Sea Level Anomaly Prediction in the Gulf Stream using Neural Network Approaches

Ocean Current Forecasting in the North Atlantic Ocean

Jules Merigot

Master's of Computer Science from Paris Dauphine University - PSL

A Master's Thesis Report submitted to the MIDO Department of Computer Science of Paris Dauphine University in Partial Fulfillment of the Requirements for the Degree of

Master IASD, Artificial Intelligence and Data Science

Paris Dauphine University - PSL
Paris, France

Supervisors:	Carlos E. Mejia, Sylvie Thiria, Anastase Charantonis
Advisor:	Luther Ollier
Referent Professor:	Benjamin Negrevergne

September 1, 2024

Acknowledgements

First and foremost I would like to thank my supervisors, Sylvie Thiria and Carlos Mejia, for their guidance and support throughout my end-of-studies research internship. Furthermore, I would like to extend a big thank you to my advisor, Luther Ollier, for his incessant wisdsom, counsel, and recommendations as I advanced through my research. I could not have done it without all of you.

I would also like to thank the Sorbonne Center for Artificial Intelligence (SCAI) for funding this internship and for providing the resources necessary to complete this work, as well as Le Laboratoire D'océanographie Et Du Climat (LOCEAN) of the CNRS and Sorbonne University for welcoming me within their laboratory and providing me with the best work environment possible, as well as all the computational and storage materials necessary to complete my research.

Abstract

Sea Level Anomaly (SLA) is an indicator of the sub-mesoscale dynamics of the upper ocean, and reveals the regional extent of anomalous water levels in the ocean which can indicate unusual water temperatures, salinities, and currents. In this study, we focused on the temporal evolution of SLA fields, complimented with Sea Surface Temperature (SST) data. SST is driven by these ocean dynamics and can be used to improve the spatial interpolation of SLA fields. We specifically explored the potential of Deep Learning (DL) solutions with Attention-based methods to forecast short-term SLA fields using SST fields as a complimentary indicator of the SLA dynamics. Our work serves as a proof of concept, therefore we worked with simulated daily SLA and SST data from the Mercator Global Analysis and Forecasting System, with a resolution of $\frac{1}{12}^\circ$ in the North Atlantic Ocean ($26.5 - 44.42°$N, -$64.25 - 41.83°$E), covering the period from 1993 to 2019. Using a modified image-to-image convolutional DL architecture with attention-based modules, we demonstrated that SST is a relevant variable for controlling the SLA prediction, and managed to improve the SLA forecast at 5 days by using the SST fields as additional information.

In order to push our research further, we investigated the implementation of a Physics-informed Neural Network (PINN) to improve predictions of physical phenomenons such as SLA and SST. PINNs integrate physical laws, including fluid dynamics and boundary conditions, directly into the learning process. This improves the model's ability to generalize from limited data and ensures that predictions comply with established physical laws and principles. We sought to create a PINN to model a simple, dynamic fluid flow of temperature using physical constraints implemented via a partial differential equation (PDE), with the objective of approaching an improved SLA and SST forecast. By striving to model this PDE along with other notable fluid equations, we furthered our understanding of modeling physical systems. Through this work, we gained significant insight into the inner workings of building and training PINNs for dynamic fluid systems, and set the groundwork for future investigation and testing.

Keywords—Deep-learning, Sea Surface Temperature, Forecast, Attention, Transformers, Physics.

Contents

Chapter 1

Introduction

Gaining a better understanding of upper-ocean dynamics is vital not only for comprehending our oceans' roles in the Earth's climate but also for various practical applications, such as optimizing navigation routes or advancing marine engineering. Measurements stemming from multi-satellite observations such as satellite altimetry have significantly contributed to this effort, providing extensive data on oceanic parameters like Sea Level Anomaly (SLA) and Sea Surface Height (SSH). SLA, which measures the difference between the mean sea surface height and the height recorded by the satellite altimeters, helps identify numerous oceanic features, such as eddies and ocean fronts [1], and informs researchers and oceanographers about global ocean circulation. Through the principle of geostrophic balance, SLA fields can be used to determine sea surface currents and identify mesoscale structures like eddies [2],[3]. These are crucial characteristics that can lead to a better understanding of our oceans' effects on our planet.

However, altimetry products are derived from interpolating data between different satellites over time [4]. As a result, enhancing the resolution of SLA and surface currents, as well as predicting short-term surface changes, remains a significant area of research [5],[6],[7]. Nevertheless, other well-known satellite-measured parameters, such as Sea Surface Temperature (SST), can aid in this effort. SST is recorded using multi-sensor measurements that offers high spatiotemporal resolution and is an effective tracer for advection, a crucial factor in SLA evolution. Numerous studies have demonstrated a strong association between SLA and SST through complex functions [8],[9].

Mathematical methods based on physical principles have been developed to enhance the interpolation of SLA fields [10]. In contrast, data-driven approaches, such as deep learning algorithms, can infer the underlying state of the system that drives ocean surface parameters. In this case, neural networks are particularly effective in processing large volumes of ocean data. Previous research has demonstrated that deep learning methods can improve SLA resolution using SST data [11], although these methods did not consider the temporal evolution of SLA and SST [12]. Additionally, predictive neural networks have been developed to forecast SLA at a small resolution ($\frac{1}{4}^{\circ}$) without incorporating other parameters like SST [13],[14]. To solve for this issue, we can implement a novel convolutional neural network to project SLA fields while using SST as additional control data.

1.1 Sea Level Anomaly Forecasting

Our work focuses on the Gulf Stream, which is a warm and swift Atlantic ocean current that originates in the Gulf of Mexico and flows up the eastern coastline of the United States, then veers east and moves toward Northwest Europe as the North Atlantic Current. This current carries warm, salty water northward along the

Northeast shelf effectively bringing heat from the tropics to high latitudes. Much of the more temperate and warm climate present in Northern Europe compared to their North American counterparts is attributed to this intense stream of warm water. Due to its influence on the climate, this makes the Gulf Stream a crucial part of the Atlantic Ocean and a key area of research in the pursuit of a better understanding of its dynamics. Reliable forecasts of SLA fields in the Gulf Stream would allow for significant advancements in this area of research and an improved comprehension for oceanographers.

Building on a previous research internship, this study aims to advance SLA forecasting in the Gulf Stream by incorporating attention-based modules within a deep learning architecture. Specifically, we utilize spatial and channel attention mechanisms to capture both spatial differences and temporal dependencies in our time-series data fields, thereby enhancing prediction accuracy. By examining the correlated temporal evolution of SLA and SST, we demonstrate that it is possible to refine SLA predictions using SST at a high resolution ($\frac{1}{12}^{\circ}$). We implemented a convolutional network algorithm based on the UNet architecture due to its robustness and stability in image processing.

1.2 Physics-informed Neural Networks

In the strive towards an even further improved modeling of this physical system, we pursued research within the field of physics-informed machine learning, specifically, Physics-informed Neural Networks (PINN). Unlike traditional neural networks that rely solely on data, PINNs incorporate physical laws directly into the learning process which not only enhances the model's ability to generalize from limited data, but also ensures that the predictions adhere to known physical laws and principles [15]. By embedding these constraints, PINNs offer a robust framework for solving complex differential equations and modeling physical systems where data might be noisy, such as oceanic systems [16]. The combination of attention mechanisms with physics-informed methods also presents a novel approach for developing more accurate and reliable predictive models in scientific and engineering domains [17]. Our objective was to use known partial differential equations for a dynamic temperature fluid flow in order to approach a modeling similar to that of our SST images in the Gulf Stream. We sought to connect the analysis of our attention-based SLA/SST forecasting model with a PINN that could forecast SST fields while restricting its predictions within physical bounds. Through our research during this project, we aim to highlight the significance of the implementation of Physics-Informed Neural Networks in our ocean-modeling field, and illustrate their potential to revolutionize computational modeling in fields such as fluid dynamics and climate science.

1.3 Report Organization

This work begins with a brief discussion of related work and the state of the art in both climate modeling and neural network methods for satellite image analysis, as well as a background on PINNs and advent in the field of machine learning. In chapter 3, we will outline our data preprocessing procedure fit for our simulated satellite data, as well as present the methods by which we achieved our SLA forecasting results, which includes the training process as well as the design of the neural network model itself. Chapter 4 presents our experimental work and our custom training methods. In chapter 5, we display our two main results: the forecast performance achieved through our training methodology, and the effectiveness of using SST to project SLA, as well discuss these results and their significance relative to other works. We then advance to our research on PINNs and their relation to our previous work for the chapter 6. This will be accompanied by various tested partial differential equations and what conclusions we can draw based on our investigation for future work. We will finally conclude the overall research of this thesis in the final chapter.

Chapter 2

Related Work

In this chapter, we will give an overview of relevant work related to this project, as well as the current state of the art in these fields. In order to lay the groundwork for our research, we will describe overviews for both weather forecasting and satellite imagery models, along with advancements in the domain of physics-informed neural networks.

2.1 Weather Forecasting

Weather forecasting using Machine Learning (ML) approaches has been a growing field of research in recent years. It traces its origins to the early use of statistical techniques which aimed to find patterns and correlations in historical weather data to make future predictions. With the advent of more powerful computers, feedforward and recurrent neural networks were explored for their ability to model the complex relationships found in weather patterns. This was briefly followed by a revolution with deep learning models like CNNs and Long-short term memory (LSTM) [18], which could handle large, high-resolution datasets to make more accurate forecasts. More recently, graph neural networks and hybrid models combining numerical weather prediction (NWP) with machine learning have further advanced the field, such as Google's previous leading model; GraphCast [19]. An attention-based model named SmaAt-UNet from 2021 [20] seeked to improve the lacking ability of numerical methods to make short-term forecasts using the latest available information. The creators of this model equipped the deep learning UNet architecture with attention modules to improve now-casting results on real-life datasets using precipitation maps in Europe, allowing it to achieve comparable predictions to other models while using only a quarter of the trainable parameters.

The current state of the art in weather forecasting belongs to the GenCast model developed by Google DeepMind in 2023 [21]. This is a diffusion-based probabilistic weather forecasting method for different weather states and a multitude of variables, specifically for medium-range weather predictions. This work stands out due to its ability to generate accurate predictions of weather patterns up to 15 days in advance using satellite imagery at a global scale with a 1-degree resolution. The key innovation in GenCast is its use of a diffusion model to generate the weather forecasts. It works by sampling from a noise distribution and iteratively refining these samples to approximate the true distribution of weather states, thus allowing it to maintain physical consistency and capture intricate weather patterns more effectively than previous models. Such a model could be applied to SLA forecasting as well since it uses satellite imagery as data, but diffusion-based models were slightly out of the picture for this research project. Another project developed by Google DeepMind for the purpose of creating a benchmark for data-driven weather forecasting models, is called WeatherBench 2 [22]. WeatherBench 2 serves as an open-source framework for evaluating weather prediction models using publicly available training and baseline data, and constantly updated state-of-the-art

models. All based upon metrics to evaluate performance established by leading world weather centers, this platform operates with a goal of accelerating progress in physical and data-driven weather modeling. Both of these models exceed the scope of the particular research of this report, but are key resources for current state-of-the-art and future weather modeling projects.

Regarding the specific work of forecasting SLA fields, the previous work done by Ollier et al. [23] had produced promising and accurate results when predicting 5, 10, 15, and 20 days ahead. His work employed a deep residual UNet which he modified to fit the needs of his research. We seek to improve upon his work by introducing attention-based modules within the UNet architecture similar to the SmaAt-UNet model in order to select key patterns in the feature maps and enhance the quality of our forecasts.

2.2 Physics-informed Machine Learning

Recent advances in deep learning have revolutionized the fields of computer visions and image analysis, leading to a desire to find more specific solution of particular problems. Notably, the merging of machine learning and scientific computing have led to the increased modeling of physical systems via data-driven methods rather than classical numerical methods. This has led to the emergence of physics-guided machine learning, and to the advent of physics-informed neural networks (PINNs) [15], which introduced a novel approach for solving forward and inverse problems involving partial differential equations (PDEs). For an in-depth breakdown of the formulation of PINNs, it is recommended to read the original paper by Raissi et al. [15]. These deep learning models are known for their capability to seamlessly incorporate noisy experimental data and physical laws into the learning process. This is accomplished by parameterizing unknown functions of interest using deep neural networks and formulating a multi-task learning problem with the aim of matching observational data and approximating an underlying PDE system [24]. However, due to the unknown nature of certain functions, this has caused many challenges in their progress, which has in turn driven researchers to explore systems modeled by known physical equations.

The combination of physics-based learning with data-driven modeling has motivated researchers to pursue the goal of modeling popular dynamical fluid systems. Ocean systems, for example, rely on known marine physics PDEs that formulate the dynamics of ocean models [25]. In one instance, published in 2023, an article explored a novel physics-based deep learning framework, named the space-time PDE-guided neural network (STPDE-Net) [26], to predict daily SST by employing a space-time partial differential equation. Another article focusing on bridging the gap between partial differential equations and trainable neural networks for practical purposes was published by Meteo France in 2021. Through the demonstration of their proposed model, *PDE-NetGen 1.0* [27], for various use cases such as Burger's Equation, they illustrate a successful training workflow. They effectively further proved the effectiveness of introducing physical knowledge in the design of a neural network to improve its modeling performances. In fact, the modeling of Burger's equation is a task we tackle ourselves later in this report, and explain the difference and similarities of our experiences compared to other articles. While the modeling methods we explored in our research focused mainly on a simple temperature advection equation, we will later discuss further investigation we conducted into the modeling of more complicated dynamical fluid equations such as the Navier-Stokes equations or the aforementioned Burger's equation.

Chapter 3

Materials and Methods

This chapter will introduce and briefly analyze the materials and data that was utilized throughout our research, alongside a description of the preprocessing steps. This will be followed by a thorough discussion of the methodology used in the building of the model and an analysis of the model architecture itself.

3.1 Materials

The Copernicus GLORYS12V1 Mercator model product [28] provides datasets from various daily parameters on a $\frac{1}{12}^{\circ}$ grid. Our primary study focused on the zone of the North Atlantic Ocean highlighted and illustrated in Figure 3.1 covering the area from 26.5° to 44.42° latitude North and -64.25° to -53.58° longitude East. The SLA and SST data selected span from 1993 to 2019, encompassing approximately 10,000 daily images. The Gulf Stream flows through this area of the Atlantic Ocean, creating a more defined dynamical evolution of sea level anomaly (SLA) and sea surface temperature (SST) in the northern part of the region. In contrast, the southern region experiences less extreme variations and smaller eddies. Therefore, we will focus on the following area for our study:

- **Northern Region:** Latitude from 33.7° to 44.42° N, representing the dynamic area influenced by the Gulf Stream flow in the Northern Atlantic Ocean.

Our results focus on the northern area, which exhibits more robust activity and is therefore more interesting to investigate with regards to gaining a better understanding of current dynamics. We effectively trained our models strictly on this region by cropping our images prior to training. Alongside the highlight of the Gulf Stream, Figure 3.1 shows an example of a simulated SLA field image before any cropping was done to isolate the dynamic zone. It is clear that the upper part contains more activity and eddies, and is therefore more relevant to our research.

3.1.1 Time-series Data

Since we are dealing with time-series data, maintaining temporal consistency when preparing our data is of paramount importance. This includes accounting for any variability or inconsistencies in our data such as seasonal changes. Due to the initial focus of our forecast study being on short-time predictions of 5 to 10 days in advance, it is crucial that our data does not exhibit periodicity caused by seasonal variability. As can be seen in Figure 3.2, a Mann-Kendall [29],[30] seasonal test can be performed to detect trends and show that the SLA and SST have a positively increasing trend over the years from 1993 to 2019. Additionally, in the SST trend plot, the seasonal variability in temperature can be clearly seen during each year. This will be accounted for in our data preprocessing step in order to prevent any unwanted effects on our experiments.

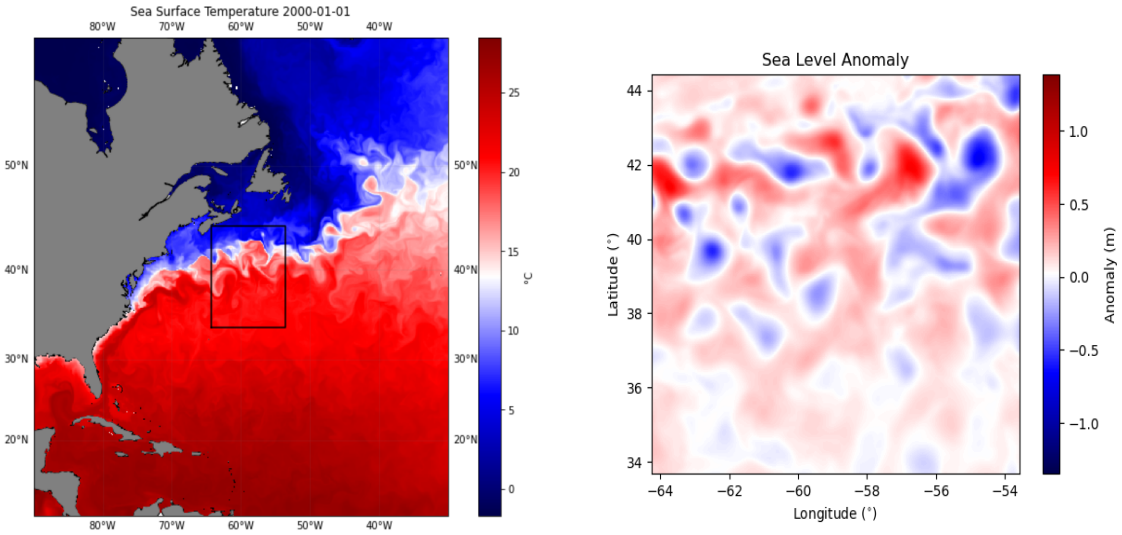

Figure 3.1: Gulf Stream Location *(left)* and Example Simulated Image of Sea Level Anomaly (SLA) from GLORYS (Mercator Océan International) *(right)*.

Since we will need to split our dataset into training, validation, and testing sets, it would be preferable to have all three sets be representative of the full data without inter-annual variability. If the test set is significantly different from the training or validation sets in terms of average temperature and anomaly height, then the precision of our results will be affected.

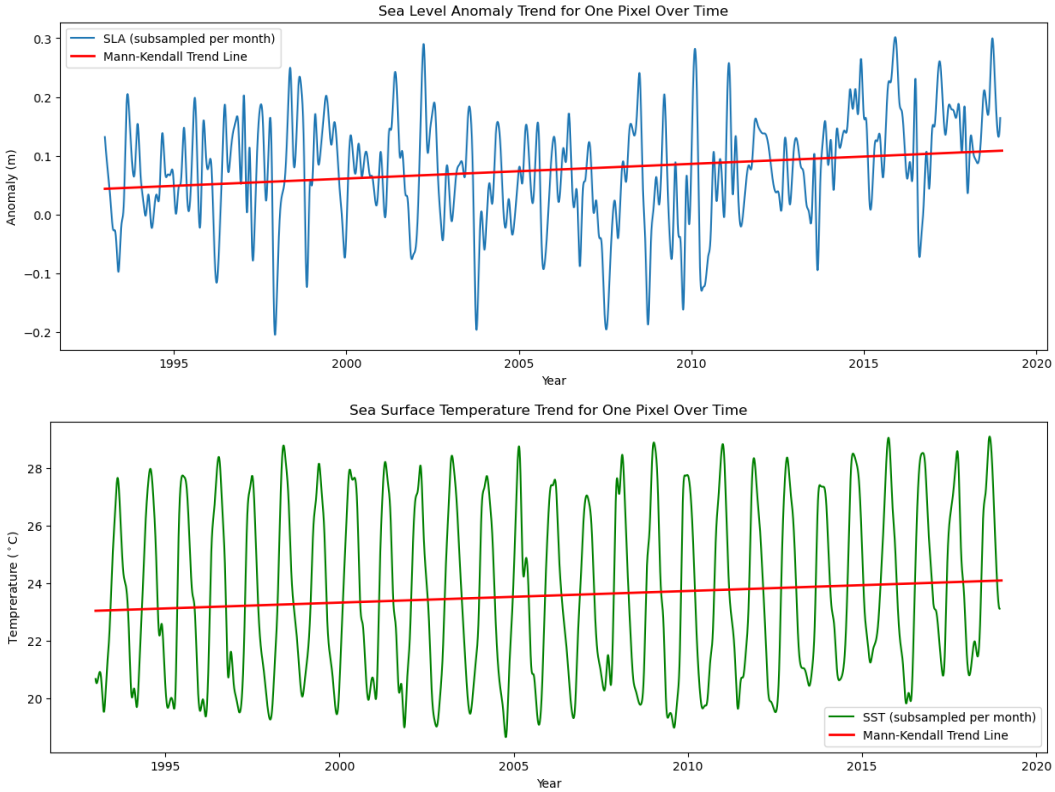

Figure 3.2: Trend of SLA (*top)* and SST *(bottom)* for one pixel in the Gulf Stream area over time from 1993 to 2019.

3.1.2 Data Preprocessing

Keeping in mind the trend of increasing temperature found via the Mann-Kendall seasonal test, it was vital to correctly split our data while maintaining temporal consistency. We separated our data into three independent datasets for training, validation and testing. The same datasets were used for all experiments presented in this study. Twenty days were removed from the learning dataset at the beginning of 1994 and at the end of 2017 to separate the test from the learning phase. The three datasets cover different periods and are normalized separately :

- **Training set :** 1994 - 2014 (7412 images)
- **Validation set :** 2015 - 2017 (1275 images)
- **Test set :** 1993, 2018, 2019 (873 images)

As previously mentioned, we focused on the northern area of the Gulf Stream where the most activity was present. During the pre-processing phase of our data, we used a transformation to crop the images to the upper left 128x128 pixels in order to isolate the northern dynamic area which is of more interest to us. Figure 3.3 depicts some visualization examples of the SLA and SST images after the transformation, which we will be using as our input data for our models. Our normalization is done independently on each data set by standard scaling each image. In the SLA images, the darker zones of both red and blue correspond to eddies and stronger oceanic currents, and thus lead to more significant anomalies. This can be seen by the colorbar measurements on the right of the images. We can also notice some corresponding similarities in major activity zones in both the SLA and SST images, further demonstrating their relation. These more dynamic zones are the most important and are key in understanding the currents. Therefore, when forecasting SLA images, we want to preserve these darker zones as much as possible in our predictions.

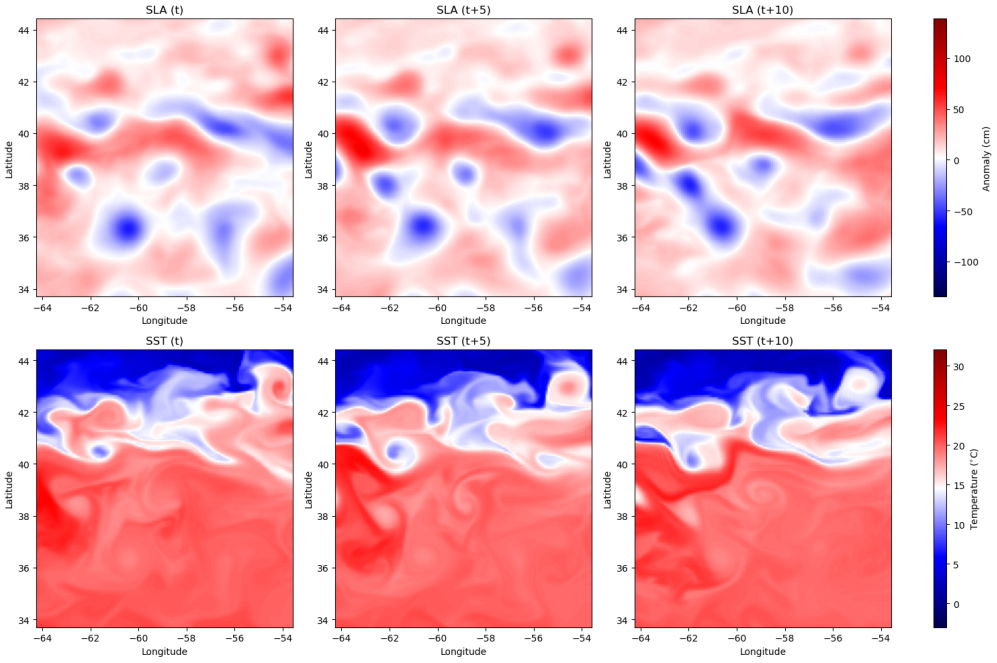

Figure 3.3: SLA example at time `t` (05/09/1993) and its evolution for `t+5` and `t+10` days ahead *(top)*, and SST example at time t (05/09/1993) and its evolution for `t+5` and `t+10` days ahead *(bottom)*, for the dynamic region of the Gulf Stream (Latitude 33.7° to 44.42° N).

3.2 Methodology

Following the work done previously by Luther Ollier outlined in Ollier et al. [23] and its success in predicting SLA fields during his own end-of-studies research internship in 2022, we decided to pursue a similar neural network approach to predict SLA fields given SLA and SST time series data.

Deep convolutional networks are specially designed to process images, using convolution operations to capture spatial context information. For this study, we selected the traditional UNet architecture for its flexibility in a wide range of image processing tasks [31]. The UNet architecture reduces the spatial resolution of the input by passing it through various layers to identify image patterns. As the network reduces the size of the input matrix, it eliminates unnecessary noise, allowing it to focus on essential patterns. Skip connections ensure that input information is propagated through these layers without being affected by the vanishing gradient problem. To further enhance the predictive capabilities of our model, we incorporated attention-based modules, specifically spatial and channel attention mechanisms, inspired by another model. These modules are designed to capture both spatial differences and temporal dependencies, thus improving prediction accuracy for time series data. By focusing on relevant features and their temporal evolution, the attention mechanisms enable the network to improve its learning of complex SLA and SST data.

3.2.1 Initial Tests and Models

The previous work done on this subject had concluded with promising results utilizing a modified residual UNet architecture using both SLA and SST data to forecast SLA images at `t+5`, `t+10`, `t+15`, and `t+20` days for a timestep `t`. Our goal was to push this further, starting by testing a modified UNet model with 3D-convolutions, as well as a model utilizing channel and spatial attention modules with a UNet architecture. While 3D-based convolutional neural network (CNN) models seemed to perform well on the data due to the inclusion of time as the third dimension, their computational inefficiency outweighed their performances. We therefore decided to opt for a 2D-based CNN but with the addition of attention-based modules, where our inherently grayscale SLA images have their sequence length (e.g. `t+5` = 5 images) become the channel by stacking the images along the channel dimension of our input tensors. Since our goal was to maintain the darker eddy zones of the SLA fields, we theorized that spatial attention would help maintain these important features, while channel attention would maintain the temporal consistency of the sequence.

3.2.2 The SmaAt-UNet Model

For the attention-based model, we decided to utilize the *SmaAt-UNet* model [20] due to its previously proven success in precipitation nowcasting tasks. While we are focusing on a forecasting goal, its performance on precipitation time-series data demonstrated in the original article make it very relevant here. Additionally, its smaller size makes it more versatile for our experiments. A traditional UNet has $17,272,577$ parameters, while the *SmaAt-UNet* has only $4,111,389$ parameters. The architecture consists of the traditional U-shaped encoder-decoder structure with four encoder-decoder modules. Encoders perform max-pooling and double convolutions to reduce image size while increasing feature maps, whereas decoders use bilinear upsampling, skip-connections, and double convolutions to reconstruct the image. This model introduces two modifications: incorporating Convolutional Block Attention Modules (CBAM) in the encoder to focus on important features, and replacing standard convolutions with depthwise-separable convolutions (DSC) to reduce parameters. DSCs break down the standard convolution into two steps: depthwise convolution followed by pointwise convolution, reducing both the number of mathematical operations and parameters compared to traditional convolutions. This architecture adaptation makes the model suitable for time-series prediction tasks by preserving original image features and highlighting relevant temporal information.

3.2.3 The SLA-SST-SmaAt-UNet Model

With slight alterations tailored to fit our needs, we were able to modify the *SmaAt-UNet* architecture to perform successfully on our data. As will be further explained in chapter 4, we wanted to test the models for SLA forecasting only, while still being able to compare model performances when the model is fed strictly SLA images as an input and when it is fed both SLA and SST images as inputs. To accommodate for this second investigation task, our modifications involved allowing the model to accept two inputs, for SLA and SST images which are then concatenated along the channel dimension, as well as modifying the number of channels within the convolutional layers of the model to accommodate the new size of the concatenated input. We additionally modified the model by extracting the last time step from the input sequences and adding it back at the end of the model's output in order to preserve temporal consistency. We called this model the *SLA-SST-SmaAt-UNet*, of which the architecture can be seen in Figure 3.4.

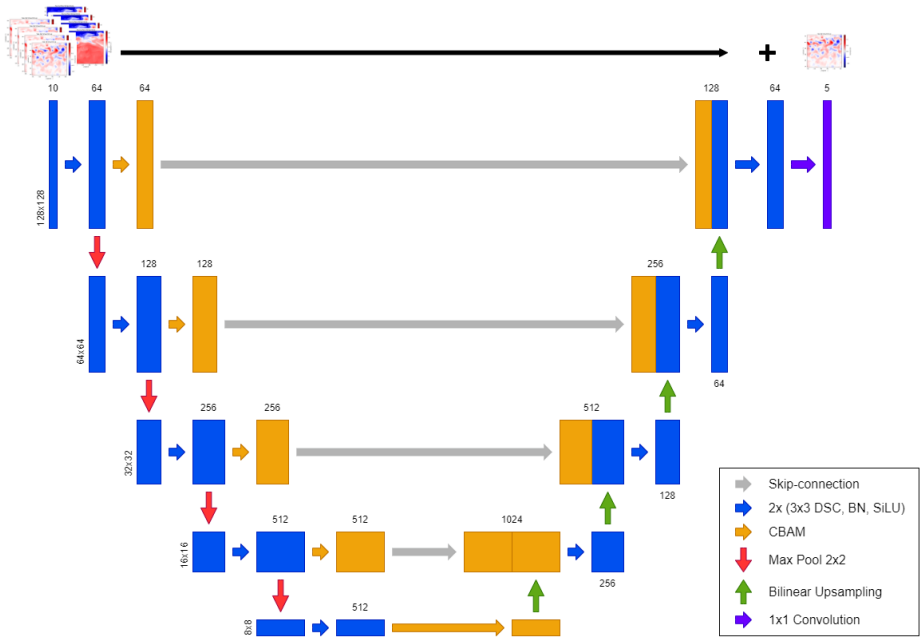

Figure 3.4: The *SLA-SST-SmaAt-UNet* architecture : Each bar represents a multi-channel feature map. The numbers above each bar display the amount of channels; the vertical numbers on the left side correspond to the x-y-size. The *SLA-SST-SmaAt-UNet* builds on the UNet architecture, incorporating depthwise separable convolutions and Convolutional Block Attention Modules (CBAMs) to enhance feature extraction, particularly for identifying specific dark regions like eddies in oceanographic images.

A significant modification in this model is our use of SiLU (Sigmoid Linear Unit) activations in the depthwise-separable convolutions instead of the traditional ReLU activations used in the original *SmaAt-UNet* model. We chose SiLU activation functions instead of ReLU for several reasons. SiLU, is defined as $\text{SiLU}(x) = x \cdot \sigma(x)$ where $\sigma(\cdot)$ is the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$. SiLU provides smoother and more continuous activation gradients compared to ReLU due to its non-linearity [32]. This non-linearity and non-zero gradient for all input values leads to better gradient flow during backpropagation, thus potentially improving model convergence. We noticed a decrease in error when testing with SiLU activations as opposed to ReLU activations, as will be discussed in chapter 4.

However, the key relevance of the original *SmaAt-UNet* model lies in the use of CBAMs [33]. These modules apply both channel and spatial attention mechanisms, which are crucial for highlighting relevant features and regions in the data. The channel attention mechanism focuses on important features across different channels by leveraging average and max-pooling operations, thereby enhancing the representation of significant features. The spatial attention mechanism, on the other hand, processes concatenated average and max-pooled feature maps through a convolutional layer to identify crucial spatial regions.

Inspired by the original CBAM paper by Woo et al. [33], we can describe the spatial and channel processes in the following way. Given an intermediate feature map $\mathbf{F} \in \mathbb{R}^{C \times H \times W}$, for channels C (sequence lengths), height H, and width W of our images, CBAM applies a sequential attention mechanism by first inferring a 1D channel attention map $\mathbf{M}_\text{c} \in \mathbb{R}^{C \times 1 \times 1}$, followed by a 2D spatial attention map $\mathbf{M}_\text{s} \in \mathbb{R}^{1 \times H \times W}$, as shown in Figure 3.5. The process can be summarized as:

$$\begin{aligned} \mathbf{F}' &= \mathbf{M}_\text{c}(\mathbf{F}) \otimes \mathbf{F}, \\ \mathbf{F}'' &= \mathbf{M}_\text{s}(\mathbf{F}') \otimes \mathbf{F}', \end{aligned} \tag{3.1}$$

where $\otimes$ denotes element-wise multiplication. In this operation, the attention values are appropriately broadcasted across dimensions, resulting in the refined output $\mathbf{F}''$.

Figure 3.5 is taken from the original CBAM paper, and depicts the computation process of each attention map. CBAMs are therefore particularly relevant for our application since our previously mentioned goal was to focus on specific darker regions, such as eddy formations which can appear as complex, low-contrast features that standard convolutions might miss. Through the application of these attention mechanisms, CBAMs allow the model to selectively emphasize these regions of interest, thus improving the detection and prediction accuracy of the crucial features.

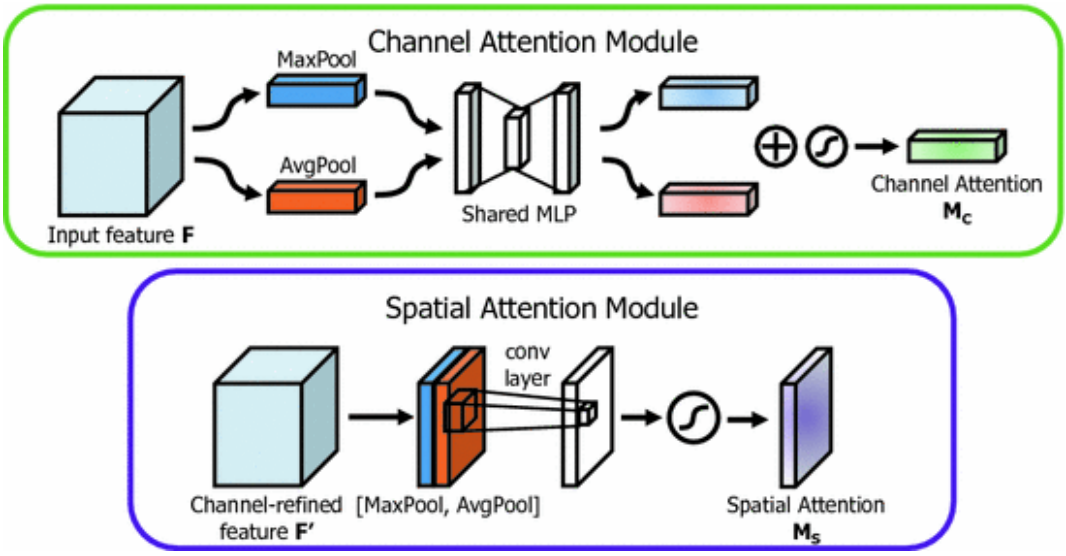

Figure 3.5: Diagram of each attention sub-module from the original CBAM paper [33]. As illustrated, the channel sub-module utilizes both max-pooling outputs and average-pooling outputs with a shared network; the spatial sub-module utilizes similar two outputs that are pooled along the channel axis and forward them to a convolution layer.

As was mentioned previously, the replacement of standard convolutions with depthwise-seperable convolutions (DSC) is a crucial feature as it significantly reduces the number of parameters of our model. Table 3.1 shows a comparison of different models' parameters. These DSCs manage to enhance efficiency of our model without excessively reducing its effectiveness. This allowed us to run more experiments as well as test considerably more hyperparameter settings in order to obtain the best possible forecast performances. Our proposed model, like the *SmaAt-UNet*, has significantly less parameters than the UNet, and due to the smaller size of our images, also has slightly less parameters than the original *SmaAt-UNet*. In our PyTorch implementation we use DSCs with one kernels-per-layer.

Model	Parameters
UNet	17,272,577
SmaAt-UNet	4,111,389
SLA-SST-SmaAt-UNet	4,033,826

Table 3.1: Number of parameters of the compared models.

Finally, another key modification was the extraction and addition of the last time step from the input images to the outputted predicted images, depicted in Figure 3.6. Since spatiotemporal data such as our SLA and SST fields have important inherent temporal dynamics, this extraction and reincorporation process ensures that the predictions are consistent with the most recent state of the inputs. By adding the last time step to the final output, we are changing the function we want to approximate. Instead of building the next time step, the model is building the modification that needs to be applied to the last time step in order to obtain the next time step. This effectively helps maintain the temporal integrity of the predicted values.

Figure 3.6: Extraction and addition scheme: the last time step of the SLA and SST sequences are extracted before concatenation of the two input tensors, and added to the output sequences of the network.

Overall, our proposed *SLA-SST-SmaAt-UNet* model retains the essential CBAM blocks to enhance the model's ability to accurately extract eddy regions across both SLA and SST sequences, but includes new SiLU activations for regularization of excessively high weights and smoother gradient flow, as well as incorporates the last time steps of the sequences to ensure temporal and spatial consistency. Additionally, the model's smaller size allows for an increased number of experiments as will be seen in the next chapter.

Chapter 4

Experiments

The focus of our study was on the forecasting of the SLA fields only. The other objective of our work was to find the number of time steps for which the forecast is still accurate; the prediction horizon. This is evaluated by certain performance metrics such as the root-mean squared error (RMSE) which will be elaborated on further in our results. A description of the hyperparameters used for our tests can be found in Appendix A. Our pre-processing, transformation, and dataloading process allows us to have input tensors of shape `[batch_size, channels, longitude, latitude]` for the SLA and SST images. However, before feeding both SLA and SST inputs to the network, we began with strictly SLA trials to situate the performance of the network. This allowed us to establish a baseline performance for our first model, and set up comparisons with the pre-existing model developed by Ollier et al. in 2022.

We trained, validated, and tested our model on the data sets outlined in section 3.1.2. We know that the time evolution of the SLA is driven by ocean eddies whose return time is approximately 20 days in the Gulf Stream [34]. Therefore, to serve as an initial trial, we fed to the network a sequence of 20 SLA images, denoted as `[t-19, ..., t]`, in order to produce a forecasted output of the next 20 SLA images, denoted as `[t+1, ... t+20]`. Our input tensor was then of size `[16, 20, 128, 128]`. This first model was denominated the *SLA-SmaAt-UNet* since it only processed SLA data before the introduction of SST fields. Since initial results at `t+20` were promising but not as accurate as desired, we decided to test multiple prediction horizons. Accordingly, our prediction horizons would we separated by five days, with results sampled at four different time steps, corresponding to times `t+5, t+10, t+15, t+20`. This provides a better understanding of our model's performance across both short-range and long-range forecasting tasks. The performance of our model was evaluated via the results of our experiments when forecasting at all four time steps individually, as well as when forecasting for strictly `t+20`.

4.1 Leveraging SST Data

We introduced the SST data in order to refine the model's SLA predictions since it acts as additional information for the model to learn. Our *SLA-SST-SmaAt-UNet* model was modified to accept two inputs, one for SLA images and one for SST images, which are concatenated along the channel dimension. When forecasting for a specific prediction horizon, the input length must match the desired length of the output tensors along the channel dimension. With the introduction of the SST images, the loss is calculated for SLA and SST fields separately and then added together in order to return one loss value. Therefore, a separate training must be conducted when predicting at different time steps, since each model is trained on sequences of length equal to a time step in `[t+5, t+10, t+15, t+20]`. A more detailed description of our loss function can be found below in section 4.2.

Something significant we noticed during our tests with only SLA fields compared to our tests with both SLA and SST fields, was the convergence of the training. Testing the *SLA-SmaAt-UNet* with strictly SLA fields caused the model's training to converge almost immediately to a relatively low validation loss. While this still produced good results as will be examined later, the model was training for only a couple epochs. This same behavior was observed when testing for all time-steps `t+5, t+10, t+15, t+20`, implying that the model was potentially lacking information to learn. However, when introducing the SST fields as additional information for the model to learn, we observed a significantly more encouraging convergence with a proper loss reduction. Even when employing the same training system, the *SLA-SST-SmaAt-UNet* trained over more epochs as it learned the dependencies from both the SLA and SST fields. As will be seen in chapter 5, both sets of tests still produced encouraging metric values and results.

4.2 Weighted Loss Function

A key aspect of our experimental work was the introduction of a weighted mean-squared error (MSE) loss function to improve model performance over longer forecast horizons. During our early testing phase, we observed a decline in prediction quality as the forecast extended further, which is a common challenge. To address this, we applied a weighted loss function that assigned greater importance to the loss on later time steps, encouraging the model to focus on improving the predictions of these later images. Initially, we experimented with both exponential and quadratic weight increases. We computed the MSE loss for each time step, applied a corresponding weight, quadratic or exponential, and averaged these weighted losses to produce the final loss value. However, we found that while exponential weights with higher base values intensified the focus on later images, they did not significantly enhance RMSE results. Conversely, a quadratic increase in weights for the sequences at `t+5`, `t+10`, and `t+15` yielded better forecast performance, leading us to adopt this approach in our training process.

4.3 Evaluation Metrics

We consider the following metric to evaluate SLA predictions: the root-mean-square error (RMSE) computed on the test set for each image in the set (see equation 4.1).

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(y_{target} - \hat{y}_{pred})^2} \tag{4.1}$$

Where N is the total number of test pixels in an image, and n is the index over the test pixels used to compute the RMSE. This metric allows for an average error of our prediction accuracy while maintaining the original base measurement units of our data. Since RMSE is an L2 penalty which penalizes the sum of squares of weights, it is a good fit for our evaluation due to both our weighted loss function and for minimizing the error for longer-range forecasting. We will also be considering a L1 regularisation penalty, the mean absolute error (MAE) (see equation 4.2).

$$MAE = \frac{1}{N}\sum_{i=1}^{n}|y_{target} - \hat{y}_{pred}| \tag{4.2}$$

Where N and n are the same as in the equation for RMSE. This ensures a robust and comprehensive comparison of the performances of our two models. Additionally, we use persistence performances as a threshold to assess our model's performances. Persistence refers to a prediction that assumes either zero uniform velocities or velocities that cancel each other out.

Chapter 5

Results and Discussion

In this chapter we will examine the various performances of the models based on the metric statistics that were calculated in our experiments. Following training of our two discussed models, the *SLA-SmaAt-UNet* and the *SLA-SST-SmaAt-UNet*, we selected for both models the ones with the lowest validation loss from their training run. These best performing models were then used to calculate our metric on the test set. When examining our results from the testing set, the model's predictions along with the corresponding ground truth are extracted, inverse-scaled to reverse the standard normalization performed in the data pre-processing step, and plotted alongside each other to visualize the differences.

As can be seen in Table 5.1, our *SLA-SST-SmaAt-UNet* model outperforms the other models at the short-range and long-range time steps, as well as the persistence baseline at time step `t+5`, when measuring with an RMSE metric. The other models being compared are the *SLA-RES-UNet*, the pre-existing model-to-beat developed by Ollier et al. [23], and the *SLA-SmaAt-UNet*, our first developed model without the additional SST data. With a significantly lower RMSE value at time `t+5` for both our models compared to the persistence baseline and model-to-beat, we can confidently say that they perform well at short-range SLA forecasting tasks. The *SLA-SST-SmaAt-UNet* performs better than the *SLA-SmaAt-UNet* at times `t+5` and `t+20`, but has a slightly higher error at the two medium-range time steps.

Model	Metric	t + 5	t + 10	t + 15	t + 20
SLA-RES-UNet	RMSE (cm)	13.13	18.53	22.07	24.59
SLA-SmaAt-UNet	RMSE (cm)	9.31	**15.63**	**18.94**	21.67
SLA-SST-SmaAt-UNet	RMSE (cm)	**8.93**	15.97	19.11	**20.87**
Persistence	RMSE (cm)	13.07			

Table 5.1: Comparison of architecture performances with RMSE (cm) when forecasting at each time step individually on the dynamic region of the Gulf Stream (Latitude 33.7° to 44.42° N). The *SLA-RES-Unet* architecture is the pre-existing model by Ollier et al. (2023).

Something important to note in these results is that the RMSE values at each time step in the table are produced from a training run where the model is trained on only image sequences of length corresponding to that time step. This is as opposed to training our model on sequences of 20 images, testing it on a sequence of 20 images, and pulling the individual test metrics at time-steps `t+5, t+10, t+15,` and `t+20`. All the corresponding MAE metrics, along with the table with the RMSE values produced at each time step when trained strictly on sequences of 20 images for the *SLA-SmaAt-UNet* and *SLA-SST-SmaAt-UNet* models can be found in Appendix B.

To gain a better idea of the trend of the increase in RMSE over the sequence of 20 images, we can look to Plot 5.1. Since the RMSE values presented here are the **average** RMSE value for each time step calculated over all the batches present in our test set, we are also able to calculate the standard deviation of each RMSE value, which are the error bars in the plot. These standard deviations offer a better understanding of the reliability of the *SLA-SST-SmaAt-UNet* model's prediction at each time step. As expected, as the prediction horizon increases, so does the standard deviation of the mean RMSE of each forecast. We can observe an interesting trend in the increase of RMSE values for the sequence of `[t+1, ..., t+20]` SLA images. While the RMSE increase is of a linear fashion at first, it eventually begins to increase at a slower rate before `t+10` and begins to resemble a quadratic growth as the image sequence progresses. This makes sense due to the implementation of our weighted loss function which included a quadratic growth for the weights over the sequence of predicted images. The emphasis on later images due to the quadratic weights forces the model to minimize the error more aggressively for later images, resulting in a less noticeable increase in RMSE as we move to the later parts of the sequence. The standard deviation also increases for each image over time, which is expected. An analysis of the boxplots of the different time steps will be presented later on as well. Aside from the very first time step, which is expected, our model performs better than the persistence baseline at every time step, thus corroborating our results from Table 5.1.

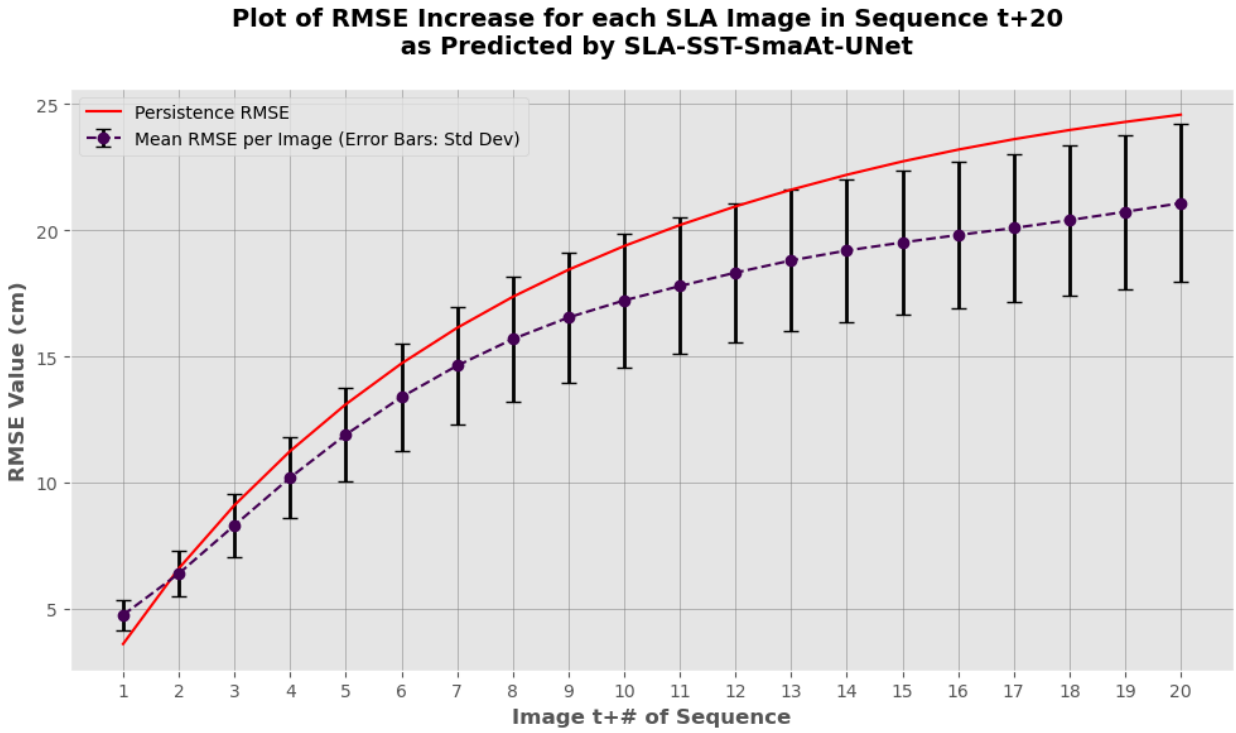

Figure 5.1: Plot of the increase in RMSE (cm) with the associated standard deviation error bars for each SLA image when forecasting a sequence of length `t+20`, as predicted by the *SLA-SST-SmaAt-UNet* model on the dynamic region of the Gulf Stream (Latitude 33.7° to 44.42° N).

A comprehensive comparison in the performances of the *SLA-SmaAt-UNet* and the *SLA-SST-SmaAt-UNet* with boxplots in terms of RMSE at the different forecast ranges is depicted in the Figure 5.2. While the mean RMSE at each time step is a good metric to measure average performance at each prediction horizon, the boxplots allow for a more comprehensive evaluation of the consistency and reliability of each model's

forecasts when they are trained. As was evident from the metric results in Table 5.1, both the models largely outperform the persistence baseline at time step `t+5`. It is apparent from these boxplots that the *SLA-SST-SmaAt-UNet* informed with additional SST fields performs better at short-range tasks, but falls just short of the purely SLA-informed model at medium-range tasks. The most striking point, however, is the comparison in interquartile ranges for both the short-range forecast and the long-range forecast. The *SLA-SST-SmaAt-UNet* has a shorter boxplot at time-steps `t+5` and `t+20`, suggesting its predictions are more reliable than the model trained without the SST fields for these prediction horizons. This supports our previous speculation that forcing the network to learn complimentary SST fields allows for a somewhat improved representation of the underlying dynamics of SLA.

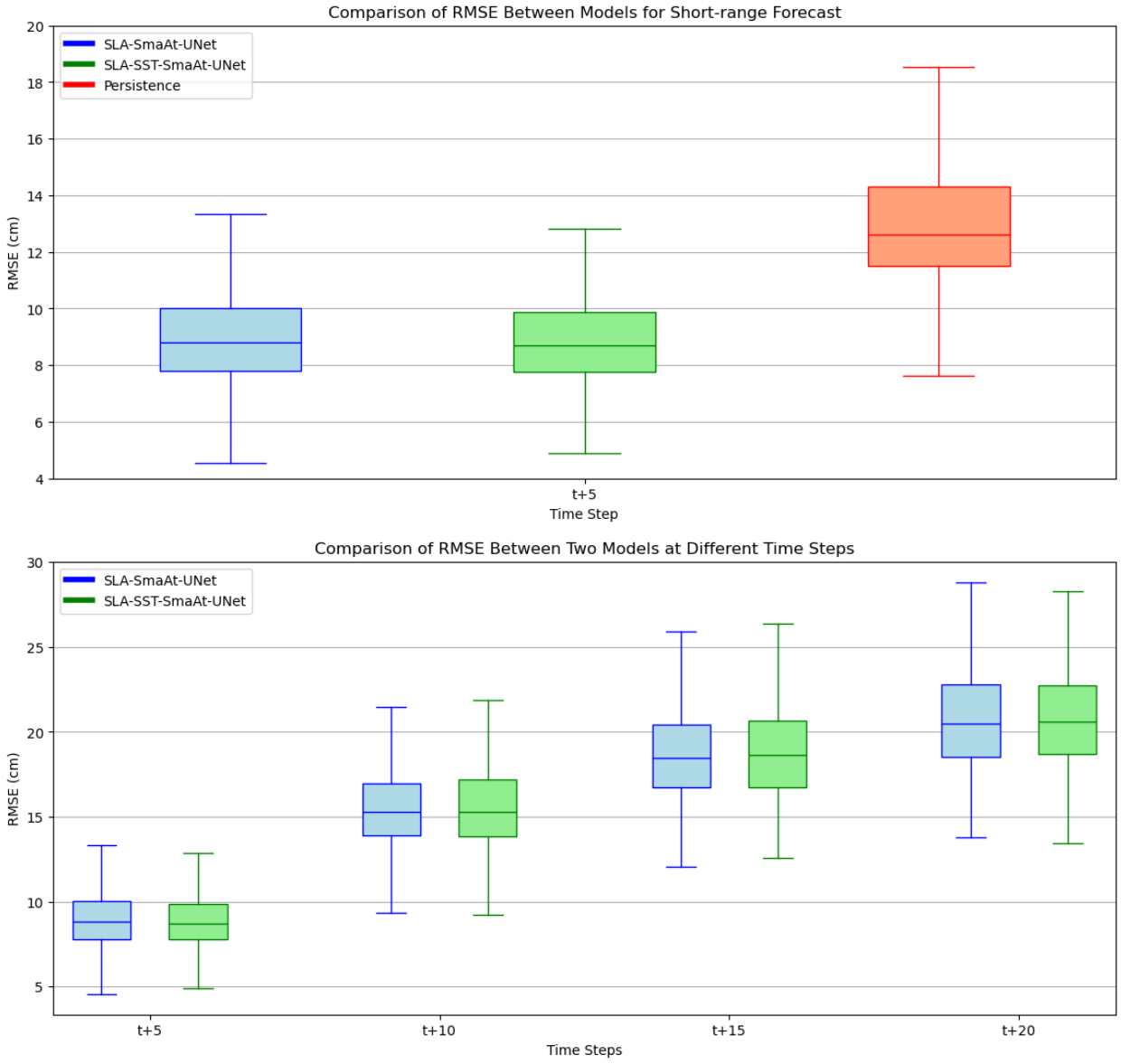

Figure 5.2: Boxplots of the mean RMSE (cm) with standard deviations estimated on the whole test set on the dynamic region of the Gulf Stream (Latitude 33.7° to 44.42° N) at time step `t+5` *(top)*, and for the four different time steps *(bottom)* forecasted individually.

In Figure 5.3, an example of the evolution of the SLA field predictions of our *SLA-SST-SmaAt-UNet* model up to time step `t+5` can be seen compared to the corresponding ground truth images. While the first image at time step `t+1` seems indistinguishable from the target image, as we move along the sequence of images, some noticeable differences begin to form in our predictions compared to the ground truths. The sequences of differences between the two images visualized in the bottom row tells the same story, with an apparent increase in geostrophic contrast. However, we can infer from these results that the model conserves the darker features corresponding to eddies over the progression of the sequence. It is clear from these results that the *SLA-SST-SmaAt-UNet* is very good at forecasting short-range images as can be seen from the results at `t+1` through `t+3`. The differences between the predicted and target images become more prominent from `t+3` onwards, indicating a slight loss in the model's predictive accuracy as the temporal gap increases.

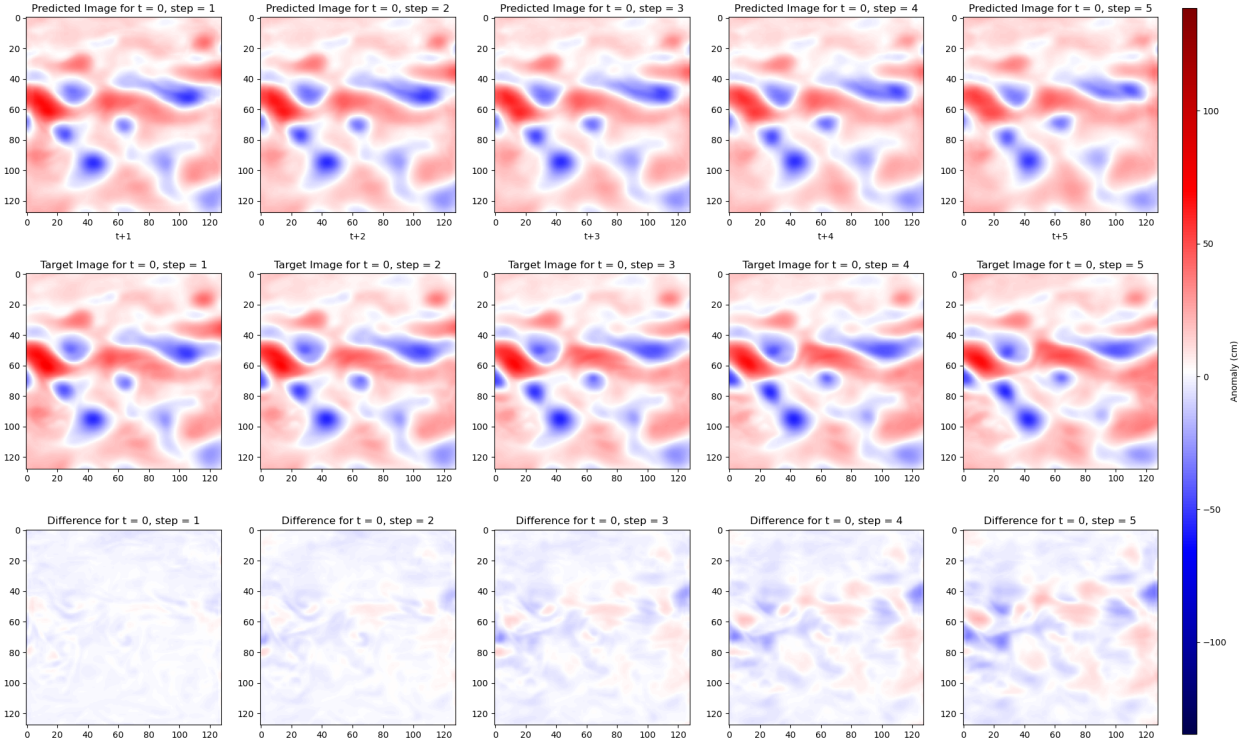

Figure 5.3: Predictions by *SLA-SST-SmaAt-UNet* of SLA *(top row)*, the Ground Truth Images *(middle row)*, and the difference between the two *(bottom row)* for times `[t+1, ..., t+5]` when forecasting for a sequence of 5 images on the dynamic region of the Gulf Stream (Latitude 33.7° to 44.42° N).

In order to visualize the results of the set time steps when trained for sequences of 20 images, we can look to Figure 5.4. We have plotted the forecasted results at time steps `t+5, t+10, t+15, t+20` for both the *SLA-SmaAt-UNet* and the *SLA-SST-SmaAt-UNet* as well as the ground truth images for comparison. We can notice a clear breakdown in the geostrophic current dynamics of the SLA fields over time which demonstrates the lack of accuracy of the models' for long-range prediction tasks. Since the *SmaAt-UNet* was designed for nowcasting tasks in the original article [20], these results are expected. In the case of our two models, the short-term forecast results maintain the major eddy formations present in the ground truth images, but these dissipate in long-range predictions. However, the `t+20` SLA image forecasted by the *SLA-SST-SmaAt-UNet* preserves a darker formation in a location similar to the one in the target image.

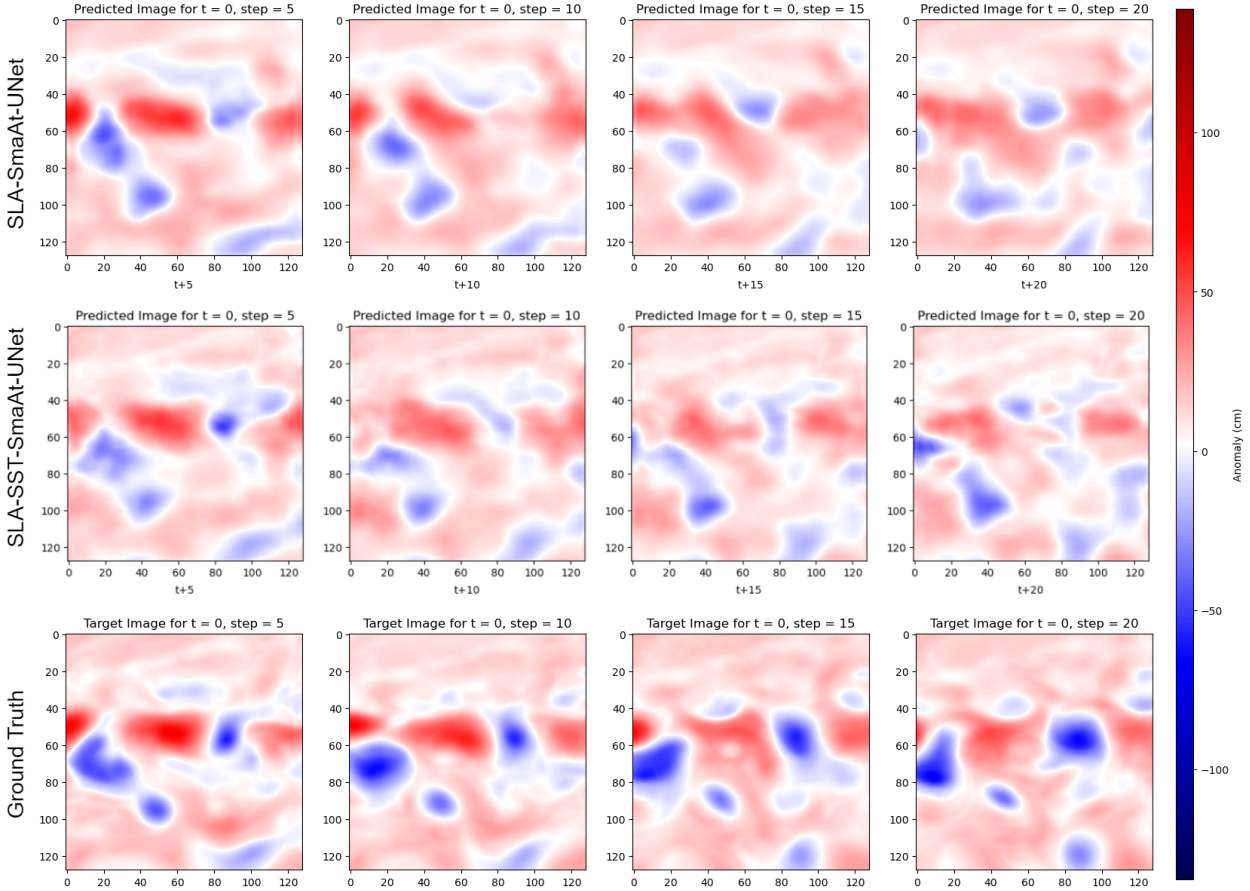

Figure 5.4: Predictions by *SLA-SmaAt-UNet* of SLA *(top row)*, predictions by *SLA-SST-SmaAt-UNet* of SLA *(middle row)*, and the Ground Truth Images *(bottom row)* for times `[t+5, t+10, t+15, t+20]` when forecasting strictly for a sequence of 20 images on the dynamic region of the Gulf Stream (Latitude 33.7° to 44.42° N).

As presented by the results above, we have demonstrated the differing efficiencies of our models. Our proposed *SLA-SST-SmaAt-UNet* model that learns additional SST data on top of the SLA fields, does in fact improve prediction accuracy for short-range forecasts (`t+5`) compared to both the model-to-beat as well as the persistence baseline. While the additional SST information seems to muddle the medium-range forecasts since our first *SLA-SmaAt-UNet* model learning strictly SLA data performed better at these time steps, we were able to observe that the complimentary SST fields allows the network's predictions to be more reliable for long-range forecasts (`t+20`). It can be observed, however, that these improvements do not represent major differences, which implies that while SST data helped our model learn additional features, a large success in both our attention-based models can be attributed to the spatial and channel attention modules incorporated by the CBAMs. This allows these models to maintain relevant features in the data. The number and position of troughs and bumps that characterize ocean circulation via the geostrophic relationship are similar in both our models' predictions and the original ocean images. This highlights the improvement made upon the established model-to-beat by Ollier et al. [23] via the incorporation of Attention-modules. This research project is a testament to their usefulness in forecasting tasks, and corroborates the claim that they enable improved short-range forecasting performance on simulated satellite data.

Chapter 6

Physics-informed Neural Networks

In this chapter, we will give some background on Physics-informed Neural Networks (PINNs) models and their relevance in our investigations, as well as the research we conducted. This is exploratory research that is meant as an extension of the official internship subject in order to advance our investigations and findings since it was not originally part of the thesis. We will provide explanations about these conclusions as well.

6.1 Introduction

As a step further in our research into forecasting and modeling ocean systems, we wanted to explore a subject that would make more sense physically. The time series of SLA and SST images are governed by time-dependent partial differential equations (PDEs). Therefore, we wanted to expand our already stable integral representation approach found in our deep learning *SLA-SST-SmaAt-UNet* model by inducing physical laws into the process. PINNs allow for this, and the physics that is added when solving the problem act as a regularizer for the neural network by forcing it to adhere to known physical laws. A key aspect of learning these physical dynamics with neural networks is to consider that dynamical systems obey an Ordinary Differential Equation (ODE) or a Partial Differential Equations (PDE). In these cases, we are assuming that the PDE describing the physical system is known as well as the initial and border conditions. Since PINNs are neural networks that are trained to produce a specific solution of a nonlinear PDE given data [15], it is necessary to define the PDE and boundary conditions that describe the particular problem at hand. To serve as initial trials, we selected a problem context involving the fluid flow of a temperature field through a canal described by a partial differential equation. Along with velocity equations and a PDE for temperature advection, we were able to set a basis for our research. In the following sections, we will draw out our research objectives and the problem set-up. This will allow us to transition to the work and experiments conducted, followed by investigations we pursued using other well-known dynamical fluid PDEs.

6.2 Objectives

Our objectives could be broken down in the following steps: (1) Solve the temperature advection PDE using a numerical method such as a simple explicit finite difference update; (2) Create and train a PINN on this PDE, allowing it to randomly sample data within the boundary conditions and recreate the flow of the temperature field as it learns the PDE; and (3) Train our previous model on the images generated by the classic solver from step 1 in order to compare them to the results of our PINN. An extended additional step, to incorporate image data of simple SST fields into our PINN to improve training, was also considered, but only if time allowed. Most importantly however, was to dive into the field of physics-informed machine learning and further investigate the applicability of PINNs to the world of climate modeling.

6.3 Temperature Advection Problem

For the modeling of a dynamic fluid flow of a temperature field described by a partial differential equation for temperature advection, the problem can be set up in the following way. We consider a fluid flow in a canal of width L_1 and length L_2, where the velocity is given by the following equations:

$$U = U_0 \sin\left(\frac{\pi y}{L_1}\right) \tag{6.1}$$

$$V = V_0 \sin\left(\frac{\pi y}{L_1}\right) \tag{6.2}$$

where U is the velocity in the East-West direction and V is the velocity in the North-South direction. We also define the following parameters for our context, where $U_0 = 0.5$ m/s, $V_0 = 0.05$ m/s, $L_1 = 2 \cdot (10^5)$ m, and $L_2 = 2 \cdot (10^5)$ m.

The temperature T of the fluid flow for $x = 0$ as an initial boundary condition is given by:

$$T = T_0 \sin\left(\frac{2\pi y}{L_1}\right) \sin\left(\frac{2\pi t}{P}\right) \tag{6.3}$$

where $T_0 = 1$ is the initial temperature and $P = 40$ days is the time period of the simulation.

The temperature is advected by the following partial different equation:

$$\frac{\partial T}{\partial t} + U\frac{\partial T}{\partial x} + V\frac{\partial T}{\partial y} = 0 \tag{6.4}$$

In order to solve this, we must respect the Courant–Friedrichs–Lewy (CFL) condition which is a necessary condition for convergence while solving certain partial differential equations numerically [35],[36]. This ensures that the numerical domain of dependence contains the physical domain of dependence, so the numerical velocity must be greater than the physical velocity. In our case, we must have $\Delta t < 5 \cdot (10^3)$ s if we are using a spatial grid of 5 km for our solver, since this grid spacing will be necessary to create the space and time arrays X, Y, and $time$.

6.3.1 Classic Solver

For our numerical solving method which would serve as our baseline when comparing to a PINN model, we decided to use a simple explicit finite difference method. For this, we simply need to solve the temperature advection PDE using this discrete update scheme. The temperature T is updated at the next time step $n + 1$ based on the current values a time n:

$$T_{i,j}^{n+1} = T_{i,j}^{n} - \Delta t \left(U(Y_j)\frac{T_{i,j}^{n} - T_{i-1,j}^{n}}{\Delta x} + V(Y_j)\frac{T_{i,j}^{n} - T_{i,j-1}^{n}}{\Delta y} \right) \tag{6.5}$$

This is a straightforward explicit update scheme where the new value of T is directly computed using the current values and the given discretization parameters; spatial steps Δt, Δy and time-step Δt. Implemented in Python, this finite difference algorithm effectively solves the matrix T for all temperatures over time based on the initial temperature conditions and spatial boundary conditions provided in the problem set-up. In Figure 6.1, we can see two plotted examples of the temperature field of the fluid flow in the canal at two different time steps. This is the ideal result for our PINN model which will be described next.

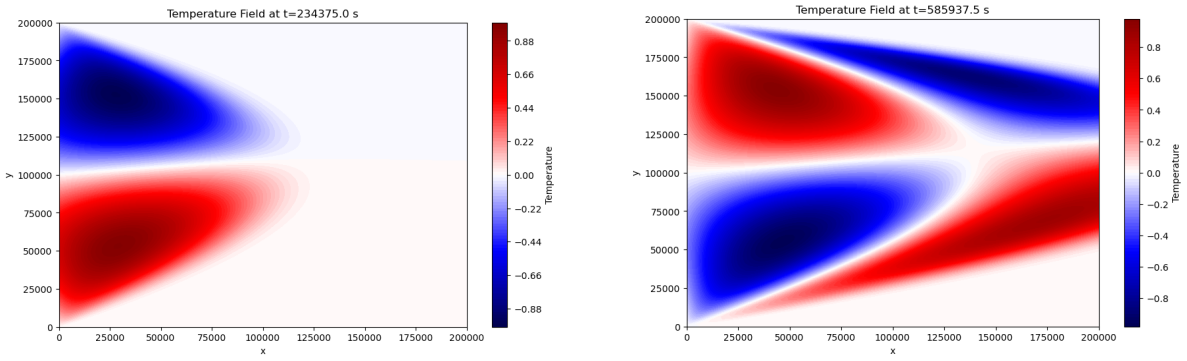

Figure 6.1: Examples of the plot filled contour of the temperature fields of the dynamic fluid flow in the canal at two different time steps (in seconds), one earlier *(left)* and one later *(right)*, with the associated temperatures.

6.3.2 The PINN Model

Physics-informed Loss Physical machine learning works by injecting physics-based learning somewhere into the process. Some projects have this directly within the model architecture, but we instead inject the physics of the fluid dynamics in a custom loss function. Our loss function is therefore made up of a traditional data-informed loss with the addition of a physics-based loss. The data-informed loss is computed using the initial and boundary conditions, while the physics-informed loss is calculated by randomly sampling from the domain space and constraining the data using the temperature advection PDE. This loss function can be expressed in the following way:

$$\mathcal{L} = \mathcal{L}_{data} + \lambda \cdot \mathcal{L}_{physics} \tag{6.6}$$

where λ is a weighting factor to control the weight of the physics-based loss. The data-informed loss is defined by computing the mean squared error of the estimated temperature $\hat{T}_0$ using the initial temperature boundary where $x = 0$ everywhere, as predicted by our neural network model. We will describe this model later on, but it is a feed-forward artificial neural network that we will denote as MLP for now. Therefore, the data loss is computed in the following way:

$$\hat{T}_0 = MLP(x = 0, y, t) \tag{6.7}$$

$$\mathcal{L}_{data} = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{T}_0 - T_{true} \right)^2 \tag{6.8}$$

where T_{true} is the true temperature computed using Equation 6.3 with the initial boundary of $x = 0$. For the physics-informed loss, we solve the advection PDE provided in Equation 6.4. For this, we take the derivative of $\hat{T}$ predicted by our MLP model with respect to x, y, and t using the `torch.autograd.grad` function. We then multiply them by the horizontal and vertical velocities from Equations 6.1 and 6.2. To calculate the mean squared error of this solution, we use a matrix of zeros the same size as the predicted temperature matrix. This can be expressed in the following way:

$$\mathcal{L}_{physics} = U \cdot \frac{\partial MLP(x, y, t)}{\partial x} + V \cdot \frac{\partial MLP(x, y, t)}{\partial y} + \frac{\partial MLP(x, y, t)}{\partial t} \tag{6.9}$$

The addition of these two loss terms make up the final loss for the training of our model, which will be explained next, after which we will elaborate on our data sampling process.

The Model For our neural network model, we began testing with a simple multilayer perceptron (MLP) due to its ability to learn non-linear patterns like the ones present in our dynamic fluid flow system. We used multiple hidden layers, batch normalizations [37], as well as SiLU activations [32] due to their non-linear nature. These were relatively straightforward decisions since by employing deep feed-forward neural networks we can leverage their well known capability as universal function approximators [38]. This then allows us to utilize automatic differentiation [39] (the `torch.autograd.grad` function mentioned in the previous paragraph) to differentiate our MLP with respect to the input coordinates x, y, and t described in our problem setup, and thus obtain a physics-informed neural network.

Data Sampling For our data generation, since we did not have access to actual data, we had to generate our own two sets of data; one for the physics-informed loss and one for the data-informed loss. For our physics-informed loss, we generated data samples for the numerical solution of the PDE. For this, the spatial domain was discretized into a uniform grid along both dimensions x and y, which ensured an equal number of points along each axis. This number of points was predetermined as 128 for our grid spacing. Similarly, the temporal domain was discretized with a consistent time step, Δt from the problem setup in Section 6.3, to cover the entire simulation duration of 40 days. This discretization provided us with a structured set of points across both space and time and formed the basis for our sampling. We then performed random sampling over these spatial and temporal grids by randomly choosing indices corresponding to our pre-defined grid points, and returning our x, y, and t data as tensors. The number of points sampled corresponded to a batch size which is set during our training and is used for sampling our boundary condition data as well.

For our boundary condition data, we systematically generated data points using all possible combinations of the discretized spatial and temporal points along the boundary of the domain. This boundary corresponds to our discretized spatial and temporal grid. These were once again sampled by batches to make our training more manageable when using a high number of points, similarly to our PDE data previously, and returned as tensors for our training.

6.3.3 Experiments and Results

Training When training our PINN, we generate and sample both the PDE data and the boundary condition data at each epoch in order to provide our feed-forward neural network with enough data to both learn the boundary conditions and approximate the partial differential function. We then calculate both losses, add them to produce our overall loss, and log each loss to print them at each epoch. For the PDE weight λ from Equation 6.6, we began with a value of 10.0 since we noticed that the PINN was much better at approximating the PDE than learning boundary conditions. As for our hyperparameters, we tested various values for our MLP parameters and for training parameters, and settled on 8 layers with a hidden dimension of 256, while training for 200 epochs with a batch size of 16384, corresponding to 16384 points on each grid. We used a traditional `Adam` optimizer with a learning rate of 0.01, as well as a mean squared error (MSE) loss as previously mentioned.

Observations As we ran our experiments, the first thing we noticed is that the physics-informed loss decreased extremely quickly and achieved a value close to zero almost immediately after the first epoch, while the boundary condition loss decreased gradually as expected. Additionally, after running multiple experiments, we also noticed that the training was very unstable, with drastically different loss values and loss reduction patterns at each training run, sometimes diverging completely. We experimented with adding a large weight factor to the model's PDE solution before calculating the MSE loss, but noticed very little improvement in both the training progression and the plotted results.

Results In order to maintain consistency with the plotted solution provided by the finite difference method from Section 6.3.1, we plotted our model's testing results in a similar manner. For this, we generated a spatial and temporal grid of points similarly to the one done in our data sampling step, and evaluated our model on this test set of points. As expected, the contour plot results were extremely different at each training run showcasing the instability of the model. An example of two different contour plot results can be seen in Figure 6.2.

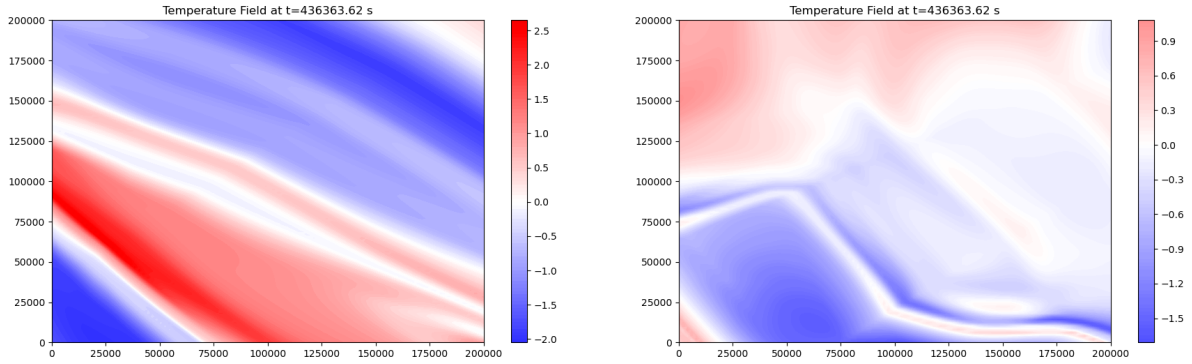

Figure 6.2: Two examples of contour plots of the temperature field of the dynamic fluid flow in the canal as predicted by the PINN model, with the associated temperatures.

While the contour plot on the left seems more promising than the one on the right, they are both still quite different from the expected results produced by the finite difference solver. We ran numerous experiments due the quickness of the training, and we found that results like the one on the left of Figure 6.2 occurred much less frequently than the ones like the plot on the right.

6.4 Additional Tests

In order to further our investigation into PINNs, we decided to test the modeling of well-known and established dynamic fluid equations. For this we selected the Navier-Stokes Equations and Burger's Equation. For this, we used the code provided by Raissi et al. in their original PINNs paper [15]. While their work was implemented in Tensorflow, we were able to find an approximate Pytorch translation in order to run our own experiments. This allows us to also utilize the original data used to produce the results in the paper.

6.4.1 Navier-Stokes Equations

The Navier-Stokes equations involve a realistic scenario of incompressible fluid flow, and describe the physics of many phenomena of scientific and engineering interest. They may be used to model the weather, ocean currents, and water flow in a pipe. Due to their very complex 3-dimensional nature in their full form, we decided to consider the Navier-Stokes equations in two dimensions (2D), like in the original article. The 2D Navier-Stokes equations, which govern the velocity field and pressure of incompressible fluids, can be described in the following way. Assuming a latent stream function, $\psi(t, x, y)$, that automatically satisfies the continuity equation $u_x + v_y = 0$, we aim to learn the unknown parameters $\lambda = (\lambda_1, \lambda_2)$ and the pressure $p(t, x, y)$ from noisy velocity field measurements. To achieve this, we define the residuals

$$\begin{aligned} f &:= u_t + \lambda_1 (u u_x + v u_y) + p_x - \lambda_2 (u_{xx} + u_{yy}) , \\ g &:= v_t + \lambda_1 (u v_x + v v_y) + p_y - \lambda_2 (v_{xx} + v_{yy}) , \end{aligned} \tag{6.10}$$

and proceed by jointly approximating $\psi(t, x, y)$ and $p(t, x, y)$ using a PINN that minimizes a mean squared error loss based on f and g. The full mathematical formulation of the Navier-Stokes equations along with the associated MSE Loss equation for the PINN model is provided in Appendix C.

Using the Navier-Stokes PINN model and training code as well as the data, we were able to launch a training of the network. Our first observation, however, was that this training was extremely slow based on the amount of data available and considering our own computational resources. This limited our experimental freedom, but still allowed us to obtain results. In Figure 6.3, is an example of a velocity field as predicted by the Navier-Stokes PINN model compared to the ground truth velocity field obtained from the data.

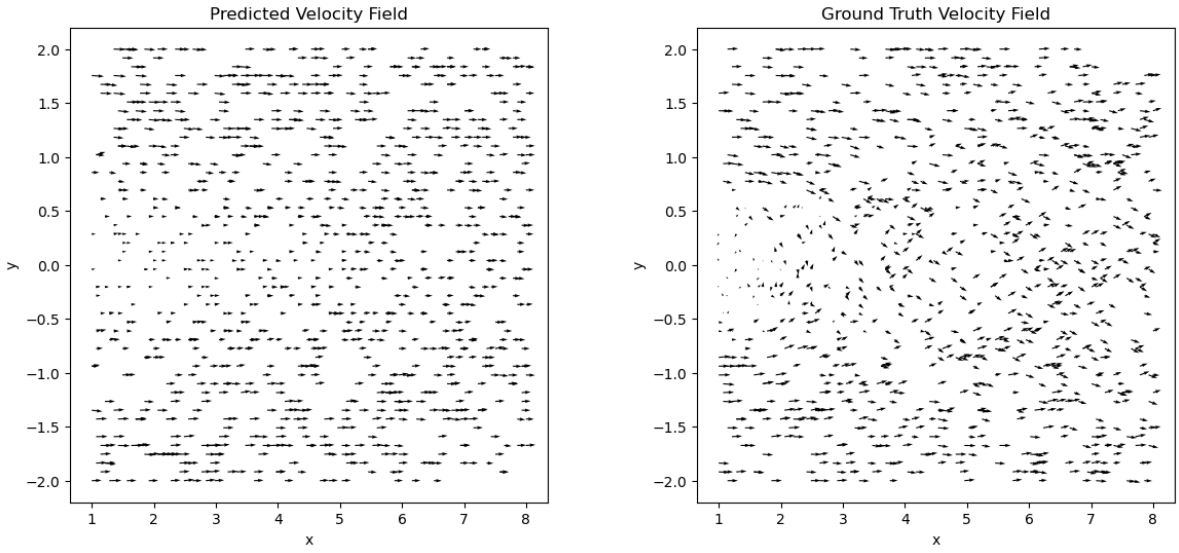

Figure 6.3: Predicted velocity field of the Navier-Stokes PINN model *(left)* compared to the ground truth Navier-Stokes velocity field *(right)*.

As can be seen in Figure 6.3, the predicted velocity field, while bearing some similarities, is still very distinct from the ground truth velocity field. During training, we observed a proper loss decrease reaching a relatively high loss value even when increasing the number of training epochs. Based on these results, it appears that the model was either not trained enough and therefore did not properly learn the physics of the Navier-Stokes PDE, or suffered from the common instability issue found in physics-informed machine learning tasks. Our biggest takeaway from this particular experiment, and one that was found to be extremely common across our investigations, was the difficulty of replication results of studies surrounding PINNs, largely due to this instability of their training.

6.4.2 Burger's Equation

In an attempt to model a simpler PDE in only 1 dimension which should yield more stable results, we decided to investigate the Burger's equation. The Burger's Equation is a fundamental partial differential equation and convection-diffusion equation present in many areas of applied mathematics, such as our case of fluid dynamics. It can be derived from the Navier-Stokes equations for the velocity field of fluid motion through its expressions for nonlinear advection and diffusion, however it is only one-dimensional and omits a pressure gradient driving the flow. In one-dimension space, Burger's equation with Dirichlet boundary conditions, can be solved by defining the residual function

$$f := u_t + uu_x - \left(\frac{0.01}{\pi}\right) u_{xx}, \tag{6.11}$$

and approximating $u(t, x)$ using a deep neural network. The model's parameters are optimized by minimizing a mean squared error (MSE) loss, which includes contributions from both the initial and boundary conditions, as well as the residual f:

$$MSE = MSE_u + MSE_f. \tag{6.12}$$

Here, MSE_u captures the error on the initial and boundary data, while MSE_f enforces the structure of Burger's equation at selected collocation points. Once again, the full mathematical formulation of Burger's equation along with the full details of the boundary conditions, initial conditions, and the exact formulations of MSE_u and MSE_f, are provided in Appendix C.

To approximate Burger's equation with a PINN, we modified the code used for the Navier-Stokes equations and implemented the new partial differential equation. However, the data used for the Navier-Stokes PINN did not apply to the Burger's equation PINN since it was specially prepared for a Navier-Stokes approximation. We therefore had to generate our own data similarly to how we generated the boundary condition data for our previous temperature field advection PDE problem from section 6.3.2. Using a similar training class adapted to the Burger's equation and its variables, we were able to successfully train a deep neural network informed with the physics of the Burger's PDE in order to predict the corresponding velocity field of the Burger's equation. Unfortunately, in order to generate enough data to properly train our PINN, we must generate a very large amount of data points which in turn significantly increased the training time. This became a major limitation on our investigations due to some training, validation, and testing cycles taking upwards of 6 hours on an NVIDIA RTX A5000 graphics card. We were able to obtain some velocity field prediction results, which can be seen compared to the target velocity field in Figure 6.4. These results represent the evolution of the velocity field u over time t and space x. To evaluate our results compared to a ground truth, we plotted the analytical velocity field solution using a mesh grid similarly to the results of the temperature advection PDE.

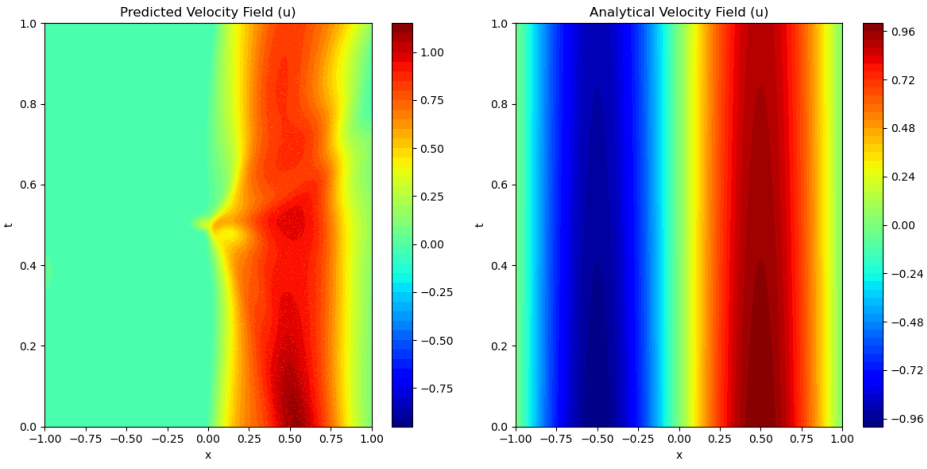

Figure 6.4: Predicted evolution of the velocity field over time and space of the Burger's PINN model *(left)* compared to the simulated analytical Burger's velocity field *(right)*.

As we can see from the comparison in velocity fields from Figure 6.4, the neural network struggles to learn the physics of the Burger's PDE. During training, we could observe a proper loss decrease, but this did not seem to be reflected in the predicted output. The PINN clearly learns the evolution of the positive magnitude of the velocity field u over time and space as can be seen in red, but completely lacks the negative magnitude of the velocity field as can be seen by the absence of the blue velocity points. Even though we were limited in the number of trainings we could run, we still noticed the same results across all experimental variations with the Burger's PINN. These results communicate a potential weakness in the neural network itself.

6.5 Discussion

Through our research and investigation into physics-informed machine learning, we managed to discern some key aspects of PINNs and their training process. For example, one of the primary advantages of using neural networks is their inherent ease of differentiation, thanks to the computational graphs that underlie these models. This was demonstrated in our of the `torch.autograd.grad` function due to the `create_graph=True` option. This property theoretically makes them well-suited for approximating the solutions of PDEs, as neural networks can approximate any continuous function [38], including multi-linear functions that are often solutions to PDEs. However, the practical implementation of these theoretical capabilities presents significant challenges. As we observed ourselves, the convergence of neural networks when applied to PDEs in practice is especially unstable and difficult to achieve consistently. Our experiments for the temperature advection PDE, which utilized a simple MLP, underscored these difficulties, as the model struggled to converge. However, like our model, these were simple tests, suggesting that perhaps our architecture was not robust enough for the complexity of the task. This is a common issue as identifying the most optimal architectures of deep neural networks for PINNs based on their specific goal, has become its own area of research [40]. Our original idea consisted of combining our successful attention-based model from our SLA forecasting investigations to fit an SST field task similar to our temperature advection PDE, but was limited due to time constraints.

One potential solution that could have been explored but was cut due to time constraints, is the application of Fourier transformations to filter out high frequencies [41]. Focusing the model on larger wave patterns, and controlling the derivatives to prevent them from exploding, would allow the network to better approximate the wave function. However, these methods are computationally intensive, and with limited computational resources available to us as well as strict deadlines, we had to balance the complexity of the approach with more practical constraints. While alternatives like Jax could offer more advanced differentiation capabilities, the steep learning curve and time constraints led us to remain within the PyTorch framework for simplicity. Additionally, the replication of results across different studies is a pervasive issue, further highlighting the instability of our approaches as well as current methods in physics-informed machine learning.

6.6 Conclusion

While we were not able to achieve all of our objectives, we gained significant insight into the training process of PINNs and their relevance for certain dynamical fluid problems which is of great importance. Had time allowed, we would have continued our investigations using special PyTorch libraries designed for the training of PINNs, so that we may implement our own temperature advection PDE. Overall, our findings suggest that even though PINNs hold promise, they require further refinement and exploration beyond the scope of this six-month internship. This project has provided a solid foundation, but it is clear that solving partial differential equations with neural networks is a complex subject that demands more time and resources to fully explore and develop effective solutions.

Chapter 7

Conclusion

As we had outlined, our goal in this research project was to work on the prediction of SLA images using deep learning methods, whilst knowing that SLA data contains inherently noisy fields that obey dynamical fluid laws. In this study, we therefore proposed our *SLA-SST-SmaAt-UNet* model which is a modified variant of a smaller and attentive version of a UNet architecture, developed for the forecasting of SLA fields in the Gulf Stream. Through our work, we have successfully demonstrated the potential of combining deep learning techniques with physical insights to improve the forecasting of these SLA images. By integrating SST data, we enhanced the accuracy of short-term and long-term SLA predictions compared to both the persistence baseline and past models. Our findings confirmed that SST is a indeed complementary variable that, when incorporated into the predictive model, improves the precision of SLA forecasts, particularly at a 5-day horizon. On the other hand, the additional SST information seemed to render our model more reliable when forecasting across longer ranges, effectively reducing the margin of error. However, the majority of our improvements over the model-to-beat can be attributed to the incorporation of the CBAM blocks. The spatial attention and channel attention modules included within these blocks allow for a proper retention of key parts of the SLA and SST feature maps, such as darker eddy formations, while maintaining the temporal integrity of the image sequences. The darker features representing the crucial dynamics of sub-mesoscale currents were of key interest to us, and were effectively maintained by our model across the different prediction horizons. It could then also be hypothesized that the success of these attention mechanisms render the additional SST information obsolete. In either case, our work shows beyond doubt the success of a smaller, more efficient, attentive deep learning model in surface-level ocean forecasting tasks.

The first part of our study is a proof of concept that deep-learning methods can effectively model the state of the ocean surface for the near future, and hence the associated time-dependent PDEs. The transition to the investigation of the effect of solving for these PDEs directly within a model during training demonstrates the potential for improving the aforementioned proof of concept using physics-informed neural networks (PINN). While other models developed under less strict time constraints use both real image data and boundary conditions alongside their PDEs, our streamlined work surrounding a temperature advection field system shows great promise. It would be interesting to conduct further research using specific libraries and frameworks fit for physics-informed machine learning, and properly train a PINN with an adequately fitting deep learning network and corresponding computational resources. Some examples of these are NVIDIA's Modulus v.22.09 framework or the `pinnstorch` package implemented in PyTorch by Bafghi et al. [42] as proposed in their paper from 2023. For our study, we wanted to start from scratch for our experiments, but we emphasize that it would be highly relevant to use an existing module to improve performance. Even though significant work remains to be done on PINNs to properly solve and model multi-dimensional equations as complex the Navier-Stokes Equations, our efforts provide a first step in this endeavor.

Appendix A

Hyperparameters

Our training hyperparameters were determined iteratively via trial and error in our experiments. Below is a description of various hyperparameter settings that were tested during our experimental stage as well as the values that produced our final results.

- **Learning rate:** We begun tests with a learning rate value of 0.01 and tested as low as 0.00001 depending on the experiments. We found that a weighted Adam optimizer [43] with a learning rate of 0.0001 and weight decay of 0.01 produced the best results. This is accompanied by a learning rate scheduler that reduces the learning rate by a factor of 2 with a patience of 3 epochs.
- **Batch size:** We tested with various batch sizes, including 16, 32, 64, and 128. We noticed that our best results occurred with a batch size of 16. Due to the smaller size of our attention-based model, this did not significantly increase computation time.
- **Epochs:** We trained our two models over 200 epochs, but implemented an `EarlyStopping` callback which monitored the convergence of the validation loss and stopped training if the validation loss had not decreased in 8 epochs.
- **Activation function:** We used the SiLU activation function for our models since it has a global minimum and a non-monotonic increase (see Figure A.1), which allows it to effectively serve as a regularizer for high weights in our models [32].

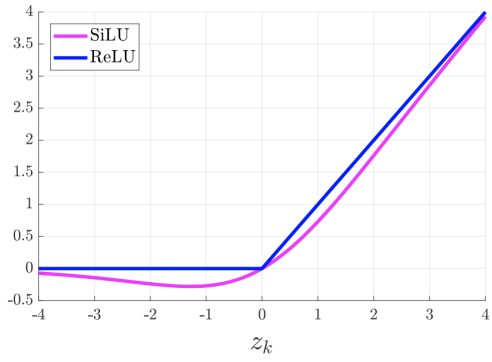

Figure A.1: Sigmoid Linear Units activation function for neural networks compared to Rectified Linear Units. The activation of the SiLU is computed by the sigmoid function multiplied by its input.

Appendix B

Further Results

Model	Metric	t + 5	t + 10	t + 15	t + 20
Persistence	RMSE (cm)	13.07	19.37	22.70	24.59
SLA-RES-UNet	RMSE (cm)	13.13	18.53	22.07	24.59
SLA-SmaAt-UNet	RMSE (cm)	9.31	15.63	18.94	21.67
SLA-SST-SmaAt-UNet	RMSE (cm)	8.93	15.97	19.11	20.87

Table B.1: Comparison of architecture performances with RMSE (cm) when forecasting at each time step individually on the dynamic region of the Gulf Stream (Latitude 33.7° to 44.42° N). The *SLA-RES-Unet* architecture is the pre-existing model by Luther Ollier (2022).

Model	Metric	t + 5	t + 10	t + 15	t + 20
SLA-SmaAt-UNet	RMSE (cm)	11.24	16.79	19.32	21.67
SLA-SST-SmaAt-UNet	RMSE (cm)	11.44	16.77	19.07	20.87

Table B.2: Comparison of architecture performances with RMSE (cm) at each time step on the dynamic region of the Gulf Stream (Latitude 33.7° to 44.42° N) when predicting strictly on sequences of 20 images.

Model	Metric	t + 5	t + 10	t + 15	t + 20
Persistence	MAE (cm)	8.51	13.10	15.83	17.58
SLA-RES-UNet	MAE (cm)	9.33	12.81	15.07	16.62
SLA-SmaAt-UNet	MAE (cm)	6.08	10.62	12.96	14.43
SLA-SST-SmaAt-UNet	MAE (cm)	6.03	10.91	13.17	14.59

Table B.3: Comparison of architecture performances with MAE (cm) when forecasting at each time step individually on the dynamic region of the Gulf Stream (Latitude 33.7° to 44.42° N).

Model	Metric	t + 5	t + 10	t + 15	t + 20
SLA-SmaAt-UNet	MAE (cm)	7.52	11.36	13.19	14.46
SLA-SST-SmaAt-UNet	MAE (cm)	8.02	11.65	13.31	14.42

Table B.4: Comparison of architecture performances with MAE (cm) at each time step on the dynamic region of the Gulf Stream (Latitude 33.7° to 44.42° N) when predicting strictly on sequences of 20 images.

Appendix C

Partial Differential Equations

C.1 Navier-Stokes Equations

The Navier-Stokes equations in two dimensions (2D) are given explicitly by

$$\begin{aligned} u_t + \lambda_1 \left(u u_x + v u_y \right) &= -p_x + \lambda_2 \left(u_{xx} + u_{yy} \right), \\ v_t + \lambda_1 \left(u v_x + v v_y \right) &= -p_y + \lambda_2 \left(v_{xx} + v_{yy} \right), \end{aligned} \tag{C.1}$$

where $u(t, x, y)$ denotes the x-component of the velocity field, $v(t, x, y)$ the y-component, and $p(t, x, y)$ the pressure. Here, $\lambda = (\lambda_1, \lambda_2)$ are the unknown parameters. Solutions to the Navier-Stokes equations are searched in the set of divergence-free functions; i.e.,

$$u_x + v_y = 0.$$

This extra equation is the continuity equation for incompressible fluids that describes the conservation of mass of the fluid. We make the assumption that

$$u = \psi_y, \quad v = -\psi_x,$$

for some latent function $\psi(t, x, y)$. Under this assumption, the continuity equation will be automatically satisfied. Given noisy measurements

$$\{t^i, x^i, y^i, u^i, v^i\}_{i=1}^N$$

of the velocity field, we are interested in learning the parameters λ as well as the pressure $p(t, x, y)$. We define $f(t, x, y)$ and $g(t, x, y)$ to be given by

$$\begin{aligned} f &:= u_t + \lambda_1 (u u_x + v u_y) + p_x - \lambda_2 (u_{xx} + u_{yy}), \\ g &:= v_t + \lambda_1 (u v_x + v v_y) + p_y - \lambda_2 (v_{xx} + v_{yy}), \end{aligned} \tag{C.2}$$

and proceed by jointly approximating $\psi(t, x, y)$, $p(t, x, y)$ using a single neural network with two outputs. This prior assumption results into a physics informed neural network $[f(t, x, y),\ g(t, x, y)]$. The parameters λ of the Navier-Stokes operator as well as the parameters of the neural networks $[\psi(t, x, y),\ p(t, x, y)]$ and $[f(t, x, y),\ g(t, x, y)]$ can be trained by minimizing the mean squared error loss

$$\begin{aligned} MSE := \frac{1}{N} \sum_{i=1}^{N} \left(\left| u(t^i, x^i, y^i) - u^i \right|^2 + \left| v(t^i, x^i, y^i) - v^i \right|^2 \right) \\ + \frac{1}{N} \sum_{i=1}^{N} \left(\left| f(t^i, x^i, y^i) \right|^2 + \left| g(t^i, x^i, y^i) \right|^2 \right). \end{aligned} \tag{C.3}$$

C.2 Burger's Equation

In one space dimension, the Burger's equation along with Dirichlet boundary conditions reads as

$$\begin{aligned} &u_t + uu_x - \left(\frac{0.01}{\pi}\right) u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1], \\ &u(0, x) = -\sin(\pi x), \\ &u(t, -1) = u(t, 1) = 0. \end{aligned} \tag{C.4}$$

Let us define $f(t, x)$ to be given by

$$f := u_t + uu_x - \left(\frac{0.01}{\pi}\right) u_{xx}, \tag{C.5}$$

and proceed by approximating $u(t, x)$ by a deep neural network. The shared parameters between the neural networks $u(t, x)$ and $f(t, x)$ can be learned by minimizing the mean squared error loss

$$MSE = MSE_u + MSE_f, \tag{C.6}$$

where

$$MSE_u = \frac{1}{N_u} \sum_{i=1}^{N_u} \left| u(t_u^i, x_u^i) - u^i \right|^2, \tag{C.7}$$

and

$$MSE_f = \frac{1}{N_f} \sum_{i=1}^{N_f} \left| f(t_f^i, x_f^i) \right|^2. \tag{C.8}$$

Here, $\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}$ denote the initial and boundary training data on $u(t, x)$ and $\{t_f^i, x_f^i\}_{i=1}^{N_f}$ specify the collocation points for $f(t, x)$. The loss MSE_u corresponds to the initial and boundary data while MSE_f enforces the structure imposed by the Burgers' equation at a finite set of collocation points.

Bibliography

[1] Zachary K. Erickson, Erik Fields, Leah Johnson, Andrew F. Thompson, Lilian A. Dove, Eric D'Asaro, and David A. Siegel. Eddy tracking from in situ and satellite observations. *Journal of Geophysical Research: Oceans*, 128(8):e2023JC019701, 2023. e2023JC019701 2023JC019701.

[2] Evangelos Moschos, Olivier Schwander, Alexandre Stegner, and Patrick Gallinari. Deep-sst-eddies: A deep learning framework to detect oceanic eddies in sea surface temperature images. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4307–4311, 2020.

[3] Xiangming Zeng, Yizhen Li, and Ruoying He. Predictability of the loop current variation and eddy shedding process in the gulf of mexico using an artificial neural network approach. *Journal of Atmospheric and Oceanic Technology*, 32(5):1098 – 1111, 2015.

[4] M.-I. Pujol, Y. Faugère, G. Taburet, S. Dupuy, C. Pelloquin, M. Ablain, and N. Picot. Duacs dt2014: the new multi-mission altimeter data set reprocessed over 20 years. *Ocean Science*, 12(5):1067–1090, 2016.

[5] J. Isern-Fontanet, J. Ballabrera-Poy, A. Turiel, and E. García-Ladona. Remote sensing of ocean surface currents: a review of what is being observed and what is being assimilated. *Nonlinear Processes in Geophysics*, 24(4):613–643, 2017.

[6] Alexandre Immas, Ninh Do, and Mohammad-Reza Alam. Real-time in situ prediction of ocean currents. *Ocean Engineering*, 228:108922, 2021.

[7] Georgy E. Manucharyan, Lia Siegelman, and Patrice Klein. A deep learning approach to spatiotemporal sea surface height interpolation and estimation of deep currents in geostrophic ocean turbulence. *Journal of Advances in Modeling Earth Systems*, 13(1):e2019MS001965, 2021. e2019MS001965 2019MS001965.

[8] Cristina González-Haro, Jordi Isern-Fontanet, Pierre Tandeo, and René Garello. Ocean surface currents reconstruction: Spectral characterization of the transfer function between sst and ssh. *Journal of Geophysical Research: Oceans*, 125(10):e2019JC015958, 2020. e2019JC015958 10.1029/2019JC015958.

[9] Jordi Isern-Fontanet, B Chapron, Guillaume Lapeyre, and P Klein. Potential use of microwave sst for the estimation of surface ocean currents. *Geophysical Research Letters - GEOPHYS RES LETT*, 33, 12 2006.

[10] Daniele Ciani, Marie-Helene Rio, Bruno Buongiorno Nardelli, Helene Etienne, and Rosalia Santoleri. Improving the altimeter-derived surface currents using sea surface temperature (sst) data: A sensitivity study to sst products. *Remote Sensing*, 12:1601, 05 2020.

[11] Théo Archambault, Anastase Alexandre Charantonis, Dominique Béréziat, Carlos Mejia, and Sylvie Thiria. Ssh super-resolution using high resolution sst with a subpixel convolutional residual network. In *Climate Informatics*, 01 2022.

[12] Sylvie Thiria, Charles Sorror, Théo Archambault, Anastase Alexandre Charantonis, Dominique Béréziat, Carlos Mejia, Jean-Marc Molines, and Michel Crépon. Downscaling of ocean fields by fusion of heterogeneous observations using deep learning algorithms. *Ocean Modelling*, 182:102174, 04 2023.

[13] Jingjing Liu, Baogang Jin, Lei Wang, and Lingyu Xu. Sea surface height prediction with deep learning based on attention mechanism. *IEEE Geoscience and Remote Sensing Letters*, PP:1–5, 12 2020.

[14] Anne Braakmann-Folgmann, Ribana Roscher, Susanne Wenzel, Bernd Uebbing, and Jürgen Kusche. Sea level anomaly prediction using recurrent neural networks. In *Proceedings of the 2017 conference on Big Data from Space*, 10 2017.

[15] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[16] Said Ouala, Ronan Fablet, Lucas Drumetz, Bertrand Chapron, Ananda Pascual, Fabrice Collard, and Lucile Gaultier. Physically informed neural networks for the simulation and data-assimilation of geophysical dynamics. In *2020 IEEE International Geoscience and Remote Sensing Symposium*, pages 3490–3493, 09 2020.

[17] Taco de Wolff, Hugo Carrillo Lincopi, Luis Martí, and Nayat Sánchez-Pi. Towards optimally weighted physics-informed neural networks in ocean modelling. *CoRR*, abs/2106.08747, 2021.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[19] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Oriol Vinyals, Jacklynn Stott, Alexander Pritzel, Shakir Mohamed, and Peter Battaglia. Graphcast: Learning skillful medium-range global weather forecasting, 2023.

[20] Kevin Trebing and Siamak Mehrkanoon. Smaat-unet: Precipitation nowcasting using a small attention-unet architecture. *CoRR*, abs/2007.04417, 2020.

[21] Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R. Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, Remi Lam, and Matthew Willson. Gencast: Diffusion-based ensemble forecasting for medium-range weather, 2024.

[22] Stephan Rasp, Stephan Hoyer, Alexander Merose, Ian Langmore, Peter Battaglia, Tyler Russel, Alvaro Sanchez-Gonzalez, Vivian Yang, Rob Carver, Shreya Agrawal, Matthew Chantry, Zied Ben Bouallegue, Peter Dueben, Carla Bromberg, Jared Sisk, Luke Barrington, Aaron Bell, and Fei Sha. Weatherbench 2: A benchmark for the next generation of data-driven global weather models, 2024.

[23] Luther Ollier, Sylvie Thiria, Anastase Charantonis, Carlos E.Mejia, and Michel Crepon. Neural network approaches for sea surface height predictability using sea surface temperature. *World Academy of Science, Engineering and Technology International Journal of Environmental and Ecological Engineering, Vol:18, No:01*, 2023.

[24] Sifan Wang, Shyam Sankaran, Hanwen Wang, and Paris Perdikaris. An expert's guide to training physics-informed neural networks, 2023.

[25] Stephen M. Griffies and Alistair J. Adcroft. *Formulating the Equations of Ocean Models*, pages 281–317. American Geophysical Union (AGU), 2008.

[26] Taikang Yuan, Junxing Zhu, Wuxin Wang, Jingze Lu, Xiang Wang, Xiaoyong Li, and Kaijun Ren. A space-time partial differential equation based physics-guided neural network for sea surface temperature prediction. *Remote Sensing*, 15(14), 2023.

[27] Olivier Pannekoucke and Ronan Fablet. Pde-netgen 1.0: from symbolic partial differential equation (pde) representations of physical processes to trainable neural network representations. *Geoscientific Model Development*, 7, 2021.

[28] E.U. Copernicus Marine Service Information (CMEMS). Marine Data Store (MDS). Global ocean physics reanalysis. Available online: https://doi.org/10.48670/moi-00021. Accessed: 02/05/2024.

[29] Henry B. Mann. Nonparametric tests against trend. *Econometrica*, 13(3):245–259, 1945.

[30] Dr. Manish Meena. Rainfall statistical trend and variability detection using mann-kendall test, sens slope and coefficient of variance - a case study of udaipur district (1957-2016). *Applied Ecology and Environmental Sciences*, 8(1):34–37, 2020.

[31] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[32] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *CoRR*, abs/1702.03118, 2017.

[33] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: convolutional block attention module. *CoRR*, abs/1807.06521, 2018.

[34] Dujuan Kang and Enrique Curchitser. Gulf stream eddy characteristics in a high-resolution ocean model: Gulf stream eddy characteristics. *Journal of Geophysical Research (Oceans)*, 118:4474–4487, 09 2013.

[35] Nickolay Y. Gnedin, Vadim A. Semenov, and Andrey V. Kravtsov. Enforcing the courant–friedrichs–lewy condition in explicitly conservative local time stepping schemes. *Journal of Computational Physics*, 359:93–105, 2018.

[36] Guillaume Laibe and Maxime Lombart. On the Courant-Friedrichs-Lewy condition for numerical solvers of the coagulation equation. *Monthly Notices of the Royal Astronomical Society*, 510:5220–5225, 2022.

[37] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[38] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert L. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[39] Atilim Gunes Baydin, Barak A. Pearlmutter, and Alexey Andreyevich Radul. Automatic differentiation in machine learning: a survey. *CoRR*, abs/1502.05767, 2015.

[40] Ana Kaplarević-Mališić, Branka Andrijević, Filip Bojović, Sran Nikolić, Lazar Krstić, Boban Stojanović, and Miloš Ivanović. Identifying optimal architectures of physics-informed neural networks by evolutionary strategy. *Applied Soft Computing*, 146:110646, 2023.

[41] Zongyi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *CoRR*, abs/2010.08895, 2020.

[42] Reza Akbarian Bafghi and Maziar Raissi. PINNs-torch: Enhancing speed and usability of physics-informed neural networks with pytorch. In *The Symbiosis of Deep Learning and Differential Equations III*, 2023.

[43] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.