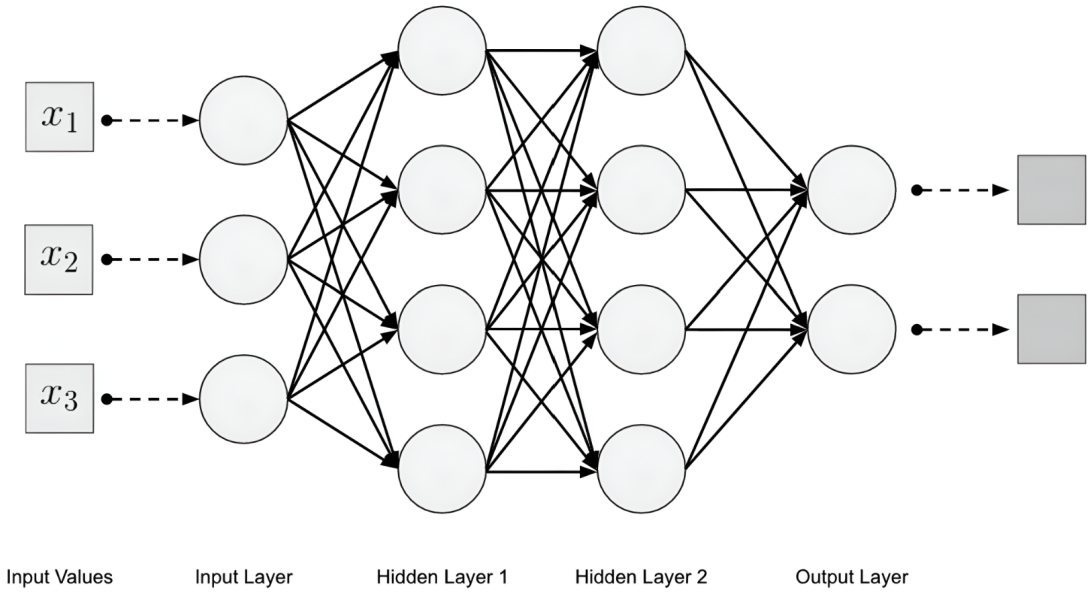

Figure 4: *Multilayer neural network with input, hidden, and output layers, showing the fully connected structure of neurons across adjacent layers. Source: [49].*

Several types of neural networks are commonly used, depending on the task [47][49]:

- **Feedforward neural networks** (FNNs): The simplest type of neural network where the information moves in only one direction - from the input layer, through the hidden layers, to the output layer. There are no cycles or loops in the network. They are typically used for tasks such as classification and regression.
- **Convolutional neural networks** (CNNs): Highly effective for processing grid-like data, such as images. They are of particular relevance in this thesis due to their ability to analyze both spatial and temporal dimensions effectively.

- **Recurrent neural networks** (RNNs): Designed to handle sequential data, such as time series or natural language. RNNs maintain a memory of previous inputs in the form of hidden states, making them effective for tasks where context or sequence matters.

2.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are specialized types of neural networks designed to process grid-like data, such as images. Artificial neurons in CNNs extract features of small portions of input images called receptive fields. There are two types of layers in CNNs - convolutional and pooling layer. In the convolutional layer, a convolution operation involves sliding a filter or kernel across the input data in steps, known as stride, to produce a feature map (Figure 5A) [48]. The filter is a small matrix of weights that are trained during the learning process to detect specific features such as edges, textures, or patterns. This involves taking the dot product between the filter and the local region of the input that it covers, producing a single output value in the feature map. After convolution, the feature maps are passed through an activation function. A pooling layer (Figure 5B) downsamples the feature maps, reducing their dimensionality and variance while preserving important features. Most common pooling operations take the mean or maximum value in the pooling window [49].

An encoder in the context of CNNs refers to the part of the network that processes the input data and transforms it into a lower-dimensional representation. This is done through a series of convolutional and pooling layers that reduce the spatial dimensions of the data while capturing the essential features [52].

After the convolutional and pooling layers, the neural network moves into fully connected layers, where the extracted features are processed. In these layers, the outputs from the previous layers are flattened into a one-dimensional vector. Each neuron in a fully connected layer is linked to all neurons in the preceding layer. This dense connection allows the model to make informed decisions based on the comprehensive set of learned features [52]. Training CNNs can be computationally expensive, often requiring high-performance GPUs.

Residual connections are a fundamental concept in deep learning, particularly in the design of deep neural networks such as ResNets (Residual Networks). In traditional deep networks, each layer feeds into the next, meaning the output of one layer becomes the input of the next. Residual connections allow the network to „skip” one or more layers and pass the input directly to a subsequent layer. Essentially, the output of a layer is added to the input of a layer further down the line [53].

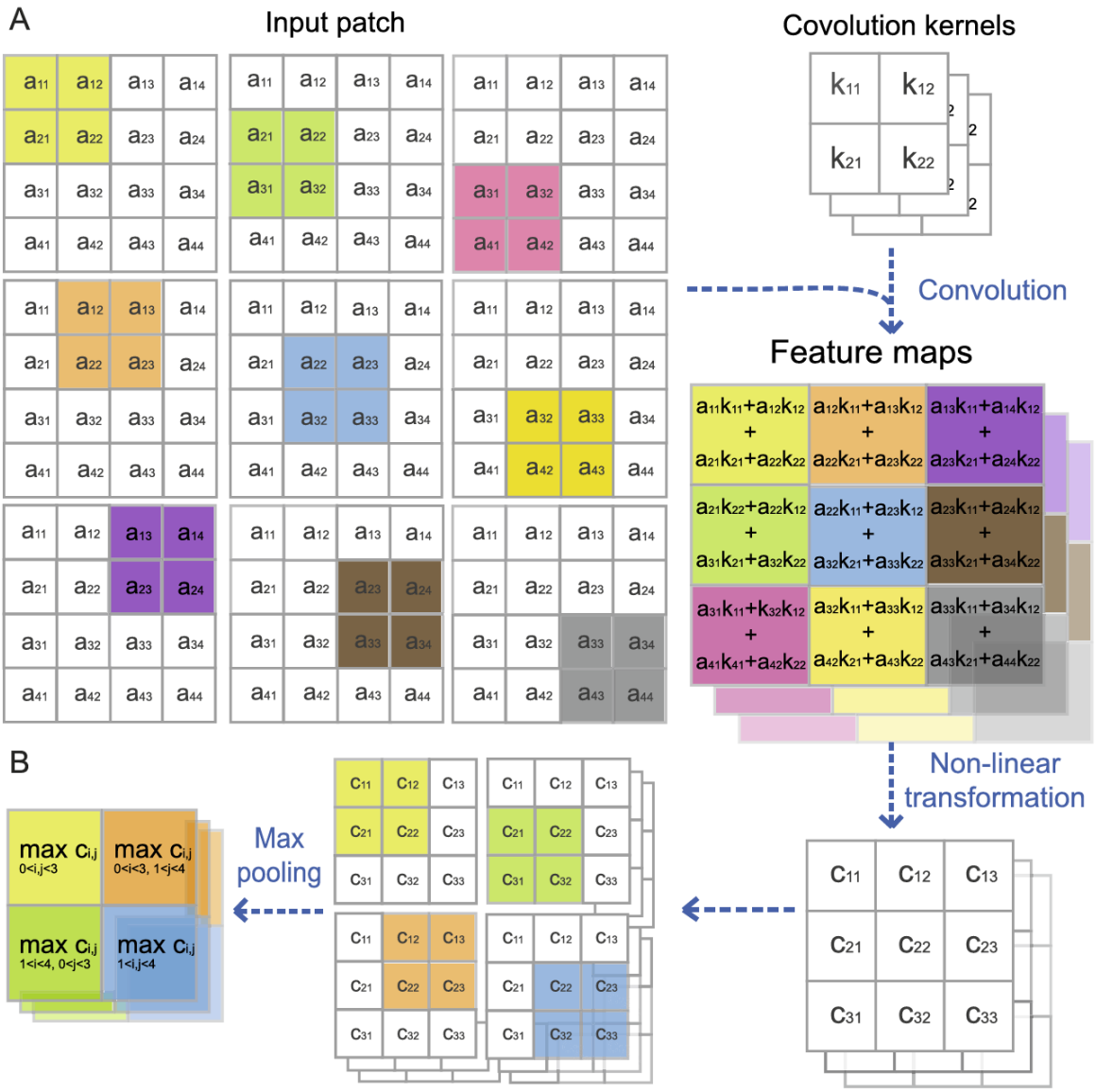

Figure 5: *Illustration of convolutional neural networks (CNNs).* ***A)*** *In the convolution layer, fields (different color blocks in the table) of the input patch (represented by a) are multiplied by matrices (convolution kernel, represented by k).* ***B)*** *In the pooling layer, the results of convolution are summarized (the max-pooling is taken as an example here). a_{ij}, c_{ij}, k_{ij} represent the number located in line i and column j in the corresponding matrix. Source: [48].*

2.3 Forward and backward propagation

During forward propagation, the input data is passed through the network layer by layer to compute the output. At the start of training, weights and biases are randomly initialized. If the output does not closely match the target label, the weights and biases need adjustment. Given that numerous weights connect each input to the output in neural networks, the learning algorithm plays a crucial role. The loss function, such as mean squared error (MSE), quantifies the difference between outputs and labels, with the objective of minimizing this loss during training [49].

Forward propagation starts at the input layer, where the input data (features) is fed into the network. Each feature corresponds to a neuron in the input layer, and each input is multiplied by a weight - a parameter learned during training. Additionally, each neuron has a bias term added to the weighted sum of inputs. The formula for this weighted sum (often referred to as the pre-activation value) for a neuron j in layer l is [49]

$$z_j^{(l)} = \sum_{i=1}^{n} w_{ij}^{(l)} a_i^{(l-1)} + b_j^{(l)}, \tag{2.5}$$

where n is the number of neurons in the previous layer $(l-1)$, $w_{ij}^{(l)}$ is the weight between the i-th neuron in the previous layer and the j-th neuron in the current layer, $a_i^{(l-1)}$ is the activation of the i-th neuron in the previous layer, $b_j^{(l)}$ is the bias term for the j-th neuron in the current layer, and $z_j^{(l)}$ is the pre-activation value for the j-th neuron.

The pre-activation value $z_j^{(l)}$ is then passed through an activation function f, which introduces non-linearity into the model. This allows the network to capture complex patterns in the data. The output of this function, known as the activation $a_j^{(l)}$, is calculated as [49]

$$a_j^{(l)} = f(z_j^{(l)}). \tag{2.6}$$

The activations from the current layer are passed on to the next layer as inputs. This process is repeated for all hidden layers in the network. Finally, the output layer provides the prediction of the network. Depending on the task, this output could be a single value or multiple values (as in regression) or a probability distribution across multiple classes (as in classification) [49].

Backward propagation (or backpropagation) is the process by which the neural network learns from the errors it makes. This process involves calculating the gradient of the loss function with respect to (w.r.t.) each weight and bias in the network and then updating these parameters to minimize the loss. The loss function L measures the difference between the network's prediction $\hat{y}$ and the target value y. For instance, in a regression task, the mean squared error (MSE) is commonly used [49]

$$L(\hat{y}, y) = (\hat{y} - y)^2. \tag{2.7}$$

The gradient of the loss function indicates how much the loss will change if the weights and biases are changed slightly. For a weight w, the gradient is denoted as $\frac{\partial L}{\partial w}$. Backpropagation utilizes the chain rule from calculus to propagate the error backward through the network. The chain rule allows the computation of the derivative of a composite function. For each weight $w_{ij}^{(l)}$, the gradient is computed as [49]

$$\frac{\partial L}{\partial w_{ij}^{(l)}} = \frac{\partial L}{\partial a_j^{(l)}} \cdot \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \cdot \frac{\partial z_j^{(l)}}{\partial w_{ij}^{(l)}}, \tag{2.8}$$

where $\frac{\partial L}{\partial a_j^{(l)}}$ is the gradient of the loss w.r.t. the activation at neuron j in layer l, $\frac{\partial a_j^{(l)}}{\partial z_j^{(l)}}$ is the gradient of the activation w.r.t. the pre-activation value, and $\frac{\partial z_j^{(l)}}{\partial w_{ij}^{(l)}}$ is the gradient of the weighted sum w.r.t. the weight.

Once the gradients are calculated, the weights and biases are updated using an optimization algorithm such as gradient descent. The update rule for a weight is [49]

$$w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta \frac{\partial L}{\partial w_{ij}^{(l)}}, \tag{2.9}$$

where η is the learning rate, a small positive value that controls the step size of the update while moving toward a minimum of the loss function. A lower learning rate ensures gradual progress, while a larger rate may speed up training but risk overshooting the optimal values and encountering divergence.

The same process goes for the bias, yielding [49]

$$b_j^{(l)} \leftarrow b_j^{(l)} - \eta \frac{\partial L}{\partial b_j^{(l)}}. \tag{2.10}$$

This process is repeated iteratively over multiple epochs (complete passes through the training dataset). Over time, the model's parameters converge to values that minimize the loss function, improving the model's accuracy.

Several optimization algorithms have been developed to improve the efficiency and effectiveness of the backpropagation process. Here are some common ones:

- **SGD** (Stochastic Gradient Descent) [54] is a popular optimization algorithm that updates weights based on the gradient of the loss function. This algorithm uses a single data point at a time to compute the gradient. For each iteration, the algorithm randomly (hence stochastic) picks one data point from the dataset and computes the gradient based on this single point, introducing high variance.
- **AdaGrad** [54] adapts the learning rate for each parameter by scaling it inversely proportional to the square root of the sum of all previous squared gradients for that parameter. Formula: $w = w - \frac{\eta}{\sqrt{G+\epsilon}} \cdot \nabla_w J(w, b)$, where G is the sum of squared gradients, and ϵ is a small constant to avoid division by zero (default 10^{-8}).
- **Adam** (Adaptive Moment Estimation) [54][55] computes adaptive learning rates for each parameter by storing both exponentially decaying averages of past gradients (first moment) and squared gradients (second moment). Formulae: $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$, $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$, where m_t and v_t are the first and second moments, respectively. g_t is the gradient at time step t, and β_1 is a decay hyperparameter (typically around 0.9), β_2 is another decay hyperparameter (typically around 0.999).

Since the estimates m_t and v_t are initialized as zero vectors, they are biased towards zero, especially during the initial steps. Adam introduces bias-corrected estimates to mitigate this effect. Formulae: $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$, $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$. Here, $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected first and second moments, respectively.

Finally, $w = w - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t$.

- **AdamW** [56] is an improvement over the traditional Adam optimizer. The primary difference is how the weight decay regularization is applied. AdamW decouples the weight decay from the gradient update, applying it directly to the weights after the update step. This leads to more effective regularization and often results in better generalization. The weight update in AdamW is similar to Adam, with an additional step for weight decay.

  Formula: $w = w - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t - \eta \cdot \lambda \cdot w$. Here, λ is the weight decay coefficient, with 0.01 default value. The first term handles the optimization step, while the second term handles the decoupled weight decay.

2.4 Hyperparameters

Hyperparameters are settings that are not learned during training but are set before the training process begins. The role of hyperparameters is critical to the performance and effectiveness of a model. Common hyperparameters include batch size, number of epochs, and regularizations such as dropout. Other hyperparameters include learning rate, activation functions, optimizer, number of layers and neurons, and kernel size in CNNs, but they have already been explained in the previous subsections. Adjusting these hyperparameters is demanding and requires many attempts to determine the optimal values.

Training is conducted over a number of epochs, which corresponds to the number of complete passes through the entire training dataset. While too few epochs can lead to underfitting, too many can cause overfitting, where the model performs well on training data but poorly on unseen data (Figure 6).

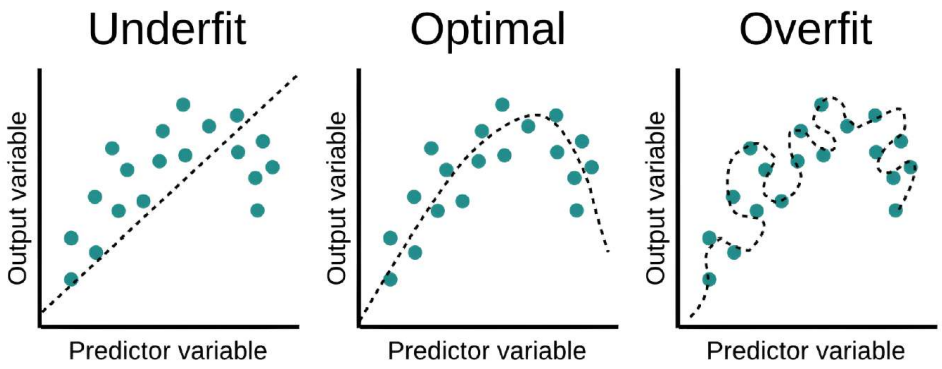

Figure 6: *Examples of underfitting, optimal fit, and overfitting. Source: [57].*

Typically, a dataset is divided into training, validation, and testing subsets. The model is trained on the training data, while its performance is evaluated and fine-tuned using the validation data to gauge how well it generalizes to unseen data. The key is to find the „sweet spot" (Figure 7), where the validation loss continues to decrease up to that point, ensuring effective learning, but begins to increase afterward, indicating overfitting. Finally, the model is tested on the testing data to make final predictions and assess its generalization performance.

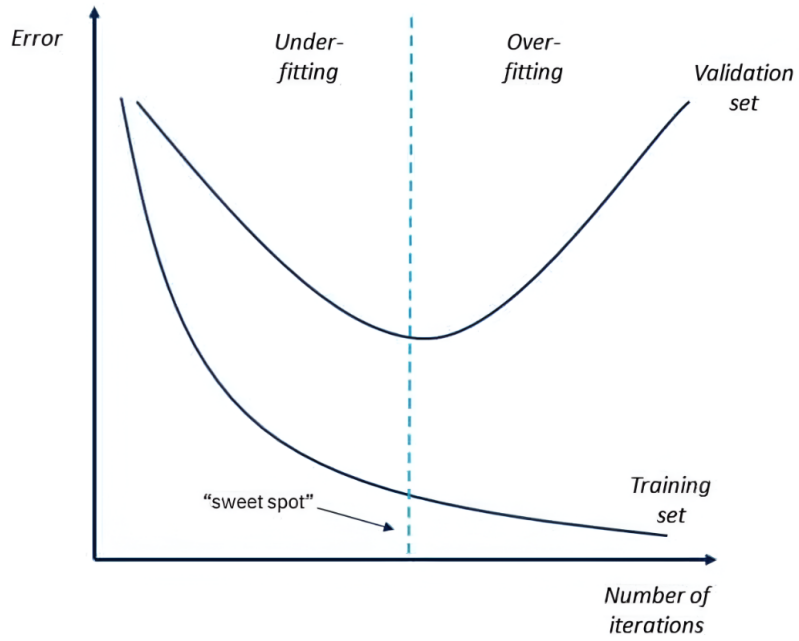

Figure 7: *Training vs. validation loss over epochs - identifying the „sweet spot" between underfitting and overfitting. Source: [58].*

The dataset is typically split into batches, with weights updated after processing each batch. This approach mitigates the computational and time costs associated with performing backpropagation on the entire dataset at once. A larger batch size includes more data points, leading to more accurate gradient estimates and more stable updates during training. However, finding the optimal batch size often requires experimentation.

Dropout regularization is a technique used in neural networks to prevent overfitting by randomly „dropping out" (i.e., setting to zero) a fraction of the neurons during training. The dropout rate (a hyperparameter) controls the fraction of neurons to drop out. For example, a dropout rate of 0.5 means that each neuron has a 50% chance of being dropped. By preventing the network from becoming too reliant on specific neurons, dropout helps the model generalize better to unseen data [59].

3 Materials and Methods

In this section, I present the essential materials and methods utilized in this research. I start by describing the sources and preprocessing of both tide gauge and atmospheric training data. Next, I describe the deep learning model architecture and training procedures, highlighting the techniques used to enhance model performance, especially for extreme events. I also discuss input data variations studies conducted to identify the most effective model configuration, followed by the evaluation methods employed to assess model performance.

3.1 Tide gauge training data

The sea surface height (SSH) data, with a 1-minute temporal resolution, was obtained from the mareographic station in Bakar (latitude 45° 18.3' N, longitude 14° 32.4' E), maintained by the Department of Geophysics at the Faculty of Science, University of Zagreb. The sea level measurements used in this study span the period from June 28th, 2003 to December 31st, 2023. Data quality control was performed by the Department of Geophysics, where the data was cleaned to remove obvious outliers.

The dataset contains some gaps, which were addressed through interpolation. Specifically, I applied linear interpolation to fill gaps shorter than 5 consecutive hours, while gaps exceeding this threshold were left unaltered. To determine the validity of the dataset, I first identified days with missing values in the interpolated SSH time series. Days without missing values were considered complete. For a day to be classified as valid, it had to meet the following criterion: the data must be complete for that day, as well as for the preceding day and the subsequent two days, ensuring a total of four consecutive days with complete data. In total, I identified 6,994 valid days, indicating that 93.4% of the dataset was usable for the research.

To isolate high-frequency sea level oscillations (hereafter referred to as HF-SLO) from lower-frequency components, I employed a Butterworth band-pass filter. The low-frequency cutoff was set at $\frac{1}{3600}$ Hz, corresponding to a period of 1 hour, while the high-frequency cutoff was set at $\frac{1}{180}$ Hz, corresponding to a period of 3 minutes. I selected these cutoff periods based on the characteristic seiche periods of around 20 minutes in Bakar Bay [38], making it essential to capture frequencies in this range. I considered components with periods longer than 1 hour irrelevant for the analysis. Only frequencies up to half the sampling rate ($\frac{1}{60}$ Hz) were retained to adhere to the Nyquist criterion[1]. I set the filter order to 20 to achieve a sharp transition between the passband and the stopband, effectively isolating the desired frequency range. The functions used for extracting HF-SLO are provided in Appendix A.

[1]For discrete time series data, such as the 1-minute interval SSH measurements, this means that frequencies above half the sampling rate (known as the Nyquist frequency) cannot be accurately represented and may result in aliasing – where higher frequencies are indistinguishably mapped to lower frequencies [60].

The resulting filtered signal, representing the HF-SLO, is shown in Figure 8a for the year 2014 alongside the measured total sea surface height. In Figure 8b, the time series is zoomed in to focus on a 2-day period, allowing for a clearer view of the filtering effect.

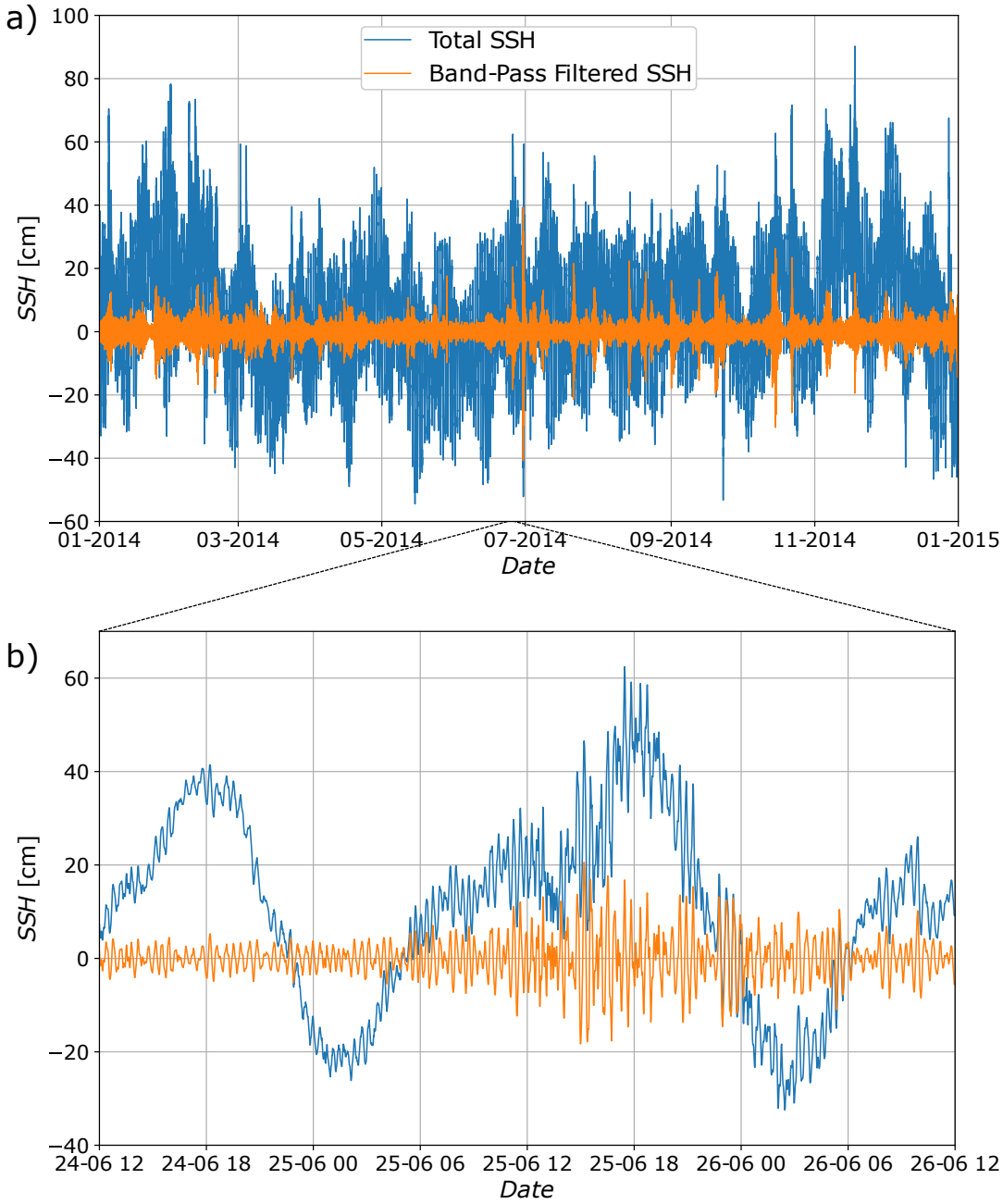

Figure 8: ***a)*** *Time series of SSH and its high-frequency component for the year 2014;* ***b)*** *Zoomed-in view of the upper figure from June 24th, 12:00 to June 26th, 12:00.*

To incorporate the frequency content of the band-pass filtered SSH data for each day into the model training, I computed the power spectral density (PSD). The PSD provides a measure of the signal's power distribution across different frequencies. This computation involved applying the fast Fourier transform (FFT) to the SSH time series, converting it from the time domain to the frequency domain.

To mitigate noise and enhance the interpretability of the spectral data, I applied a moving average filter to the computed spectrum for each day. This smoothing was done using a convolution operation with a uniform window of size 250 points, emphasizing significant frequencies while suppressing the impact of noise. The power spectra of the total SSH and the band-pass filtered SSH, along with the smoothed spectra, for the entire period are presented in Figure 9a. A distinct peak observed around a period of 20 minutes corresponds to the seiche frequency of Bakar Bay [38]. Earlier studies [61] identified three fundamental modes at periods of 26.9, 22.3, and 19.7 minutes, as well as higher modes at periods of 7.8 and 4.3 minutes [42]. These modes can be observed on Figure 9b.

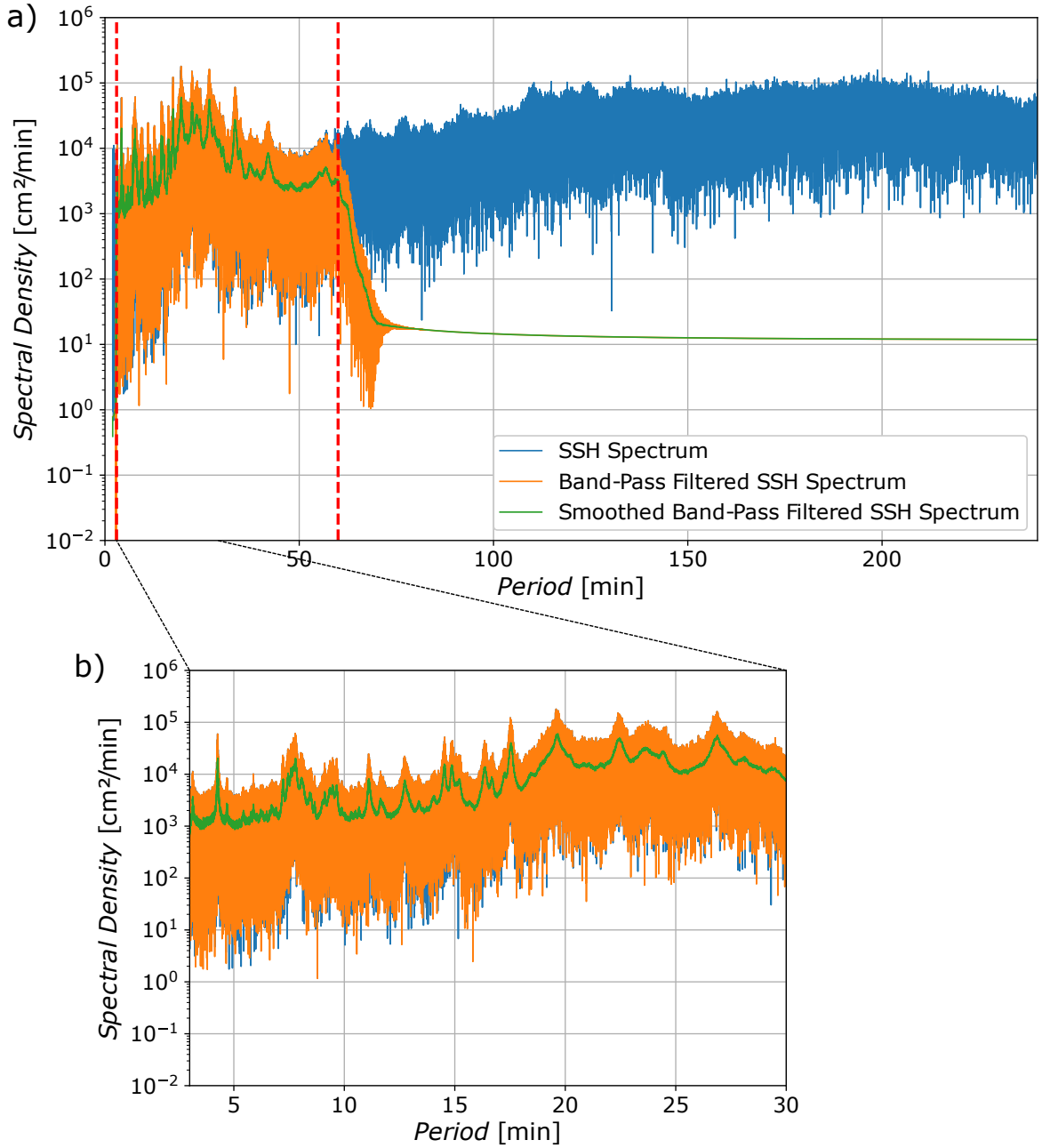

Figure 9: ***a)*** *Power spectral density of the SSH time series from the Bakar tide gauge, including the band-pass filtered SSH and its smoothed spectrum;* ***b)*** *Zoomed-in view of the spectrum, focusing on the period range from 3 minutes to 30 minutes.*

To enhance the model's performance in forecasting extreme HF-SLO events, I employed an oversampling technique. I defined extreme events as days where the HF-SLO exceeded the 99.99th percentile of its entire time series, corresponding to 19.2 cm in height. While this threshold may not signify extreme conditions in a broader context, it represents significant variability for Bakar Bay. Given the rarity of such events within the training period, I implemented oversampling to balance the training dataset and improve forecasting accuracy. I divided the training dataset into two subsets based on this threshold: one containing extreme HF-SLO events and the other comprising all HF-SLO events. The extreme events accounted for 108 days, representing 1.83% of the days included in the training period. During model training, samples were drawn from these subsets with a probability distribution of 10% for the „extreme events" subset and 90% for the „all event" subset, ensuring that the model was adequately exposed to extreme HF-SLO events.

For training the model, tide gauge input features were derived from the previous 24 hours up to the current time (0 h). The inputs include HF-SLO and their power spectrum, both represented as 1-dimensional data. The target values for the model are the absolute values of maximum daily HF-SLO amplitudes. Typically, the negative and positive maxima of the HF-SLO are comparable. Consequently, I did not differentiate between the two extremes. The model's objective is to forecast the maximum daily HF-SLO amplitude up to 72 hours in advance (i.e., from the current time, 0 h, to 72 h into the future).

To ensure that all features contribute proportionately to the model, I standardized the data before the training. Standardization brings the features to a common scale, centered around zero with a standard deviation of one. Standardization was performed using the following formula

$$x_{\text{st}} = \frac{x - \mu}{\sigma}, \tag{3.1}$$

where x_{st} is the standardized value, x is the original value, μ is the mean value of the variable, and σ is the standard deviation of the variable over the entire period, from June 28th, 2003 to December 31st, 2023. The calculated mean and standard deviations for tide gauge data can be found in Table 1.

Table 1: *Mean and standard deviation values for tide gauge variables.*

Tide Gauge Variable	Mean	Standard Deviation
HF-SLO [cm]	$3.8 \cdot 10^{-5}$	2.0
Smoothed spectrum [cm^2/min]	20.7	29.3
Unsmoothed spectrum [cm^2/min]	24.3	73.0
Daily maxima of HF-SLO [cm]	6.2	4.5

3.2 Atmospheric training data

All atmospheric data used in this study were obtained from the ERA5 reanalysis datasets [62], available through the Copernicus Climate Data Store. The atmospheric variables downloaded from the "ERA5 hourly data on pressure levels from 1940 to present" dataset [63] include the eastward (u) and northward (v) wind components, temperature (T), specific humidity (q), and geopotential (z). Mean sea level pressure (*MSLP*) data were obtained from the "ERA5 hourly data on single levels from 1940 to present" dataset [64]. The data cover the period from June 2003 to December 2023. The downloaded ERA5 data have a temporal resolution of 1 hour and a spatial resolution of $0.125° \times 0.125°$. For the analysis, all atmospheric input fields from ERA5 were cropped to focus on the Adriatic basin, represented by a 56×72 spatial grid, ranging from 39° N to 46° N and from 12° E to 21° E.

The hourly atmospheric data were downloaded from the ERA5 reanalysis at five pressure levels: 1000, 850, 700, 500, and 400 hPa. Four of these five pressure levels were utilized in the model - 1000, 850, 700, and 500 hPa - for the downloaded data. The 400 hPa data were specifically acquired to calculate the Richardson number between the layers 1000 and 850 hPa, 850 and 700 hPa, 700 and 500 hPa, and 500 and 400 hPa, due to the significance of the Richardson number in the upper atmospheric layers [23]. This approach results in a 4D input field for the Richardson number, aligning with the dimensionality of other atmospheric variables used in the model training.

Richardson number, *Ri*, represents a measure of atmospheric stability and is given by the following formula [30]

$$Ri = \frac{g}{\overline{\theta}} \cdot \frac{\Delta\theta}{\Delta z} \left(\left(\frac{\Delta u}{\Delta z} \right)^2 + \left(\frac{\Delta v}{\Delta z} \right)^2 \right)^{-1}, \tag{3.2}$$

where g is the acceleration due to gravity, $\overline{\theta}$ is the average potential temperature between two pressure levels, $\frac{\Delta\theta}{\Delta z}$ is the change in potential temperature with height, and $\frac{\Delta u}{\Delta z}$ and $\frac{\Delta v}{\Delta z}$ are the vertical gradients of the u and v wind components, representing vertical wind shear.

To calculate the Richardson number, the first step involved converting the geopotential at each pressure level into height, expressed in meters. I achieved this conversion by dividing the geopotential values by the gravitational constant, g. Next, I determined the potential temperature, θ, for each pressure level P. I calculated the potential temperature from temperature T, using the formula [65]

$$\theta = T \cdot \left(\frac{P_0}{P} \right)^{R/c_p}, \tag{3.3}$$

where P_0 is a reference pressure (1000 hPa), R is the specific gas constant for dry air (287.05 J/(kg·K)), c_p is the specific heat at constant pressure, which equals 1005 J/(kg·K) for dry air.

The atmospheric input data includes mentioned variables from the previous 24 hours to 72 hours into the future. Examples of atmospheric fields are shown in Figure 10. I standardized these data according to equation (3.1), described in the previous subsection. However, due to the large file sizes and the extensive computation time required, standardization was performed only for the years 2005, 2010, 2015, and 2020. The calculated mean and standard deviations for atmospheric data are shown in Table 2.

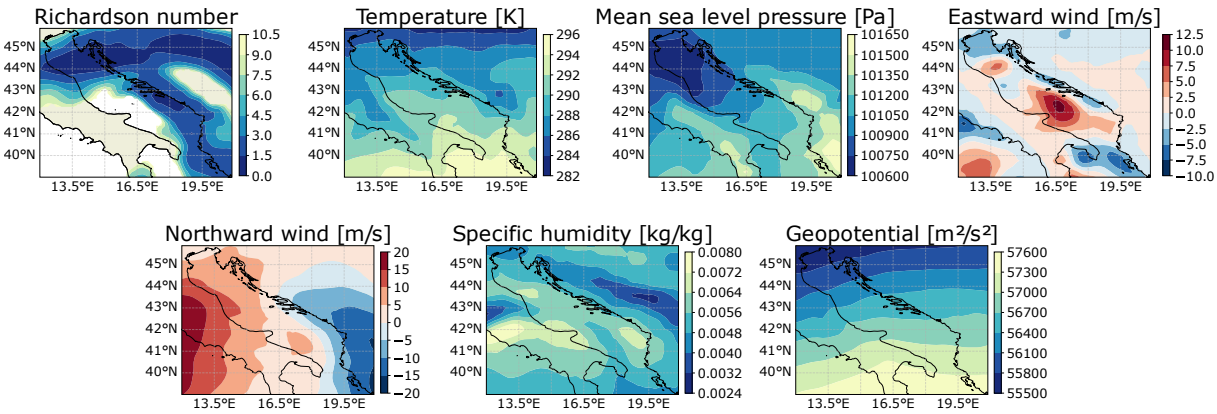

Figure 10: *Selected snapshots of original ERA5 atmospheric fields.*

Table 2: *Mean and standard deviation values for atmospheric variables.*

Atmospheric Variable	Mean	Standard Deviation
q [kg/kg]	0.004	0.003
T [K]	274.2	13.0
u [m/s]	3.2	6.9
v [m/s]	-0.8	6.7
$MSLP$ [Pa]	101584	65
z [m^2/s^2]	25379	20187
Ri	21.8	21.2

The training set encompasses the period from the start of the valid days, beginning on June 28th, 2003, through December 31st, 2020. This dataset includes only those days identified as valid based on the criteria established for data completeness.

3.3 The network training, validation, testing, Model selection

I organized the data used in this study into PyTorch files (.pt file extension). Each file contains atmospheric data spanning from 24 hours before to 72 hours after the forecasting window, tide gauge data covering the 24-hour period leading up to the forecasting window, and target data representing the next three days.

I trained the deep learning model proposed in this thesis using the MSE loss function, which quantitatively measures the difference between the forecasted values and the ground truth. The model was optimized using the AdamW optimizer, an advanced variant of the Adam optimizer [56], with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\lambda = 0.01$. I set the base learning rate to 10^{-5}, ensuring gradual and steady progress in minimizing the loss function while allowing the adaptive learning rate mechanism to fine-tune adjustments during training.

The training process involved a batch size of 128 file samples, with the model being trained over 30 epochs, allowing the model to iterate over the dataset multiple times, improving its ability to generalize from the training data. Each epoch consisted of 46 training steps (batches). The entire training process required approximately 3 to 3.5 hours on a single NVIDIA RTX A5000 graphics card [66] (CUDA version 12.4), a high-performance GPU that facilitated the efficient handling of large-scale data and complex model computations. I implemented and trained the model using the PyTorch framework, a widely recognized tool in the deep learning community for its flexibility and ease of use [67].

I validated the model using data from the year 2021, and conducted the final forecast (testing) on data from the years 2022 and 2023. I prepared the validation and testing datasets in the same manner as the training dataset. During the validation period, I identified five extreme events, and the testing period contained 15 such events. These extreme events were of particular interest, as they are more challenging to forecast and can have significant impacts on coastal areas, including the risk of flooding.

I carried out training and validation concurrently, allowing for real-time assessment of the model's performance on unseen data. After each epoch, I evaluated the model on the validation dataset to assess its ability to generalize. The validation dataset was processed in batches of 16 file samples, with the MSE loss criterion applied to each batch. I observed that the model encountered difficulties in accurately forecasting extreme events, which often led to significant spikes in batch loss during validation.

Therefore, to select the best-performing model, I took the following approach: the model corresponding to the epoch where the highest spike in validation batch loss reached its minimum, relative to other epochs, was deemed the most effective. I prioritized this method over selecting the model with the lowest average epoch validation loss, as it specifically targeted the model's ability to forecast extreme events. An increasing validation loss would indicate a decline in the model's generalization ability, suggesting overfitting or other issues, with train loss still decreasing. This selection process is illustrated in Figure 11b, where it is evident that the spike in validation batch loss reached its minimum in the 15th epoch, signifying the optimal model for forecasting extreme events. In contrast, Figure 11a shows that the lowest average validation loss occurred in the 28th epoch. However, this did not correspond to better forecasting accuracy during the testing period in preliminary tests, further supporting the chosen model selection strategy.

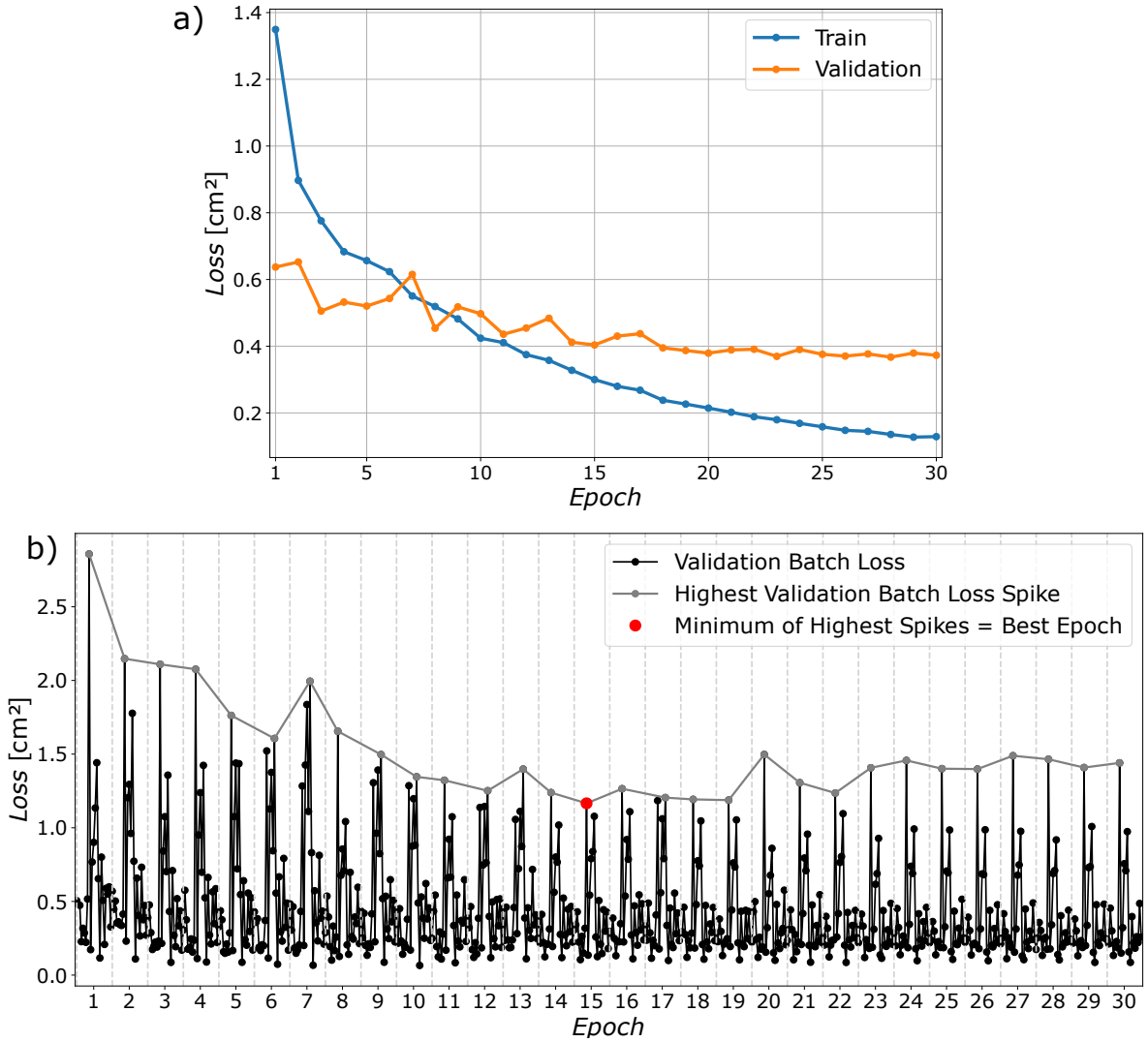

Figure 11: ***a)*** *Average train vs. average validation loss per epoch, highlighting the 28th epoch with the lowest overall loss;* ***b)*** *Validation batch loss, indicating the 15th epoch as the point where the highest spike in batch loss reached its minimum, marking the selection of the optimal model for forecasting extreme events.*

3.4 Model architecture

The model presented in this thesis is a deep convolutional neural network designed to forecast the absolute values of future daily maxima of HF-SLO. This model leverages heterogeneous data, including atmospheric and tide gauge data to make the forecast over the next 3 days. To enable the model to account for the influence of atmospheric conditions on HF-SLO, the network incorporates the forcing data over a range of time steps from the previous 24 hours to 72 hours ahead. The tide gauge data are considered only for the 24 hours leading up to the forecast window.

Figure 12 provides a summary of the model's architecture and its key components. The atmospheric encoder processes 4D atmospheric input tensors. It extracts both spatial and temporal features by using 3D and 1D convolutional layers. The output vector from the

atmospheric encoder and the 1D input tide gauge vectors are each passed through fully connected (dense) layers. The resulting outputs are then combined in a fusion-regression block, which consists of a series of fully connected layers, ultimately leading to the final forecast outputs. The whole model architecture can be found in Appendix B. Detailed descriptions of each component of the architecture are provided below.

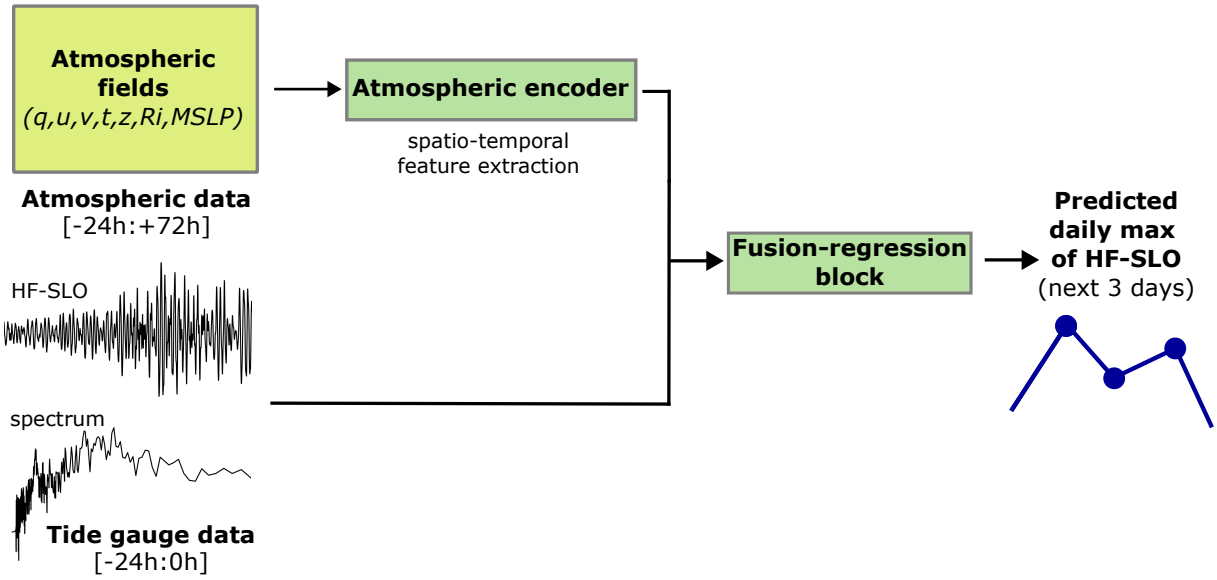

Figure 12: *Overview of the model architecture with its key components.*

3.4.1 Atmospheric encoder

The atmospheric encoder comprises two stages. The first stage utilizes a 3D max-pooling operation followed by several distinct 3D convolutional layers to process different input atmospheric components independently, including wind, specific humidity, temperature, geopotential, Richardson number, and mean sea level pressure. The 3D convolutional layers are designed to extract spatio-temporal features from the atmospheric data. The second stage of the atmospheric encoder employs a 1D convolutional layer, followed by two blocks with residual connections, for temporal feature extraction. Residual connections [53] enable the network to preserve or improve its performance by learning incremental features rather than redundant or irrelevant ones. This approach ensures that important information from earlier layers is retained, helping to preserve useful features throughout the network.

The atmospheric input tensors for the Adriatic basin are represented by a $n \times 56 \times 72$ spatial grid over 96 hourly time steps, where n is the number of vertical dimensions, which act as a number of channels.

The first stage of the atmospheric encoder processes each atmospheric input tensor (q, T, u, v, z, Ri, $MSLP$) independently. The structure for the first stage is illustrated in Figure 13. I concatenated the two wind components (u and v) into a single tensor along the vertical

dimension. Each input tensor undergoes initial downsampling using a 3D max-pooling operation with a kernel size of 1×2×2 and stride 1×2×2, reducing the spatial dimensions while preserving the temporal dimension. Consequently, each atmospheric tensor has a shape of n×96×28×36. I employed max-pooling as the initial operation to downsample the input atmospheric data. I chose this approach primarily to reduce the dimensionality of the feature maps, ensuring that the model training process remained within memory constraints. I selected max-pooling over average pooling because it yielded better results in preliminary tests.

Then, I applied a series of 3D convolutional layers followed by ReLU activation functions and dropout to each tensor. I used a dropout rate of 0.1 throughout the model to prevent overfitting. The first convolutional layer employs a kernel size of 2×3×3 with a stride of 2×3×3 and 64 output channels, producing a tensor of shape 64×48×9×12. The second convolutional layer, with a kernel size of 1×3×3, stride 1×3×3, and 256 output channels, results in a tensor of shape 256×48×3×4. The final convolutional layer uses a kernel size of 1×3×4 with 512 output channels, yielding a tensor of shape 512×48.

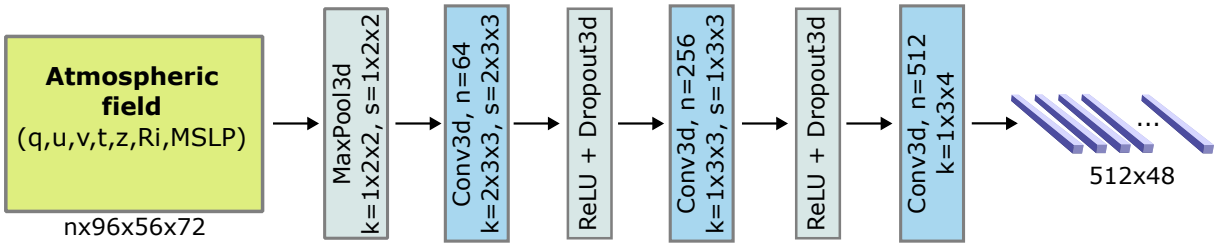

Figure 13: *Diagram illustrating the first stage of the atmospheric encoder, where each atmospheric input tensor undergoes downsampling via 3D max-pooling, followed by a series of 3D convolutional layers with ReLU activation and dropout, resulting in a tensor of shape 512×48. The variables n, k, and s denote the number of output channels, the kernel size, and stride, respectively.*

The structure for the second stage of the atmospheric encoder is shown in Figure 14. I concatenated the outputs from the first stage to form a tensor of shape (6·512)×48. Following the spatio-temporal feature extraction, I applied a 1D convolutional layer with a kernel size of 5, stride of 1, and 256 output channels to capture temporal dependencies across the concatenated features. This results in a tensor with dimensions 256×44. I chose a kernel size of 5 to ensure a receptive field of at least 10 hours, as the kernel size is multiplied by 2 due to the temporal stride of 2 in the first stage of the atmospheric encoder.

The resulting tensor is further processed by two blocks with residual connections, each involving 1D convolution with a kernel size of 1, stride of 1, and 256 output layers, followed by SELU activation function, and dropout. Subsequently, flattening converts the output tensor into a 1D vector of size 11264 (i.e., 256·44), representing the number of features.

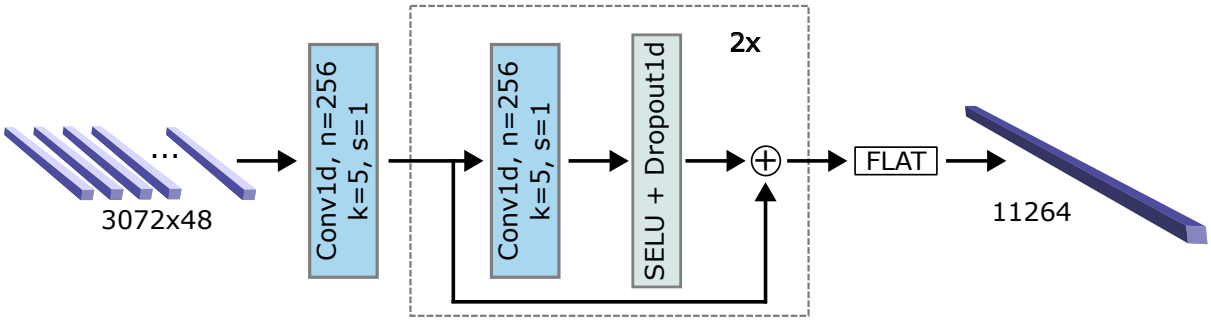

Figure 14: *Diagram illustrating the second stage of the atmospheric encoder, where the concatenated output tensor (6·512)×48 from the first stage undergoes 1D convolution, followed by two residual blocks, each applying 1D convolution with SELU activation and dropout, to capture temporal dependencies, culminating in a flattened vector of size 11264. The variables n, k, and s represent the number of output channels, kernel size, and stride, respectively.*

3.4.2 Fusion-regression block

The fusion-regression block combines the output vector from the atmospheric encoder with tide gauge vectors, which include HF-SLO and their spectral density, both from the previous 24 hours. The structure of the fusion-regression block, including the final forecast layer, is shown in Figure 15. Each 1D input signal is passed through a fully connected layer, producing three feature vectors of size 512.

The processed signals are then concatenated and passed through four blocks with residual connections, each including a fully connected layer, producing a vector of size 3·512, followed by SELU activation and dropout. The combined features are then passed through a fully connected layer to produce a feature vector of size 8192. The final fully connected layer maps the combined features to a 3-dimensional output, representing the maximum height of HF-SLO for the next 3 days.

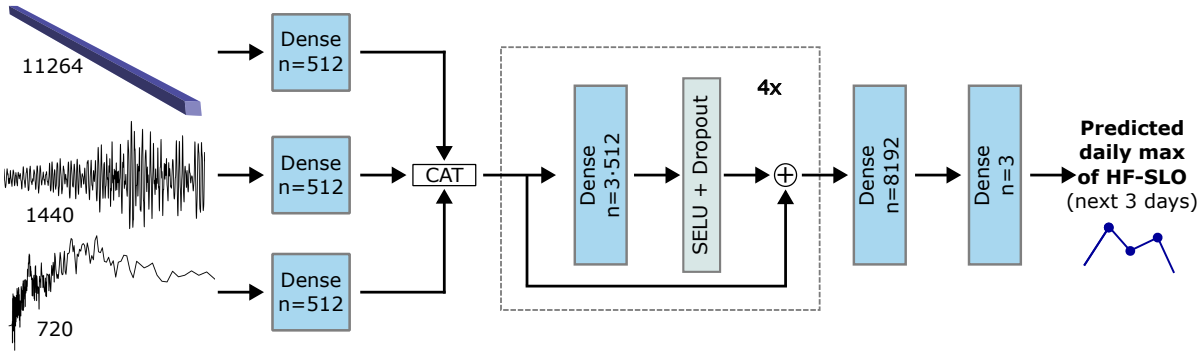

Figure 15: *Diagram illustrating the fusion-regression block, where the output vector from the atmospheric encoder is combined with tide gauge vectors. Each input is processed through fully connected layers, followed by residual blocks, to produce a feature vector that is mapped to a 3-dimensional output representing forecasted maximum heights of HF-SLO for the next 3 days.*

3.5 Input data variations and performance measures

I trained various model variants to explore the impact of different hyperparameters and input training data on the performance of the deep learning network. The primary objective was to identify the model configuration that delivers the best performance, particularly in forecasting extreme events, which are of significant interest in this study. I initially trained the network using a specific set of hyperparameters and input data as outlined in Section 3.3, which corresponds to the best performance, as will be shown in Section 4.

The purpose of the ablation study was to assess the individual contribution of specific input features and oversampling technique to the model's performance. This involved systematically removing some components from the training dataset and evaluating the impact on model performance:

- To determine the significance of the Richardson number and the frequency spectrum in training, I excluded these features from the dataset. This approach allowed us to isolate their individual impact on the model's performance. In the forward pass of the architecture, it was necessary to exclude the spectrum in the fusion-regression block and change the number of output channels in the residual connection to $2{\cdot}512$ to ensure that tensor dimensions aligned correctly during element-wise addition, preventing dimension mismatches.
- I also removed the oversampling used to balance the training data, assessing its influence on the model's ability to forecast extreme events.

In addition to the ablation study, I explored alternative training configurations to determine if different approaches could yield better results:

- I experimented with oversampling ratios of 20%, 30%, and 50% to examine whether increasing the proportion of extreme events in the training data would enhance the model's forecasting capability for such events.
- I also trained the network with batch sizes of 64 and 256 to evaluate the impact of batch size on the model's performance. The batch size affects the frequency and intensity of weight updates during backpropagation, which can influence how well the model learns from the data.
- To evaluate how spectral smoothing affects model performance, I conducted training using an unsmoothed spectrum by omitting the moving average applied during preprocessing.
- I also took a larger spatial domain into account covering the Mediterranean region, ranging from 35° N to 46° N and from 2° W to 21° E, represented by an 87×186 spatial grid. The only architectural modification required was an adjustment in the kernel size - from $1\times3\times4$ to $1\times4\times10$ - in the final 3D convolution layer of the first stage of the atmospheric encoder. This change ensured that the atmospheric data had the appropriate

shape to pass through the first 1D convolution in the second stage of the atmospheric encoder. Additionally, I conducted the training with a batch size of 32 due to RAM limitations. I standardized the data from the Mediterranean region using the same mean and standard deviation values derived from the Adriatic region, using equation (3.1).

To ensure consistency and eliminate the influence of randomness, I trained all model variants using the same initial weights and biases.

To evaluate and compare the performance of all model variants on the testing dataset, I calculated several evaluation metrics. I computed these metrics both for the entire dataset, providing an overall assessment of model performance, and separately for extreme and non-extreme events, offering insights into the model's handling of particularly challenging or typical scenarios. I defined extreme events as per the criteria outlined in Section 3.1. I used the following metrics:

- Mean absolute error (MAE) - the average of the absolute differences between forecasted y_{pred} and ground truth y_{truth} values (n is the number of forecasts)

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_{i,\text{pred}} - y_{i,\text{truth}}|. \tag{3.4}$$

- Root mean square error (RMSE) - the square root of the average squared differences between forecasted and ground truth values, which emphasizes larger errors

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{i,\text{pred}} - y_{i,\text{truth}})^2}. \tag{3.5}$$

- Bias - the average difference between forecasted and ground truth values, indicating whether the model consistently overestimates or underestimates the target variable. A bias close to zero is ideal. The bias is calculated as

$$\text{Bias} = \frac{1}{n}\sum_{i=1}^{n}(y_{i,\text{pred}} - y_{i,\text{truth}}). \tag{3.6}$$

- Forecasting accuracy - I established the accuracy thresholds for evaluating forecasts based on one standard deviation of the ground truth values from the testing period. Specifically, I calculated these thresholds as 4.39 cm for overall performance, 3.27 cm for non-extreme events, and 5.26 cm for extreme events. These values served as the thresholds to define the forecasting accuracy metric, which is the ratio of forecasts falling within these specified error margins to the total number of forecasts. This approach ensures that the accuracy metric remains meaningful and appropriately stringent, accurately accounting for the variability inherent in the data. By using these

calculated thresholds, the evaluation avoids artificially inflated accuracy scores that could result from overly lenient thresholds, especially when distinguishing between models with different input features or parameters.

I considered the model variant that achieved the best performance metrics, especially on extreme events, to be the most effective. Given the focus on accurately forecasting higher amplitudes of HF-SLO, I prioritized the ability to model extreme events over lower amplitude forecasts, which were performing adequately, as will be demonstrated in Section 4.

4 Results

In this section I present the outcomes of the experiments conducted to evaluate the performance of the deep learning models designed for forecasting the daily maxima of high-frequency sea level oscillations. I tested various model configurations, each with different input features, hyperparameter settings, and architectural modifications. The results are organized to first provide an overview of the best-performing model, followed by a detailed analysis of other model variants. Visual representations of the results are also provided to support the findings. The key metrics considered in this analysis include root mean square error (RMSE), mean absolute error (MAE), model bias, and forecasting accuracy, with a particular emphasis on the model's ability to forecast extreme events.

4.1 Model performance analysis

To evaluate the model's forecasting capabilities across varying forecast lead times, I constructed three distinct time series from daily forecasts, spanning the testing period from January 1, 2022, to December 31, 2023. These time series correspond to forecast lead times of 24, 48, and 72 hours. The first series aggregates forecasts for the first day (lead time 24 h), the second series compiles forecasts for the second day (lead time 48 h), and the third series includes forecasts for the third day (lead time 72 h). This methodology allows for a comprehensive examination of the model's performance as the forecast horizon extends.

Figures 16 and 17 present the time series comparison between the forecasted and observed daily maxima of HF-SLO for the three lead times in 2022 and 2023, respectively. It is evident that the model's forecasting accuracy diminishes as the lead time increases, in particular for extreme events. At the 24-hour lead time, the model more closely tracks the observed values, capturing the peaks of the HF-SLO with relatively high accuracy. However, as the forecast horizon extends to 48 and 72 hours, the amplitude of forecasts tends to deviate more from the observed values, especially during extreme events. Generally, the model tends to underestimate the peak values for all three forecast lead times. This is particularly evident in specific periods, such as August and September of 2022, and July and August of 2023, where the model struggles to capture the extremes accurately. For lower amplitudes, the differences between lead times are less pronounced, indicating that the model maintains similar accuracy across all lead times for non-extreme events.

However, it is important to note that the model occasionally forecasts negative values, which are not physically meaningful in this context, as I am forecasting the absolute values of maximum daily HF-SLO amplitudes. These negative predictions should be disregarded in the analysis, as they do not represent valid outcomes.

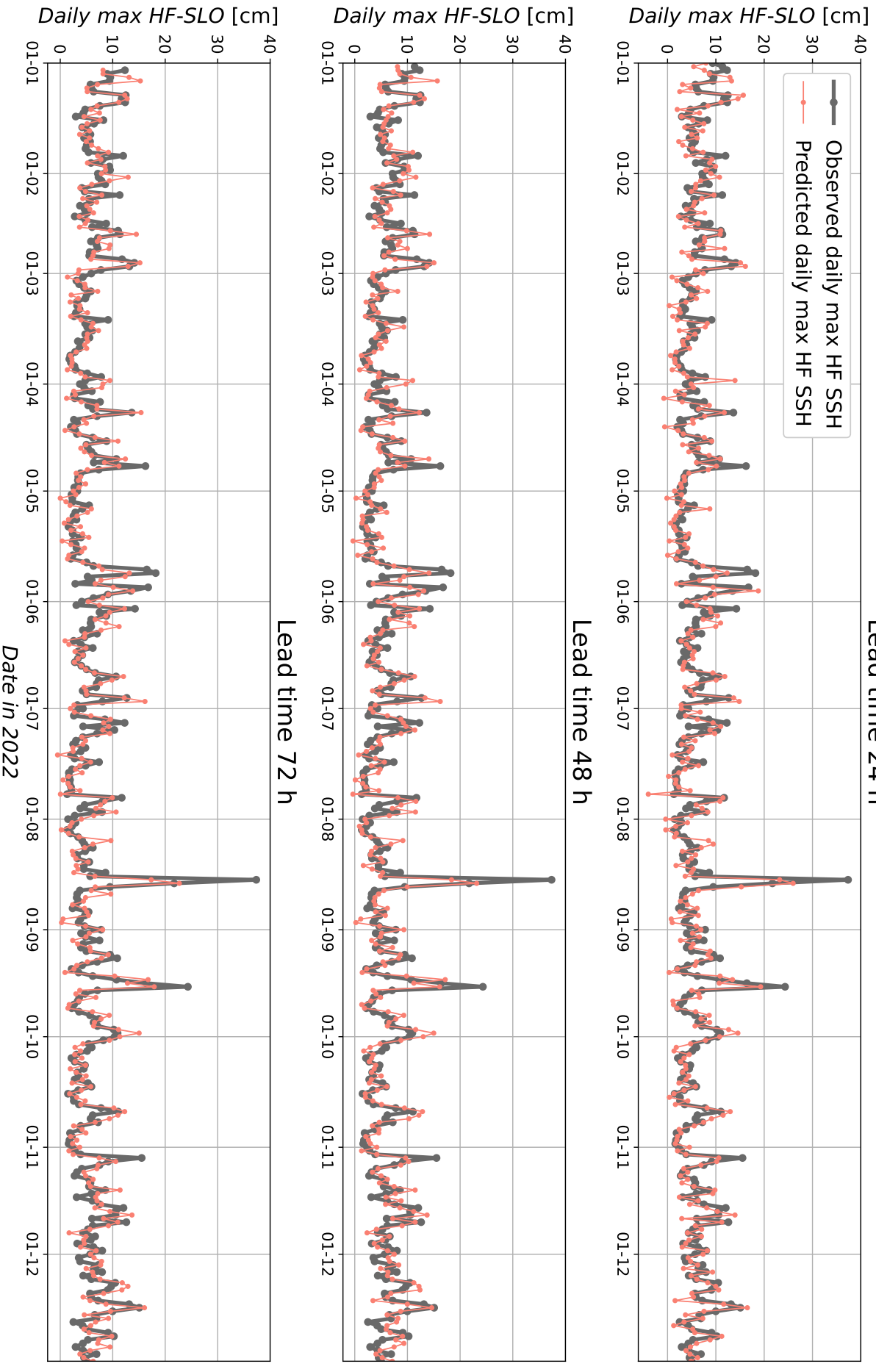

Figure 16: *Time series comparison of forecasted versus observed daily maxima of HF-SLO for 2022. The figure shows the model's performance across three lead times: 24 hours, 48 hours, and 72 hours. It highlights the model's accuracy in capturing extreme events and its tendency to underestimate peak values as the forecast horizon extends.*

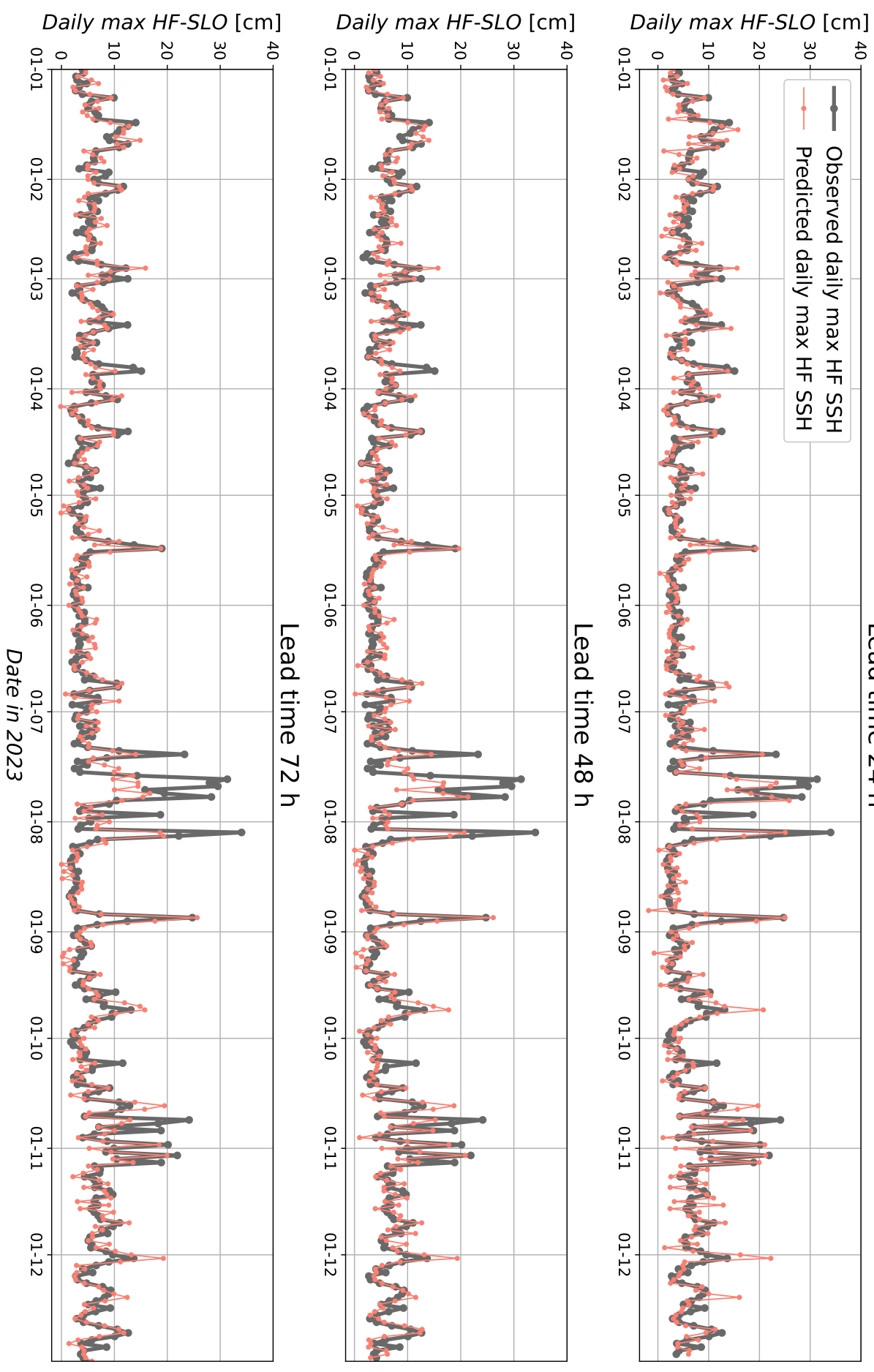

Figure 17: *Same as Figure 16 but for the year 2023.*

I quantitatively assessed the performance of the model across different lead times using four metrics: RMSE, MAE, bias, and accuracy. I calculated these metrics for the overall forecasts, as well as separately for non-extreme and extreme events, as summarized in Table 3.

Table 3: *Model performance metrics across different forecast lead times. This table shows RMSE, MAE, bias, and accuracy for lead times of 24, 48, and 72 hours, calculated for the overall forecasts, non-extreme, and extreme events. The best scores are formatted in bold.*

Lead time	Metric	Overall	Non-extremes	Extremes
24 h	RMSE [cm]	**2.70**	2.52	**7.20**
	MAE [cm]	**1.93**	**1.86**	**5.63**
	Bias [cm]	−0.14	**-0.04**	**-4.91**
	Accuracy (%)	90.11	84.01	**60.00**
48 h	RMSE [cm]	2.82	2.56	8.72
	MAE [cm]	2.02	1.91	6.96
	Bias [cm]	**-0.04**	0.09	−6.24
	Accuracy (%)	90.25	82.33	46.67
72 h	RMSE [cm]	2.84	**2.49**	9.75
	MAE [cm]	1.99	1.87	7.64
	Bias [cm]	−0.05	0.10	−7.04
	Accuracy (%)	**90.93**	**84.43**	40.00

For the overall forecasts, RMSE values indicate an increase in forecast errors as the lead time extends, rising from 2.70 cm at 24 h to 2.84 cm at 72 h. This trend is more pronounced for extreme events, where the RMSE increases significantly, from 7.20 cm at 24 h to 9.75 cm at 72 h - a deterioration of over 26% over this period. Conversely, for non-extreme events, the RMSE remains relatively stable across all forecast lead times, ranging from 2.49 cm at 72 h to 2.56 cm at 48 h. The MAE metric follows a similar trend, with errors increasing as the lead time extends. The MAE for the overall forecasts increases from 1.93 cm at 24 h to 1.99 cm at 72 h. While the MAE for non-extreme events remains relatively stable, it rises sharply for extreme events, from 5.63 cm at 24 h to 7.64 cm at 72 h.

The bias metric reveals a slight overall underestimation by the model, which is more pronounced for extreme events. At the 24-hour lead time, the bias is -0.14 cm for the overall forecasts, which is larger than at 48 h and 72 h. Although the bias for the overall forecasts and non-extreme events is close to zero, which is satisfactory, the bias for extreme events becomes significantly negative, ranging from -4.91 cm at 24 h to -7.04 cm at 72 h. This increasing negative bias with lead time suggests that the model increasingly underestimates extreme HF-SLO amplitudes as the forecast horizon extends.

Forecasting accuracy overall and for the non-extreme events remains consistently high, with accuracy slightly improving at the 72-hour lead time. Specifically, overall accuracy exceeds 90% for all lead times. However, a notable decline in accuracy is observed for extreme events, decreasing from 60% at a 24-hour lead time to 40% at a 72-hour lead time. The accuracy thresholds for evaluating forecasts were established based on one standard deviation of the

ground truth values from the testing period. These thresholds were set at 4.39 cm for the overall performance, 3.27 cm for non-extreme, and 5.26 cm for extreme events.

Figure 18 provides a scatter plot of forecasted versus observed maximum daily HF-SLO amplitudes for the three lead times (24 h, 48 h, and 72 h). Each lead time is represented by a different colour, enabling a clear visual comparison across the forecast horizons. The scatter plot highlights the overall forecasting accuracy of the model, with points closer to the diagonal line indicating better predictions. For the 24-hour lead time, the scatter plot shows a stronger linear relationship between the forecasted and observed values, particularly for extreme events, indicating better performance in short-term forecasting. The spread of points around the diagonal is lower for extreme events at 24 h, suggesting lower variance and higher accuracy. As the lead time increases to 48 h and 72 h, the spread of points broadens for higher amplitude events, indicating a decline in forecasting accuracy and exposing the systematic bias in the model's forecasts. This aligns with the findings from the time series analysis, where the model's performance deteriorates with increasing lead time, especially for extreme events. For non-extreme events, the spread of points around the diagonal remains similar across all three forecast lead times.

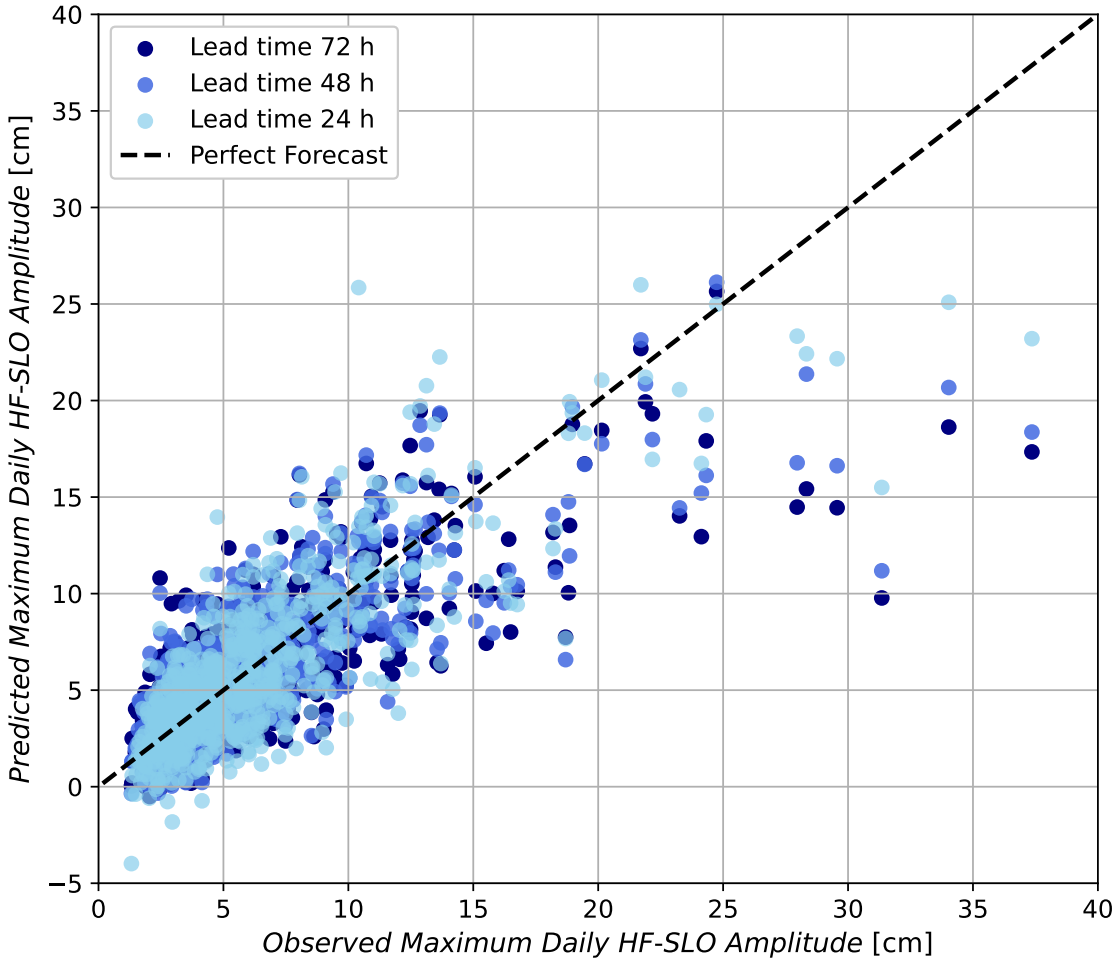

Figure 18: *Scatter plot comparing forecasted versus observed maximum daily HF-SLO amplitudes for 2022 and 2023 across lead times of 24, 48, and 72 hours. Different colors represent the different lead times, with closer alignment to the diagonal representing higher accuracy.*

To assess the robustness of the model, I trained the best-performing model 10 times, each with different initial weights and biases. This experiment revealed that initialization can significantly impact model performance, especially in deep learning. Figure 19 illustrates the MAE across different sea level amplitude bins for each training run. While the MAE is consistently low for lower amplitudes (up to 20 cm), it shows greater variability at higher amplitudes (25 cm and above), indicating increased sensitivity to initialization for extreme events. Notably, in the 25-30 cm bin, the difference in MAE between the best and worst models is significant, with values ranging from 5.98 cm to 10.99 cm. Among the 10 models, the one with the lowest MAE for extreme events (the blue line) was chosen as the best one.

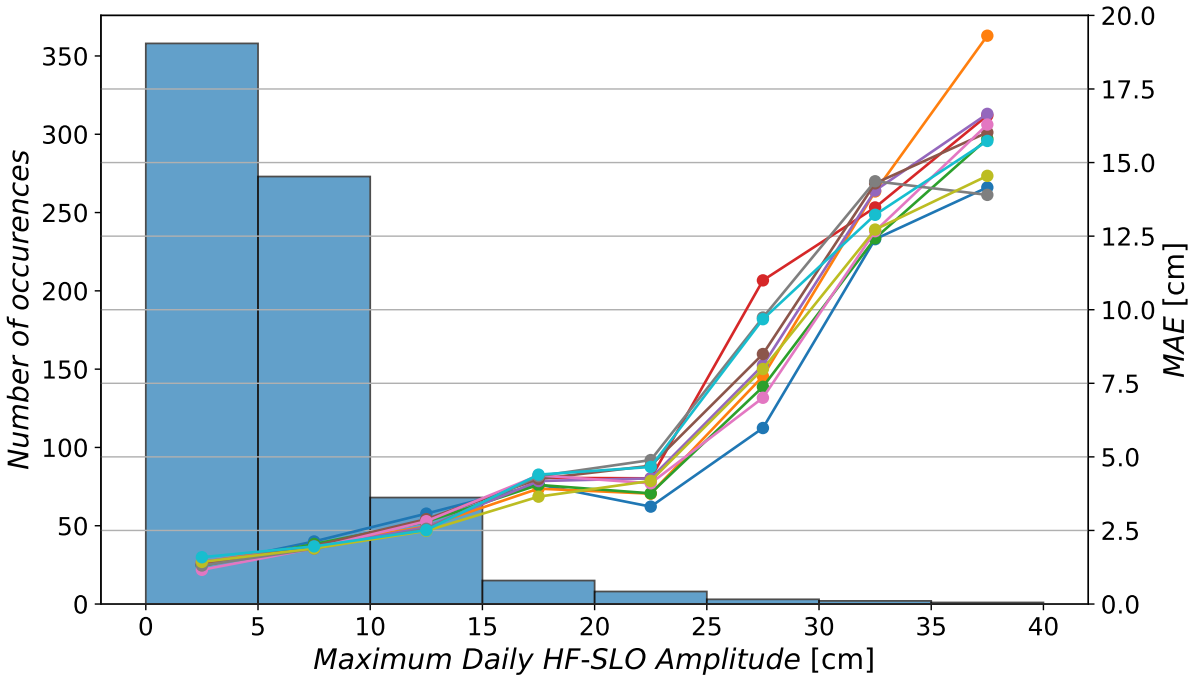

Figure 19: *Histogram showing the Mean Absolute Error (MAE) across different sea level amplitude bins for 10 training runs with varying initial weights and biases.*

4.2 Ablation study

I conducted an ablation study to evaluate the impact of input data and the oversampling technique on the model's performance. The study involved removing elements such as the Richardson number, the frequency spectrum, and the 10% oversampling individually in separate experiments, retraining the model in each case, and comparing the results to the best-performing model (analyzed in the previous subsection, hereafter referred to as $\text{Model}_{\text{Best}}$). Training and testing were consistent with the original setup, using data from 2003–2020 for training and independent testing set from January 1st, 2022, to December 31st, 2023.

Figure 20 presents the time series forecasts from May to August 2023, a period rich in extreme events, allowing for a clear comparison of performance across models with different

ablations at a 24-hour lead time. The best-performing model $\text{Model}_{\text{Best}}$ and the „No Richardson number" model ($\text{Model}_{\text{noRi}}$) forecasted extreme events more accurately than the „No Spectrum" ($\text{Model}_{\text{noSpec}}$) and „No Oversampling" ($\text{Model}_{\text{noOS}}$) models, with $\text{Model}_{\text{Best}}$ and $\text{Model}_{\text{noRi}}$ performing similarly.

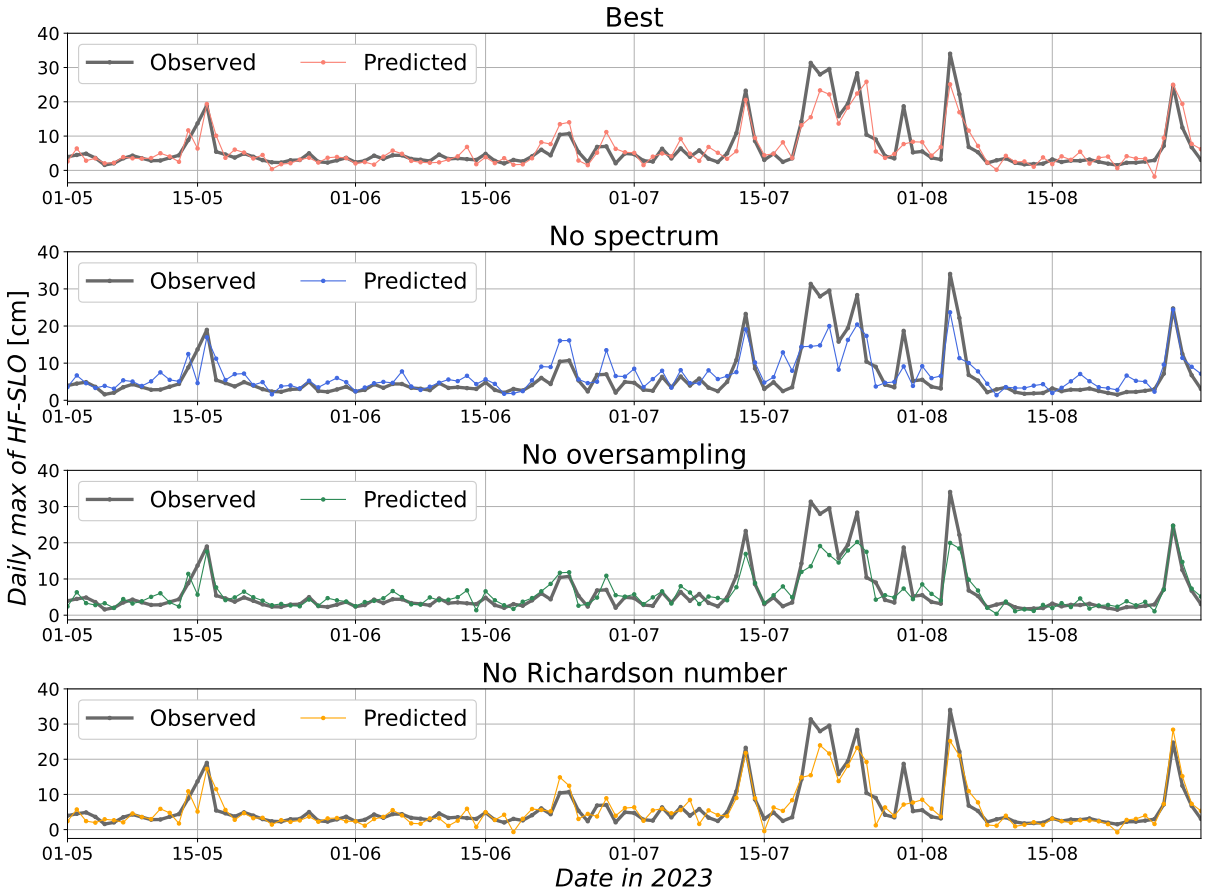

Figure 20: *Time series comparison of forecasts and ground truth from May to August 2023, illustrating the performance of different model variants after ablation at a 24-hour lead time.*

Table 4 further illustrates the differences between these models across performance metrics. Despite $\text{Model}_{\text{noRi}}$ having the same accuracy for extreme events (60%) and slightly lower RMSE and MAE overall and for non-extremes, I selected the $\text{Model}_{\text{Best}}$ as the optimal model as it outperforms the others in RMSE and MAE for extreme events. $\text{Model}_{\text{Best}}$ also outperforms $\text{Model}_{\text{noRi}}$ in 9 out of 15 extreme events in terms of error, compared to 6 out of 15 for $\text{Model}_{\text{noRi}}$. This focus on extreme events was crucial in the model selection.

$\text{Model}_{\text{noSpec}}$ variant underperformed compared to the $\text{Model}_{\text{Best}}$ in all metrics, making it the worst-performing among these four models. $\text{Model}_{\text{noOS}}$ excelled in forecasting non-extreme events, showing the lowest errors for these cases due to the lack of oversampling, which allowed it to focus more on non-extremes. However, removing the spectrum and oversampling led to a significant performance drop in forecasting extremes, with RMSE increasing by 29% and 34%, respectively, confirming their essential role in accurate forecasts. Conversely, removing the Richardson number had the least impact, resulting in a 3% RMSE increase for extreme events.

Table 4: *Model performance metrics for different variants in the ablation study, including overall, non-extreme, and extreme events. Metrics reported are RMSE, MAE, bias, and accuracy. The best scores are formatted in bold.*

Model variant	Metric	Overall	Non-extremes	Extremes
Best	RMSE [cm]	2.70	2.52	**7.20**
	MAE [cm]	1.93	1.86	**5.63**
	Bias [cm]	−0.14	**-0.04**	−4.91
	Accuracy (%)	90.11	84.01	**60.00**
No spectrum	RMSE [cm]	2.87	2.57	9.28
	MAE [cm]	2.10	1.99	7.53
	Bias [cm]	0.60	0.76	−6.92
	Accuracy (%)	89.70	82.47	40.00
No oversampling	RMSE [cm]	**2.53**	**2.14**	9.64
	MAE [cm]	**1.72**	**1.59**	7.74
	Bias [cm]	**-0.01**	0.15	−7.42
	Accuracy (%)	**94.37**	**88.78**	40.00
No Richardson number	RMSE [cm]	2.63	2.42	7.44
	MAE [cm]	1.85	1.77	5.96
	Bias [cm]	−0.55	−0.46	**-4.83**
	Accuracy (%)	91.76	85.27	**60.00**

Bias results also reveal important differences between the models. $\text{Model}_{\text{noSpec}}$ exhibits a positive bias of 0.60 cm overall, which is higher than $\text{Model}_{\text{Best}}$, and indicates a tendency to overestimate values in non-extreme cases. $\text{Model}_{\text{noOS}}$, while showing the lowest bias overall, demonstrates a significantly higher negative bias (-7.42 cm) for extreme events, reflecting its weaker performance in forecasting extremes. In contrast, $\text{Model}_{\text{Best}}$'s bias showcases its more balanced performance across all scenarios.

Figure 21 displays the MAE values across different HF-SLO amplitude bins for each model variant. $\text{Model}_{\text{Best}}$ generally showed the lowest errors across the bins, particularly in the higher amplitude ranges, though all models performed solidly at lower amplitudes. The difference is most notable in the 25-30 cm bin, where $\text{Model}_{\text{Best}}$'s MAE ($5.98$ cm) was significantly lower than that of the $\text{Model}_{\text{noSpec}}$ (10.22 cm), representing an increase of approximately 71%.

4.3 The impact of oversampling, batch size, domain and smoothing

I conducted alternative training configurations to evaluate the impact of varying oversampling ratios (20%, 30%, and 50%), batch sizes (64 and 256), spatial domain (Mediterranean), and frequency spectrum smoothing on the model's performance. These were done in separate experiments. I retrained the model in each case and compared the results to the best-performing model, $\text{Model}_{\text{Best}}$. Training and testing were consistent with the original setup, as described in the previous subsection.

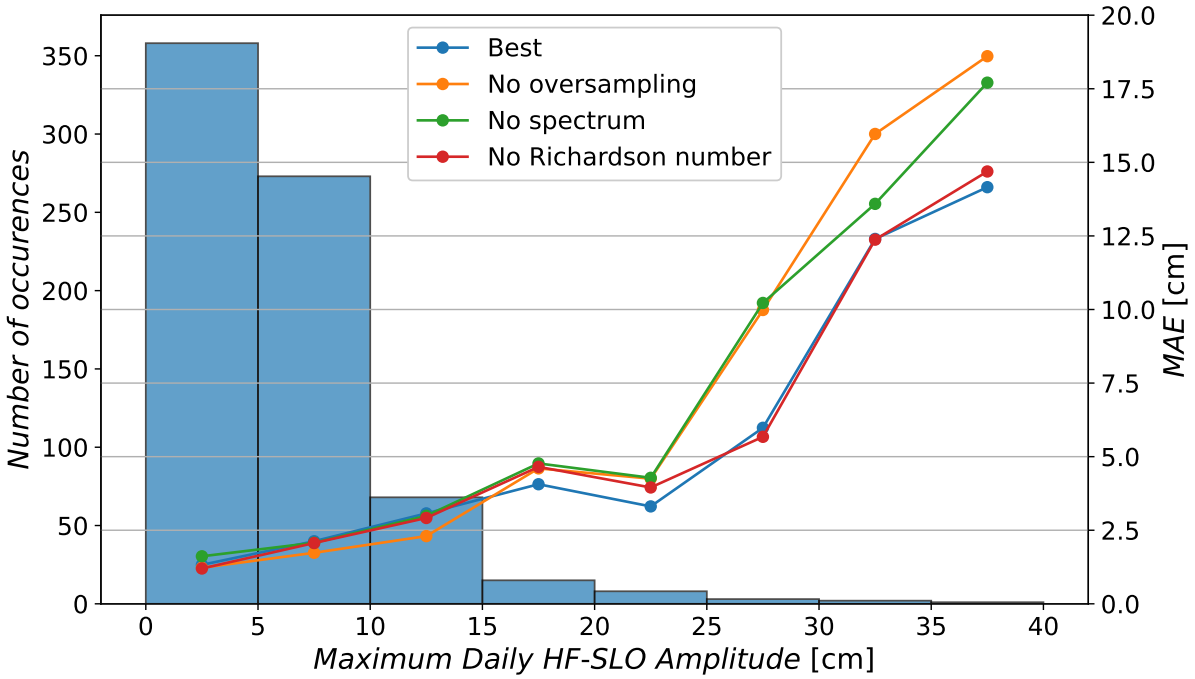

Figure 21: *Mean Absolute Error (MAE) values across different HF-SLO amplitude bins for each model variant in the ablation study.*

Figure 22 also presents the time series forecasts from May to August 2023 across models with different modifications at a 24-hour lead time. Visually, the "Oversampling=30%" ($\text{Model}_{\text{OS30}}$) and "Batch size=64" ($\text{Model}_{\text{BS64}}$) models performed most similarly to $\text{Model}_{\text{Best}}$. All other models performed worse at forecasting extreme events, although they generally performed satisfactorily for lower amplitudes. Notably, the "Mediterranean domain" model ($\text{Model}_{\text{Med}}$) visually underperformed in the lower amplitude ranges.

Table 5 further illustrates the differences between these models across the performance metrics. After $\text{Model}_{\text{Best}}$, models $\text{Model}_{\text{BS64}}$ and $\text{Model}_{\text{Med}}$, both with 53.33% forecasting accuracy performed best. The other models, such as $\text{Model}_{\text{OS20}}$, $\text{Model}_{\text{OS50}}$, $\text{Model}_{\text{BS256}}$, and $\text{Model}_{\text{UnSpec}}$, each achieved 40% accuracy for extreme events. For non-extreme events, $\text{Model}_{\text{BS256}}$ reached the highest accuracy of 86.26%, with the other models showing slightly lower but still satisfactory accuracies. The greatest increase in RMSE for extreme events was seen in $\text{Model}_{\text{UnSpec}}$ and $\text{Model}_{\text{OS50}}$, with a rise of approximately 17% compared to $\text{Model}_{\text{Best}}$.

Bias analysis reveals that most models tend to overestimate the target values, showing a positive bias in forecasting, except for $\text{Model}_{\text{Best}}$ and $\text{Model}_{\text{Med}}$, which exhibited a slight negative or neutral bias overall and for non-extreme events. The lowest bias in forecasting extreme events was recorded by $\text{Model}_{\text{BS64}}$ (-4.61 cm), followed by $\text{Model}_{\text{Best}}$ (-4.91 cm). The highest bias for extreme events was observed in $\text{Model}_{\text{Med}}$ (-6.08 cm) and $\text{Model}_{\text{UnSpec}}$ (-6 cm).

Figure 23 also displays the MAE values across different HF-SLO amplitude bins for each model variant. $\text{Model}_{\text{Best}}$ generally showed the lowest errors across the bins, especially in the higher amplitude ranges (20-30 cm bins and 35-40 cm bin), although it underperformed compared to $\text{Model}_{\text{Med}}$ and $\text{Model}_{\text{BS256}}$ in the 30-35 cm bin. However, these two models generally showed less effective performance across the other bins. All models performed consistently well at lower amplitudes, with $\text{Model}_{\text{Med}}$ showing the weakest performance here.

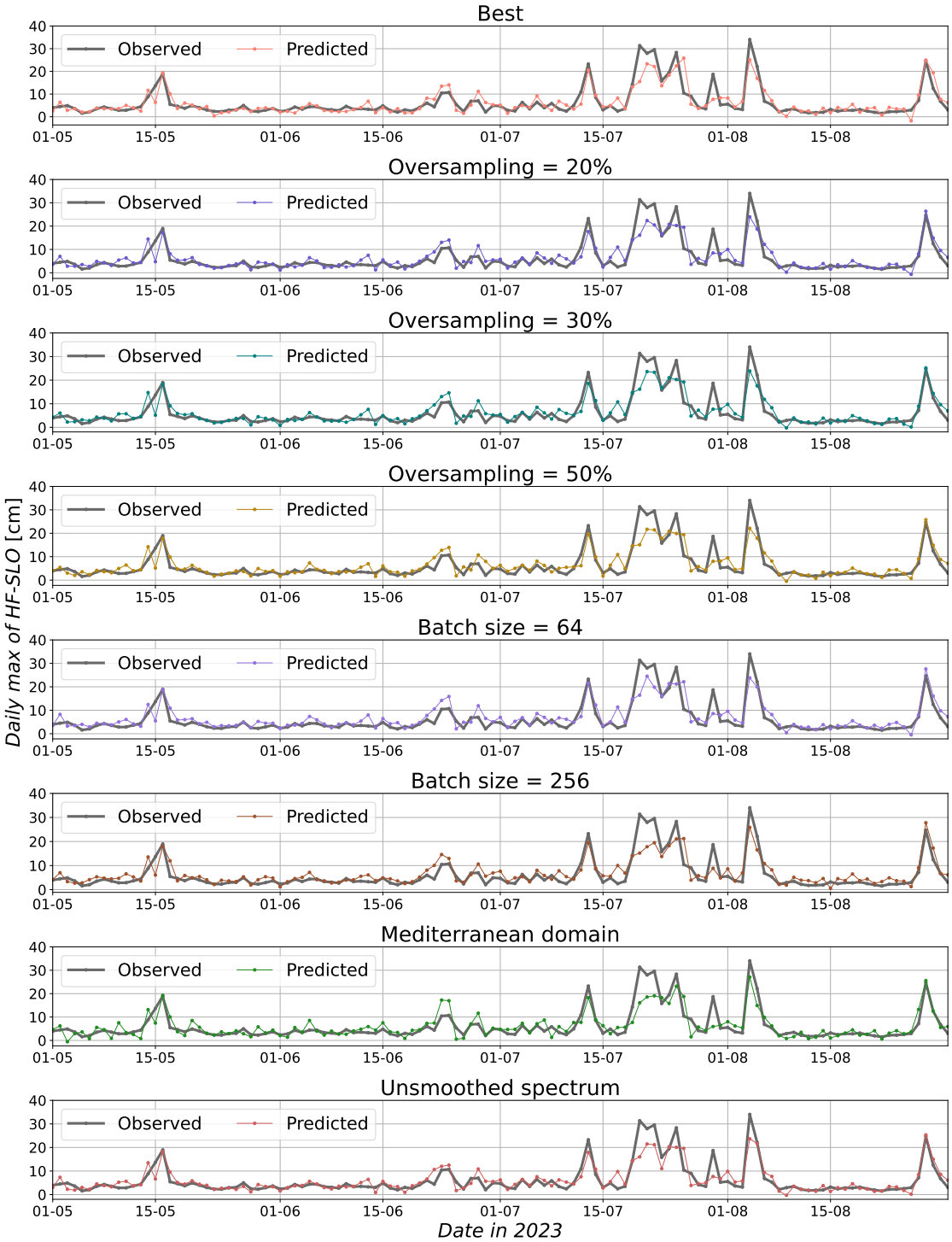

Figure 22: *Time series comparison of forecasts and ground truth from May to August 2023, illustrating the performance of different model variants with various modifications at a 24-hour lead time during alternative training configurations.*

Table 5: *Model performance metrics for different variants in the alternative training configuration study, including overall, non-extreme, and extreme events. Metrics reported are RMSE, MAE, bias, and accuracy. The best scores are formatted in bold.*

Model variant	Metric	Overall	Non-extremes	Extremes
Best	RMSE [cm]	2.70	2.52	**7.20**
	MAE [cm]	1.93	1.86	**5.63**
	Bias [cm]	−0.14	−0.04	−4.91
	Accuracy (%)	90.11	84.01	**60.00**
Oversampling=20%	RMSE [cm]	2.70	2.47	8.11
	MAE [cm]	1.96	1.86	6.62
	Bias [cm]	0.35	0.47	−5.57
	Accuracy (%)	90.66	84.15	40.00
Oversampling=30%	RMSE [cm]	2.71	2.49	7.79
	MAE [cm]	1.95	1.86	6.34
	Bias [cm]	0.28	0.39	−5.13
	Accuracy (%)	91.48	83.73	46.67
Oversampling=50%	RMSE [cm]	2.71	2.45	8.40
	MAE [cm]	1.92	1.81	6.90
	Bias [cm]	0.19	0.32	−5.75
	Accuracy (%)	**92.03**	84.71	40.00
Batch size=64	RMSE [cm]	2.75	2.54	7.80
	MAE [cm]	2.01	1.93	6.20
	Bias [cm]	0.81	0.93	**-4.61**
	Accuracy (%)	91.21	83.31	53.33
Batch size=256	RMSE [cm]	**2.62**	**2.37**	8.12
	MAE [cm]	1.89	1.79	6.65
	Bias [cm]	0.30	0.43	−5.82
	Accuracy (%)	91.90	**86.26**	40.00
Mediterranean domain	RMSE [cm]	2.81	2.59	7.99
	MAE [cm]	2.02	1.92	6.57
	Bias [cm]	−0.12	**0.00**	−6.08
	Accuracy (%)	90.11	84.29	53.33
Unsmoothed spectrum	RMSE [cm]	2.64	**2.37**	8.44
	MAE [cm]	**1.86**	**1.76**	6.69
	Bias [cm]	**0.10**	0.23	−6.00
	Accuracy (%)	91.35	84.01	40.00

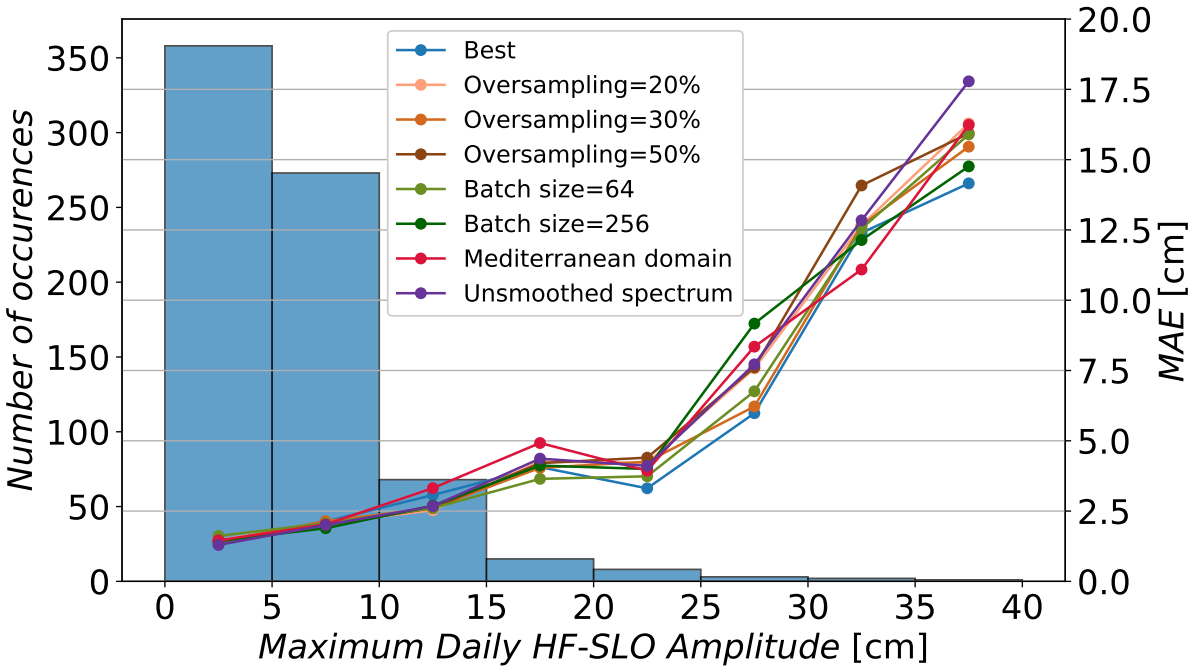

Figure 23: *Mean Absolute Error (MAE) values across different HF-SLO amplitude bins for each model variant in the alternative training configuration study.*

5 Discussion and Conclusion

High-frequency sea level oscillations are rapid fluctuations in sea level that occur over short time scales, typically ranging from minutes to a few hours. These oscillations can be driven by various factors, including atmospheric disturbances such as pressure changes, wind bursts, and seismic activity. Key phenomena include seiches - standing waves in enclosed or semi-enclosed basins, and meteotsunamis - long ocean waves generated by atmospheric disturbances such as gravity waves or pressure jumps. Unlike predictable tidal changes, these oscillations are sudden and can cause significant short-term sea level variations, especially in low-tidal regions such as the Mediterranean, where they pose risks of coastal flooding and damage, making accurate forecasts crucial.

In this thesis, I developed a deep learning model capable of forecasting the maximum daily height of high-frequency sea level oscillations in Bakar Bay. The model demonstrated high accuracy for non-extreme events and showed promising potential for forecasting extreme events, highlighting the capability of deep learning models to achieve even greater precision with further refinement. The results validate the effectiveness of the proposed model, particularly in its handling of extreme events. Among the various configurations explored, I identified the best-performing model, which integrated atmospheric and tide gauge data with an oversampling technique that ensured adequate exposure to extreme events during training.

The performance analysis across different forecast lead times showed that the model was most effective in short-term forecasts. At a 24-hour lead time, the model achieved an RMSE of 2.70 cm, an MAE of 1.93 cm, and a bias of -0.14 cm. Its accuracy in forecasting extreme events was 60%, outperforming other configurations. However, as the forecast horizon extended to 72 hours, performance declined, with RMSE increasing to 2.84 cm and accuracy for extreme events dropping to 40%. This decline is consistent with the inherent uncertainty of longer-range forecasts, a common challenge in predictive modeling.

The ablation study highlights the critical importance of specific model components. The inclusion of the frequency spectrum and the applied oversampling technique significantly enhanced the model's ability to forecast extreme events. Conversely, removing the Richardson number had the least impact, likely because the atmospheric variables it represents were already included independently in the model. These findings highlight the value of incorporating diverse and relevant input features and the role of oversampling in improving the model's capacity to forecast these rare events. The best-performing model consistently outperformed others in forecasting extreme events.

Further analysis of alternative training configurations highlighted the influence of training parameters on model performance. Adjustments to oversampling ratios and batch sizes had notable effects, with the best performance observed when using a 30% oversampling ratio and a batch size of 64, after the best-performing model. The Mediterranean domain model, however,

underperformed, suggesting that a smaller spatial domain, such as the Adriatic, is sufficient for accurate forecasts in this context.

The downsampling in the architecture was performed primarily to reduce the dimensionality of the input feature maps, making model training more memory-efficient within the available computational resources. However, since max-pooling was applied as the first operation on the raw input data, it may not have had the benefit of learned context to determine which features are most relevant. As a result, max-pooling simply reduces the input by retaining the maximum values in each pooling window, which could potentially discard lower but significant values, such as a low Richardson number indicating atmospheric instability. This trade-off was necessary to manage the computational demands of the model, but it does highlight a potential area for further optimization.

Future research could benefit from integrating real-time mesoscale atmospheric data that could capture atmospheric disturbances on smaller spatial scales to further enhance the accuracy of forecasts, particularly for extreme events. Extending the input data time range to include longer historical data windows may also improve the model's understanding of temporal dynamics, though previous studies [68] suggest that this approach may not always yield better results. The broader significance of this research lies in its potential to influence the development of more accurate early warning systems for coastal hazards, particularly in regions vulnerable to extreme sea level events.

Acknowledgments

Sea level data used in this research were measured by the Department of Geophysics, Faculty of Science, University of Zagreb. The collection and financing of the data are supported by the projects StVarAdri and CroClimExtremes, funded by the Croatian Science Foundation under grant numbers HRZZ-IP-2019-04-5875 and HRZZ-IP-2022-10-4144.

„ERA5 hourly data on single levels from 1940 to present" and „ERA5 hourly data on pressure levels from 1940 to present" datasets were downloaded from the Copernicus Climate Change Service (2023). The results contain modified Copernicus Climate Change Service information 2020. Neither the European Commission nor ECMWF is responsible for any use that may be made of the Copernicus information or data it contains.

Computational resources used in this research were provided by the Slovenian Environment Agency (ARSO). The GPU used for this work is the property of ARSO, and their support is gratefully acknowledged.

Bibliography

[1] P. Arias et al., "Climate change 2021: The physical science basis. contribution of working group i to the sixth assessment report of the intergovernmental panel on climate change; technical summary," 2021. URL: https://www.ipcc.ch/report/ar6/wg1/.

[2] S. Vitousek, P. Barnard, C. Fletcher, N. Frazer, L. Erikson, and C. Storlazzi, "Doubling of coastal flooding frequency within decades due to sea-level rise," Scientific Reports, vol. 7, 2017. DOI: https://doi.org/10.1038/s41598-017-01362-7.

[3] M. Taherkhani, S. Vitousek, P. Barnard, N. Frazer, T. Anderson, and C. Fletcher, "Sea-level rise exponentially increases coastal flood frequency," Scientific Reports, vol. 10, 2020. DOI: https://doi.org/10.1038/s41598-020-62188-4.

[4] P. Woodworth, M. Menendez, and W. Gehrels, "Evidence for Century-Timescale Acceleration in Mean Sea Levels and for Recent Changes in Extreme Sea Levels," Surveys in Geophysics, vol. 32, pp. 603–618, 2011. DOI: https://doi.org/10.1007/s10712-011-9112-8.

[5] W. Sweet and J. Park, "From the extreme to the mean: Acceleration and tipping points of coastal inundation from sea level rise," Earth's Future, vol. 2, 2014. DOI: https://doi.org/10.1002/2014EF000272.

[6] C. Ferrarin et al., "Integrated sea storm management strategy: the 29 October 2018 event in the Adriatic Sea," Natural Hazards and Earth System Sciences, vol. 20, no. 1, pp. 73–93, 2020. DOI: https://doi.org/10.5194/nhess-20-73-2020.

[7] M. Menendez and P. Woodworth, "Changes in extreme high water levels based on a quasi-global tide-gauge data set," Journal of Geophysical Research, vol. 115, 2010. DOI: https://doi.org/10.1029/2009JC005997.

[8] B. D. Hamlington et al., "Understanding of Contemporary Regional Sea-Level Change and the Implications for the Future," Reviews of Geophysics, vol. 58, no. 3, 2020. DOI: https://doi.org/10.1029/2019RG000672.

[9] D. Pugh and P. Woodworth, "Sea-Level Science: Understanding Tides, Surges, Tsunamis and Mean Sea-Level Changes". National Oceanography Centre, Liverpool: Cambridge University Press, 2014. DOI: https://doi.org/10.1017/CBO9781139235778.

[10] P. Woodworth, "Seiches in the Eastern Caribbean," Pure and Applied Geophysics, vol. 174, 2017. DOI: https://doi.org/10.1007/s00024-017-1715-7.

[11] A. Rabinovich, "Twenty-Seven Years of Progress in the Science of Meteorological Tsunamis Following the 1992 Daytona Beach Event," Pure and Applied Geophysics, vol. 177, 2020. DOI: https://doi.org/10.1007/s00024-019-02349-3.

[12] S. Monserrat, I. Vilibić, and A. B. Rabinovich, "Meteotsunamis: atmospherically induced destructive ocean waves in the tsunami frequency band," Natural Hazards and Earth System Sciences, vol. 6, no. 6, pp. 1035–1051, 2006. DOI: https://doi.org/10.5194/nhess-6-1035-2006.

[13] W. Munk, F. Snodgrass, and G. Carrier, "Edge Waves on the Continental Shelf," Science, vol. 123, no. 3187, pp. 127–132, 1956. DOI: https://doi.org/10.1126/science.123.3187.127.

[14] A. E. Yankovsky, "Long-Wave Response of the West Florida Shelf to the Landfall of Hurricane Wilma, October 2005," Journal of Coastal Research, vol. 24, pp. 33 – 39, 2008. DOI: https://doi.org/10.2112/06-0824.1.

[15] S. Henderson and A. Bowen, "Simulations of Dissipative, Shore-Oblique Infragravity Waves," Journal of Physical Oceanography, vol. 33, 2003. DOI: https://doi.org/10.1175/2398.1.

[16] J. Aucan and F. Ardhuin, "Infragravity waves in the deep ocean: An upward revision," Geophysical Research Letters, vol. 40, no. 13, pp. 3435–3439, 2013. DOI: https://doi.org/10.1002/grl.50321.

[17] G. Dodet, A. Melet, F. Ardhuin, X. Bertin, D. Idier, and R. Almar, "The Contribution of Wind-Generated Waves to Coastal Sea-Level Changes," Surveys In Geophysics, vol. 40, no. 6, pp. 1563–1601, 2019. DOI: https://doi.org/10.1007/s10712-019-09557-5.

[18] I. Vilibić and J. Šepić, "Global mapping of nonseismic sea level oscillations at tsunami timescales," Scientific Reports, vol. 7, 2017. DOI: https://doi.org/10.1038/srep40818.

[19] T. Hibiya and K. Kajiura, "Origin of the Abiki phenomenon (a kind of seiche) in Nagasaki Bay," Journal of the Oceanographical Society of Japan, vol. 38, pp. 172–182, 1982. URL: https://api.semanticscholar.org/CorpusID:198197231.

[20] A. Rabinovich and S. Monserrat, "Generation of Meteorological Tsunamis (Large Amplitude Seiches) Near the Balearic and Kuril Islands," Natural Hazards, vol. 18, pp. 27–55, 1998. DOI: https://doi.org/10.1023/A:1008096627047.

[21] A. Rabinovich, Handbook of Coastal and Ocean Engineering, pp. 193–236. 2009. DOI: https://doi.org/10.1142/9789812819307_0009.

[22] M. Orlić, "The first attempt at cataloguing tsunami-like waves of meteorological origin in Croatian coastal waters," Acta Adriatica, vol. 56, pp. 83–95, 2015. https://hrcak.srce.hr/148129.

[23] J. Šepić, I. Vilibić, A. Rabinovich, and S. Monserrat, "Widespread tsunami-like waves of 23-27 June in the Mediterranean and Black Seas generated by high-altitude atmospheric forcing," Scientific Reports, vol. 5, p. 11682, 2015. DOI: https://doi.org/10.1038/srep11682.

[24] R. Plougonven and F. Zhang, "Internal gravity waves from atmospheric jets and fronts," Reviews of Geophysics, vol. 52, no. 1, pp. 33–76, 2014. DOI: https://doi.org/10.1002/2012RG000419.

[25] J. Proudman, "The Effects on the Sea of Changes in Atmospheric Pressure," Geophysical Supplements to the Monthly Notices of the Royal Astronomical Society, vol. 2, no. 4, pp. 197–209, 1929. DOI: https://doi.org/10.1111/j.1365-246X.1929.tb05408.x.

[26] H. P. Greenspan, "The generation of edge waves by moving pressure distributions," Journal of Fluid Mechanics, vol. 1, pp. 574–592, 1956.

[27] Brnas, Tina and Ruić, Krešimir and Šepić, Jadranka and Balić, Marijana and Pupić Vurilj, Mia, "Mediterranean meteotsunamis of May 2021 and June 2022: Observations, data analysis and synoptic background," Geofizika, vol. 40, 2023. DOI: https://doi.org/10.15233/gfz.2023.40.8.

[28] H. Pellikka, J. Šepić, I. Lehtonen, and I. Vilibić, "Meteotsunamis in the northern Baltic Sea and their relation to synoptic patterns," Weather and Climate Extremes, 2022. DOI: https://doi.org/10.1016/j.wace.2022.100527.

[29] B. Galperin, S. Sukoriansky, and P. S. Anderson, "On the critical Richardson number in stably stratified turbulence," Atmospheric Science Letters, vol. 8, no. 3, pp. 65–69, 2007. DOI: https://doi.org/10.1002/asl.153.

[30] Skybrary, "Richardson number." URL: https://skybrary.aero/articles/richardson-number, 2023. Accessed: 2024-08-13.

[31] J. Šepić and M. Pervan, "Analysis of the eastern Adriatic sea level extremes," St open, vol. 4, pp. 1–19, 2023. DOI: https://doi.org/10.48188/so.4.10.

[32] I. Fine, A. Rabinovich, R. Thomson, and E. Kulikov, Submarine Landslides and Tsunamis, pp. 69–88. 2003.

[33] B. Levin and M. Nosov, Physics of Tsunamis, pp. 1–327. 2009.

[34] C. Ramis and A. Jansà, "Condiciones meteorológicas simultáneas a la aparición de oscilaciones del nivel del mar de amplitud extraordinaria en el Mediterraneo occidental," Revista de Geofísica, no. 39, pp. 35–42, 1983.

[35] A. Jansa and C. Ramis, "The Balearic rissaga: from pioneering research to present-day knowledge," Natural Hazards, vol. 106, 2021. DOI: https://doi.org/10.1007/s11069-020-04221-3.

[36] M.-d.-M. Vich and R. Romero, "Forecasting meteotsunamis with neural networks: the case of Ciutadella harbour (Balearic Islands)," Natural Hazards, vol. 106, 2021. DOI: https://doi.org/10.1007/s11069-020-04041-5.

[37] J. Šepić, I. Vilibić, and S. Monserrat, "Quantifying the probability of meteotsunami occurrence from synoptic atmospheric patterns," Geophysical Research Letters, vol. 43, 2016. DOI: https://doi.org/10.1002/2016GL070754.

[38] J. Šepić, M. Orlić, and I. Vilibić, "The Bakar Bay seiches and their relationship with atmospheric processes," Acta Adriatica, vol. 49, no. 2, pp. 107–123, 2008. URL: https://hrcak.srce.hr/31813.

[39] M. Kasumović, "Harmonička analiza plime i oseke luke Bakar," Radovi Geofizičkog instituta u Zagrebu, vol. 3, no. 1, pp. 1–9, 1952.

[40] M. Orlić and M. Pasarić, "Vodostaj Jadranskog mora i globalne klimatske promjene," Pomorski zbornik, vol. 32, no. 1, pp. 481–501, 1994.

[41] M. Pasarić and M. Orlić, "Long-term meteorological preconditioning of the North Adriatic coastal floods," Continental Shelf Research, vol. 21, p. 263–278, 2001. DOI: https://doi.org/10.1016/S0278-4343(00)00078-9.

[42] I. Međugorac, M. Pasarić, and M. Orlić, "Long-term measurements at Bakar tide-gauge station (east Adriatic)," Geofizika, vol. 39, no. 1, pp. 149–162, 2022. DOI: https://doi.org/10.15233/gfz.2022.39.8.

[43] M. Rus, A. Fettich, M. Kristan, and M. Ličer, "HIDRA2: deep-learning ensemble sea level and storm tide forecasting in the presence of seiches – the case of the northern Adriatic," Geoscientific Model Development, vol. 16, no. 1, pp. 271–288, 2023. DOI: https://doi.org/10.5194/gmd-16-271-2023.

[44] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.

[45] E. Alpaydin, Introduction to Machine Learning, third edition. Adaptive Computation and Machine Learning series, MIT Press, 2014.

[46] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. Campbell, "Introduction to Machine Learning, Neural Networks, and Deep Learning," Translational Vision Science & Technology, vol. 9, 2020. URL: https://api.semanticscholar.org/CorpusID:216022295.

[47] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, pp. 436–444, 2015. DOI: https://doi.org/10.1038/nature14539.

[48] C. Cao, F. Liu, H. Tan, D. Song, W. Shu, W. Li, Y. Zhou, X. Bo, and Z. Xie, "Deep Learning and Its Applications in Biomedicine," Genomics, Proteomics Bioinformatics, vol. 16, pp. 17–32, 2018. DOI: https://doi.org/10.1016/j.gpb.2017.07.003.

[49] J. Patterson and A. Gibson, Deep Learning: A Practitioner's Approach. O'Reilly Media, Inc., 1st ed., 2017.

[50] Cornell University, "Neural Networks and Machine Learning." URL: https://blogs.cornell.edu/info2040/2015/09/08/neural-networks-and-machine-learning/, 2015. Accessed: 2024-08-23.

[51] G. Ajmera, "Decoding Activation Functions: Navigating the Landscape for Image Segmentation." URL: https://medium.com/@girishajmera/decoding-activation-functions-navigating-the-landscape-for-image-segmentation-f091e971e475, 2019. Accessed: 2024-08-23.

[52] M. Z. Alom, T. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. Nasrin, M. Hasan, B. Van Essen, A. Awwal, and V. Asari, "A State-of-the-Art Survey on Deep Learning Theory and Architectures," Electronics, vol. 8, p. 292, 2019. DOI: https://doi.org/10.3390/electronics8030292.

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," CoRR, 2015. URL: https://arxiv.org/abs/1512.03385.

[54] S. Ruder, "An overview of gradient descent optimization algorithms," CoRR, 2017. URL: https://arxiv.org/abs/1609.04747.

[55] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2017. URL: https://arxiv.org/abs/1412.6980.

[56] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," CoRR, 2019. URL: https://arxiv.org/abs/1711.05101.

[57] Educative.io, "Overfitting and Underfitting." URL: https://www.educative.io/answers/overfitting-and-underfitting, 2024. Accessed: 2024-08-23.

[58] Javatpoint, "Overfitting in Machine Learning." URL: https://www.javatpoint.com/overfitting-in-machine-learning, 2024. Accessed: 2024-08-23.

[59] R. Lakshya, "Dropout Regularization." URL: https://ruhelalakshya.medium.com/dropout-regularization-b27885b4c55b, 2021. Accessed: 2024-08-23.

[60] Analog Devices, Inc., "What the Nyquist Criterion Means to Your Sampled Data System Design." URL: https://www.analog.com/media/en/training-seminars/tutorials/MT-002.pdf, 2024. Accessed: 2024-08-12.

[61] J. Goldberg and K. Kempni, "On Bakar-Bay oscillations and general problem of bay seiches (in Croatian)," Prirodoslovna istraživanja Kraljevine Jugoslavije, vol. 21, pp. 129–234, 1938.

[62] H. Hersbach et al., "The ERA5 global reanalysis," Quarterly Journal of the Royal Meteorological Society, vol. 146, pp. 1999–2049, 2020. DOI: https://doi.org/10.1002/qj.3803.

[63] H. Hersbach et al., "ERA5 hourly data on pressure levels from 1940 to present." Copernicus Climate Change Service (C3S) Climate Data Store (CDS), 2023. DOI: https://doi.org/10.24381/cds.bd0915c6. Accessed on 01-04-2024.

[64] H. Hersbach and other, "ERA5 hourly data on single levels from 1940 to present." Copernicus Climate Change Service (C3S) Climate Data Store (CDS), 2023. DOI: https://doi.org/10.24381/cds.adbb2d47. Accessed on 01-04-2024.

[65] NOAA Atlantic Oceanographic and Meteorological Laboratory, "Potential Temperature," 2023. URL: https://www.aoml.noaa.gov/ftp/pub/hrd/annane/prelim_notes/Potential_Temperature.pdf. Accessed: 2024-08-13.

[66] NVIDIA Corporation, "NVIDIA RTX A5000," 2023. URL: https://www.nvidia.com/en-us/design-visualization/rtx-a5000/. Accessed: 2024-08-12.

[67] NVIDIA Corporation, "PyTorch Glossary," 2023. URL: https://www.nvidia.com/en-us/glossary/pytorch/. Accessed: 2024-08-12.

[68] L. Žust, A. Fettich, M. Kristan, and M. Ličer, "Hidra 1.0: Deep-learning-based ensemble sea level forecasting in the northern adriatic," 2020. DOI: https://doi.org/10.5194/gmd-14-2057-2021.

Appendix A

```
def butter_bandpass(lowcut, highcut, fs, order):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    sos = butter(order, [low, high], analog=False, btype='band',
    output='sos')
    return sos

def butter_bandpass_filter(data, lowcut, highcut, fs, order):
    sos = butter_bandpass(lowcut, highcut, fs, order=order)
    y = sosfiltfilt(sos, data)
    return y

def compute_psd(times, sealevels):
    # Computes Power Spectral Density (PSD)
    import numpy.fft as fft
    # Remove NaNs
    timesnn = times[~np.isnan(sealevels)]
    sealevelsnn = sealevels[~np.isnan(sealevels)]

    n = len(timesnn)
    t = range(n)

    dataFFT = abs(fft.fft(sealevelsnn))
    dataF = fft.fftfreq(sealevelsnn.size, t[1]-t[0])
    periods = 1. / dataF[range(int(n/2))]
    spectrum = dataFFT[range(int(n/2))]

    return periods, spectrum
```

Appendix B

```
import torch, torch.nn as nn

class Model(nn.Module):
    def __init__(self):
        super().__init__()

        drop = .1

        # 4D INPUTS
        self.atmo_spatio_temporal = nn.ModuleList()
        for i in range(6):
            self.atmo_spatio_temporal.append(nn.Sequential(
                nn.MaxPool3d(kernel_size=(1, 2, 2), stride=(1, 2, 2)),
                nn.LazyConv3d(64, kernel_size=(2, 3, 3),
                stride=(2, 3, 3)), nn.ReLU(), nn.Dropout3d(drop),
                nn.LazyConv3d(256, kernel_size=(1, 3, 3),
                stride=(1, 3, 3)), nn.ReLU(), nn.Dropout3d(drop),
                nn.LazyConv3d(512, kernel_size=(1, 3, 4)),
            ))

        self.atmo_temporal = nn.LazyConv1d(256, kernel_size=5)

        self.atmo_residual = nn.ModuleList([
            nn.Sequential(nn.LazyConv1d(256, kernel_size=1),
            nn.SELU(), nn.Dropout1d(drop)),
            nn.Sequential(nn.LazyConv1d(256, kernel_size=1),
            nn.SELU(), nn.Dropout1d(drop)),
        ])

        # 1D INPUTS
        self.dim1 = 2 ** 9
        self.dim2 = 2 ** 13

        self.atmo_ssh = nn.ModuleList([
            nn.LazyLinear(self.dim1),
            nn.LazyLinear(self.dim1),
            nn.LazyLinear(self.dim1),
            nn.ModuleList([
                nn.Sequential(nn.LazyLinear(3 * self.dim1), nn.SELU(),
                nn.Dropout(drop)),
                nn.Sequential(nn.LazyLinear(3 * self.dim1), nn.SELU(),
                nn.Dropout(drop)),
                nn.Sequential(nn.LazyLinear(3 * self.dim1), nn.SELU(),
                nn.Dropout(drop)),
                nn.Sequential(nn.LazyLinear(3 * self.dim1), nn.SELU(),
```

```
                nn.Dropout(drop)),
            ]),
            nn.LazyLinear(self.dim2),
        ])

        # FINAL
        self.final = nn.LazyLinear(3)

    def forward(self, q, u, v, Ri, t, msl, z, sshhf_input, sshhf_spectrum):

        batch_size = q.shape[0]

        # PROCESSING 4D INPUTS
        encodings = []
        for i, x in enumerate(((u, v), q, Ri, t, msl, z)):
            if type(x) == tuple:
                x = torch.cat(x, dim=2)
            if len(x.shape) == 4:
                x = x[:, :, None]
            x = x.permute(0, 2, 1, 3, 4)
            x = self.atmo_spatio_temporal[i](x)
            x = x.squeeze(4).squeeze(3)
            encodings.append(x)

        # joining and applying temporal block
        x = torch.cat(encodings, dim=1)
        x = x.view(batch_size, x.size(1), -1)
        x = self.atmo_temporal(x)
        for layer in self.atmo_residual:
            x = x + layer(x)
        features_from_4D_inputs = x.view(batch_size, -1)

        # PROCESSING 1D INPUTS
        x = torch.cat((
            self.atmo_ssh[0](sshhf_input),
            self.atmo_ssh[1](sshhf_spectrum),
            self.atmo_ssh[2](features_from_4D_inputs),
        ), dim=1)
        for layer in self.atmo_ssh[3]:
            x = x + layer(x)
        features = self.atmo_ssh[4](x)

        # FINAL
        x = self.final(features)

        return x
```