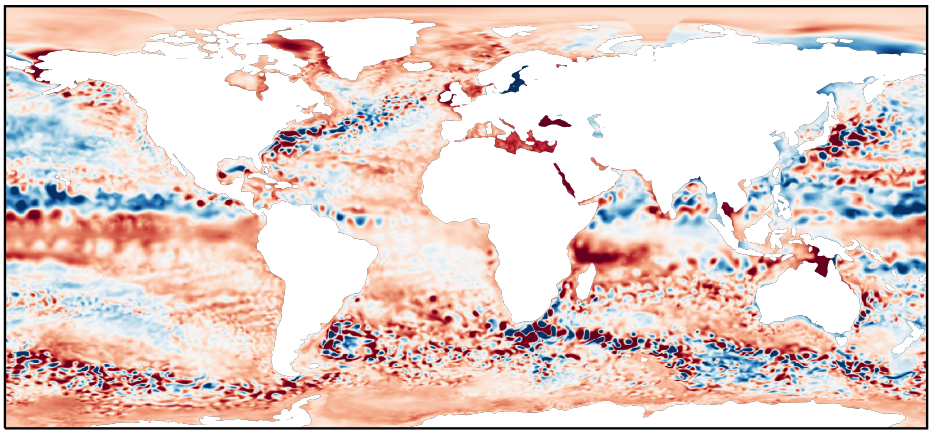
M2 RESEARCH INTERNSHIP REPORT

Identification and filtering of oceanic chaos by Machine Learning



Mickaël LALANDE

Master 2 in Earth, Planetary and Environmental Sciences

Atmosphere-Climate-Continental Landmass Programme

UFR PhITEM - Grenoble Alpes University

Supervisors: Thierry Penduff and Redouane Lguensat

Institute of Environmental Geosciences (IGE), Grenoble, France

MEOM Research Group

04/02/2019 - 28/06/2019 (5 months)

UFR PhITEM

UNIVERSITÉ Grenoble Alpes

OSUG

Observatoire des Sciences de l'Univers de Grenoble

Master in Earth, planetary and Environmental Sciences

Non-plagiarism certificate

I, the undersigned (First name, FAMILY NAME)
Mickaël LALANDE

Author of the report entitled (Title)
Identification and filtering of oceanic chaos by Machine Learning

Declare that the above-cited report results from my personal work and that I have neither forged, falsified nor copied all or part of another persons work to present it as mine.

All sources of information used and all author citations have been included following standard usage.

I am aware of the fact that failing to cite a source or failing to cite it fully and properly constitutes plagiarism, and that plagiarism is considered a serious offence within the university that can be sanctioned severely by law.

At (place) Saint-Martin-d'Hères,
the (date) May 14, 2019

Student's signature

Acknowledgments

I would like to thank a lot my supervisors: Thierry Penduff for putting his trust on my ambition in experimenting a subject that combines machine learning and oceanography. I was excited to learn about machine learning and explore the potential of these methods on a geoscientific data set. Besides that, I had the chance to learn and discover an oceanic simulation that was a great asset to complete my past atmosphere and cryosphere studies. Thierry Penduff was always there to share with me his physical oceanography knowledge. Also, big thanks to Redouane Lguensat who made me discover everything about machine learning and shared with me his expertise. He always had ideas when we had issues, and helped pushing the project forward to this current state. Both Thierry Penduff and Redouane Lguensat were complementary supervisors from different backgrounds, and made this work go beyond what I imagined, so thanks again to them.

I would like also to thanks all the other members of the MEOM team who work all together in with great energy. In particular, big thanks to Sally Close whose work was the precursor of this study. She was always open for discussions and helped me grasp ideas related to her work. I would like to thank her a lot for her patience and her pedagogy. Thanks to Jean-Marc Molines for always being available to help accessing the data necessary for the project. Many thanks also to Laurent Brodeau for making me discover his interpolation tool Sosie[1] that allows a better interpolation on the ORCA grids types. Thanks for his time trying to make it work on the observational data. Thanks to Adekunle Ajayi for his help on spectral analysis and sharing his PowerSpec tool[2] developed with Julien Le Sommer that allowed me to save much time and go further on my result analysis. Thanks to Aurélie Albert for making me discover Dask combined with xarray and thanks to all the other people I shared some time with during this internship.

I am also grateful to Mondher Chekki who allowed me to use the GRICAD computational services from the Grenoble Alpes University, and without whom I could not have used global data and pursued this study as far as I went. Thanks also to Laure Zanna who paid a visit to our lab, and accepted to exchange and discuss about our results and the method presented here. Thanks to Josiane Brasseur for her efficiency in managing administrative work in the team. Thanks also to Audrey Monsimer, my office colleague, with whom I had great exchanges. Last but not least, special thanks to Nicolas Le Bihan for kindly accepting to review the present report.

[1] `https://github.com/brodeau/sosie`

[2] `https://github.com/adeajayi-kunle/PowerSpec`

Abstract

The variability of the oceanic surface (sea-level anomalies or SLA) observed by satellite altimeters combines two components: a "forced" component determined by the atmospheric variability, and an "intrinsic" component emerging from the ocean itself with a chaotic behavior. Our goal is to estimate the forced variability in global SLA observations (1993-2017) through a deep learning algorithm trained on an existing ensemble ocean simulation. Our algorithm estimates the forced model variability with a 2 cm RMS error and a temporal correlation of 0.9, a skill similar to that of a bandpass filter proposed recently, but with an ability to adapt its transfer function locally and identify larger-scale intrinsic patterns. The method is applied to altimetric observations, and shown to significantly increase the variance of winter SLA explained by the variance of the North Atlantic Oscillation (NAO) index. While the present algorithm could be further improved, this study proposes a new denoising approach to reveal deterministic signals, with encouraging results for the exploration of geophysical applications of artificial intelligence.

Résumé

La variabilité de la surface océanique (anomalies de hauteur de mer ou SLA) observée par les altimètres satellites comporte deux composantes : une composante « forcée » déterminée par la variabilité atmosphérique et une composante « intrinsèque » émergeant de l'océan lui-même avec un comportement chaotique. Notre objectif est d'estimer la variabilité forcée dans les observations globales de SLA (1993-2017) à l'aide d'un algorithme d'apprentissage profond entrainé sur une simulation d'ensemble océanique existante. Notre algorithme estime la variabilité forcée du modèle avec une erreur RMS de 2 cm et une corrélation temporelle de 0,9; performance similaire à celle d'un filtre passe-bande proposé récemment, mais avec une capacité d'adaptation locale de sa fonction de transfert et d'identification de la variabilité intrinsèque à de plus grandes échelles. La méthode est ensuite appliquée aux observations altimétriques et montre que la variance de la SLA expliquée par la variance de l'indice de l'Oscillation Atlantique Nord (NAO) augmente de manière significative en hiver. Bien que l'algorithme actuel soit perfectible, cette étude propose une nouvelle approche de débruitage permettant de révéler des signaux déterministes, avec des résultats encourageants pour l'exploration d'applications géophysiques de l'intelligence artificielle.

Contents

1 Introduction

Historically, the ocean has been seen as passive, meaning that it was only responding to the atmospheric forcing or the atmospheric/ocean coupling. However, eddying-permitting modeling studies have revealed that an intrinsic variability emerges from the ocean itself at various time and space scales. For example, Penduff et al. (2011) has shown with a NEMO-based 1/4° global ocean/sea-ice simulation driven for 327 years by a repeated climatological atmospheric forcing that a strong, low-frequency, intrinsic variability spontaneously emerges from the ocean itself. Thus, the modern view of the ocean's dynamics comprises the combination of a deterministic atmospherically-forced component and an intrinsic component.

In order to better characterize the intrinsic variability of the ocean, the OCCIPUT[3] (Oceanic Chaos - ImPacts, strUcture, predicTability) project team performed a pioneering 50-member ensemble of 1/4° global ocean/sea-ice simulations (covering the period 1960-2015), starting from perturbed initial conditions then driven by the same atmospheric forcing for all the 50 members. The quick divergence of all members and loss of correlation (random phase) between them demonstrated the chaotic behavior of the ocean[4]. This experiment allowed the atmospherically-forced component (variability of the ensemble mean), and the intrinsic component (difference between any given member and the ensemble mean), characterized by the ensemble standard deviation, to be estimated. The results revealed and confirmed the importance of the oceanic chaos up to large spatiotemporal scales (multiple decades and basin-scale) and affect several climate-relevant variables, such as the Atlantic meridional overturning circulation (AMOC) (Leroux et al., 2018), sea surface temperature (SST), ocean heat content (OHC) (Penduff et al., 2018), and sea surface height (SSH) (Close et al., 2019).

The main objective of this study is to estimate the atmospherically-forced component from observed SSH fields derived from altimetry. The observed SSH evolution is an important climate indicator (Church et al., 2013), but its forced component (e.g. the anthropogenic sea-level rise) can be locally hidden by its intrinsic component (Llovel et al., 2018). Close et al. (2019) have already succeeded in estimating both components (forced and intrinsic) for SSH fields with a simple band-pass filter with appropriate cutoff scales. In this study, we aim to test the ability of a data-driven approach based on machine learning to reach the same goal, hopefully with improved skill.

Interest in machine learning has increased considerably in recent years, and several studies have previously applied it in geoscience. Examples include, but are not limited to: numerical climate modeling and

[3] `https://meom-group.github.io/projects/occiput/`

[4] In this report, we will mainly mention "intrinsic variability"; however, it has to be kept in mind that this intrinsic variability also has a chaotic nature.

weather prediction (Krasnopolsky and Fox-Rabinovitz, 2006), typhoon-ocean feedback parameterization (Jiang et al., 2018), air quality modeling (Keller et al., 2017), etc. More recently, Bolton and Zanna (2019) used machine learning in oceanography for data inference and subgrid parameterization, resulting in encouraging results.

The focus of this study is to examine the possibility of using machine learning, more precisely deep convolutional networks, for extracting the intrinsic variability from observed SLA fields, using the OCCIPUT ensemble simulation hindcast introduced above, and also used in Close et al. (2019). The objective is to investigate how a classical machine learning method coupled with the available amount of data can provide a new method for filtering intrinsic oceanic variability.

This report is presented as follows: in section 2 we present the ensemble experiment, the observational data, and the machine learning method used in this study. Section 3 shows the predictions of the SLA forced signal extracted by the machine learning algorithm and compares it with the filter used in Close et al. (2019). Improvements and suggestions concerning the machine learning method are discussed in Section 4. A summary and conclusions are given in Section 5.

2 Data and methods

2.1 Model and observational data

The model data used in this study are the outputs from the OCCIPUT experiment; the motivation and methodology are described in Penduff et al. (2014) and Bessières et al. (2017). It is a set of 50 ensemble members of global ocean-sea ice hindcasts driven by historical atmospheric forcing using the Drakkar Forcing Set DFS5.2 (Dussin et al., 2016). The model configuration is based on NEMO v.3.5 (Madec, 2012), with 0.25° horizontal resolution and 75 vertical levels. The outputs are saved as 5-day averages. The spread of the ensemble is generated by applying a stochastic perturbation on the density equation over one year (Brankart et al., 2015), which is then shut off.

In this study, we focus on the SSH variable on the period 1979-2015 (37 years), and we work directly on the tripolar ORCA025 model grid at 1/4° resolution. This grid has the advantage of being quasi-isotropic and allows the ocean dynamics to be better resolved at high latitudes (Bernard et al., 2006). The model data have been preprocessed as described in Close et al. (2019): the global mean SSH signal has been subtracted from each member at each time step and the long-term trends have been removed by applying a LOWESS filter (Cleveland, 1979). This processing yields 50 versions of 5-day mean SLA maps covering

a 37-year period.

Observational SLA data are derived from altimetry. We use the 0.25° Merged-Gridded Sea Level Anomalies from the DT-MSLA-H Global product, distributed by the Copernicus Marine Environment Monitoring Service over the period 1993-2017 with the same 5-day mean temporal resolution.

2.2 Definition of the forced and intrinsic contributions

The total SLA is a complex signal having different contributions emerging from the atmospheric forcing or generated by the ocean itself. It is impossible to separate both components from observations alone. However, the ensemble simulation allows an estimation of the atmospherically-forced variability to be made via the ensemble mean: it is the common signal for all the members. This definition is not perfect and can be questionable in certain areas where, for example, ensemble distributions are bimodal or where the intrinsic variability is much larger than the forced variability. However, for most of the ocean it is a good estimator, and this study is based on this assumption. Figure 1 shows the different components for one snapshot on the model grid. We will consider this as the ground truth used to train the machine learning algorithm.

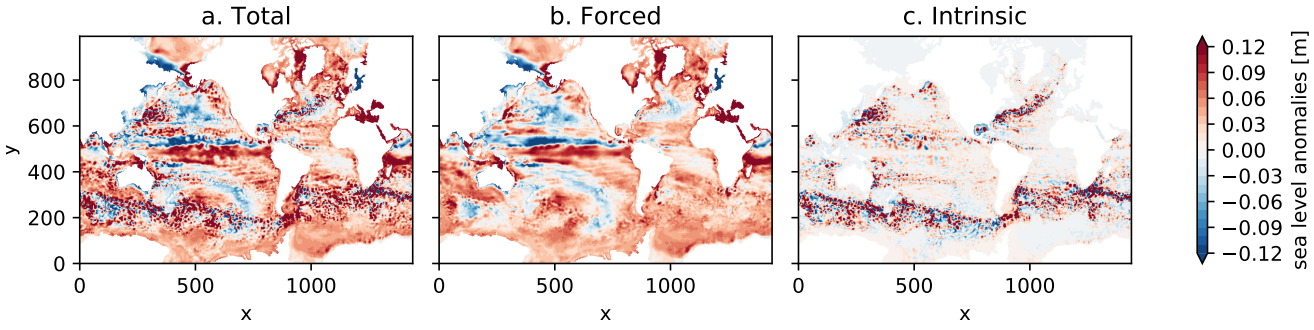


Figure 1: Example of SLA contributions (January 3, 1979; member 50): (a) total signal, (b) forced signal (ensemble mean) and (c) intrinsic signal (total-forced). The data are taken from the OCCIPUT ensemble, and represented on the ORCA025 model grid.

2.3 Machine Learning method

The method aims to extract the forced signal from the total signal of a single ensemble member. It is computationally impossible to handle all data at the same time (more than 300 Go). Thus, we choose some subareas that fit into the available GPU memory, and that are physically interesting based on the results of Close et al. (2019). Figure 2.b shows that the intrinsic contribution (dark zones) extends from

less than 50 km towards spatial scales larger than 2000 km. We thus decided to take zones large as at least two times this spatial scale: 4000 km that correspond to zones of about 160x160 pixels in the ORCA025 model grid[5].

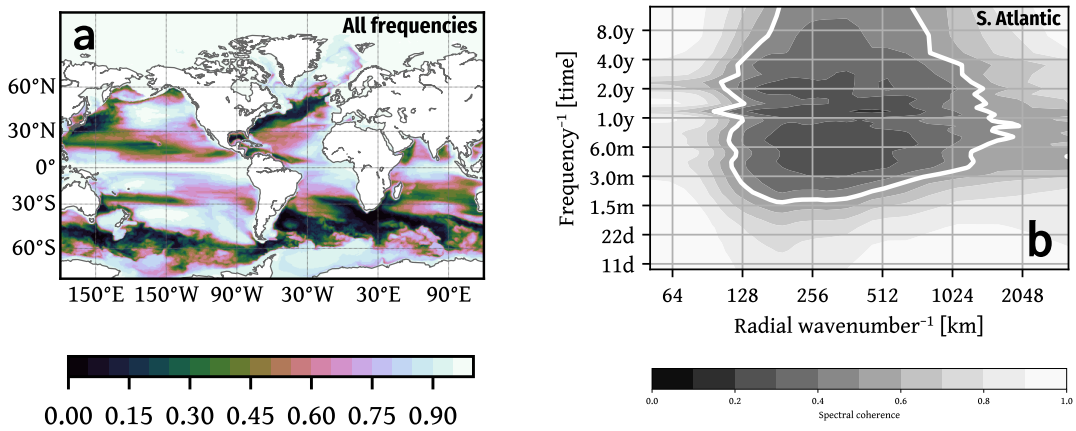


Figure 2: Adapted from Close et al. (2019): (a) Ratio of forced to total variance for all time scales, (b) Spectral coherence between the forced and total signals in radially-averaged wavenumber frequency space in the South Atlantic. The thick white lines show the coherence=0.5 contours.

In practice, we worked on the original model grid ORCA025 without taking into account the latitudes and longitudes, and we fed the algorithm with snapshots of the SLA field independently from each other. Figure 3 shows the training zones, which were chosen incrementally starting from zone1 (subset of the South Atlantic Ocean): an area where the forced to intrinsic ratio is very contrasted (Figure 2.a). Then, we added zones where the error was high during our previous predictions. We hypothesize that the patterns learned from these zones are representative of the wider oceanic variability. Furthermore, we do not take into account the deformation of the structures due to the model grid.

To perform the training we chose a supervised regression learning method (described in Sections 2.4 and 2.5). The training is performed on the period 1979-2008 (30 years). The first 40 members are used for the training (corresponding to 87600 samples for one zone) and the remaining 10 members are used for the validation (21900 samples for one zone). The last 7 years (2009-2015) are used for testing the predictive skill of the algorithm on another period. The total SLA fields (Figure 1.a) are given as an input of the algorithm and the corresponding forced SLA fields (Figure 1.b) are used as an output to supervise our

[5] This size was initially impossible to handle using MEOM's internal computing resources; fortunately, we were granted access to the supercomputing and data infrastructure GRICAD of the Grenoble Alpes University which offers about 200 GB of RAM and the possible parallelization on 4 NVIDIA V100 GPUs (with 32 GB of VRAM), hence making this study possible. We performed preliminary tests on smaller regions not presented on this report but available on GitHub: `https://github.com/Doremiska/M2_Internship`.

machine learning algorithm.

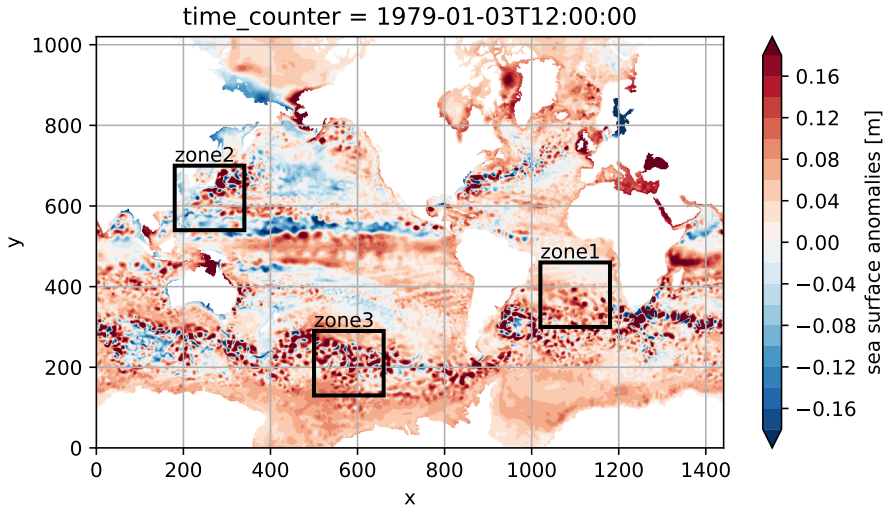


Figure 3: Training zones based on the ORCA025 model grid, that correspond to 160x160 pixels (approximately 4000x4000 km). See Table 1 for details on zones.

zone	**x**	**y**	**≃ longitude**	**≃ latitude**
1	1020-1180	300-460	32°W-7°E	44°S-10°S
2	180-340	540-700	117°E-157°E	10°N-44°N
3	500-660	130-290	162°W-122°W	67°S-46°S

Table 1: Zones pixel cuts values from ORCA025 model grid and approximate longitude and latitude equivalents.

2.4 Predictive algorithm: U-Net

Machine learning is a form of statistical modeling that has witnessed an enormous interest in the last decade thanks to two crucial factors:

- New developments in optimizing ANNs (Artificial Neural Neworks) with billions of parameters which led to the birth of Deep Learning, a family of techniques that sparked off major breakthroughs and state-of-the-art results in several signal/image processing applications.
- The proliferation of data and the increasing computational capabilities (HPC, GPUs, etc.).

Deep Learning is a vibrant area of research, and numerous techniques and neural architectures are designed and tested and published in the literature at a very rapid rate. Because of this, the choice of a deep learning

algorithm is not straightforward. To analyze the two-dimensional structures of SLA, we decided to focus on the use of deep Convolutional Neural Networks (CNN) because of their ability to efficiently handle 2D data using locality assumptions, and therefore reducing the number of parameters needed for optimization (Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; Dong et al., 2016). From this family, many different architectures exist in the literature, we tried, for example: the ResNet architecture (He et al., 2015), an architecture inspired from Bolton and Zanna (2019), plain convolutional encoder-decoder, and the U-Net architecture (Ronneberger et al., 2015). We found that the architecture inspired by Bolton and Zanna (2019) and the U-Net architecture performed best in our preliminary study[6]. However, we had to abandon the algorithm inspired by Bolton and Zanna (2019) because it is not fully convolutional. It includes fully connected layers at the end of the architecture that require a huge number of parameters and cannot be used on large zones[7]. Therefore, the main algorithm adopted for this study is based on the U-Net architecture (Figure 4), described below.

The first step of the U-Net architecture consists of applying a convolutional operation (described on Figure 5.a). This operation consists of multiplying the convolution filters weights with the pixels of the input image (total SLA) and adding a bias (not shown). The first step of Figure 4 shows an example with 32 convolutional filters that are applied successively on the input image; the feature maps (results of the convolution process) are then stacked (first blue box with a volume of 160x160x32). To keep the same image size at the output, a zero padding of 1 pixel around the image needs to be added, before using 3x3 filters. The convolutional filters scan the image with two overlapping pixels. Just before the production of the feature map, an activation function: $f(x) = max(0, x)$ (ReLU, Nair and Hinton (2010)) is applied after each convolution operation to introduce nonlinearities. The second step in Figure 4 performs the same operation as the first one; however, instead of having a 3x3 convolutional filter applied on the input image, a 3x3x32 convolutional volume filter is applied on the previous feature map (160x160x32), followed by the ReLU activation function. The third step is a max pooling operation (described in Figure 5.b), which consists of taking the absolute maximum value out of 2x2 pixels. This operation reduces the image size by 2 in both dimensions. Thus, in the Figure 4 example, the feature map before step 3 of dimensions 160x160x32 is transformed to have dimensions 80x80x32 (the last dimension is not affected). Then, the next purple arrows of Stage 1 and 2 consist of the same convolutional operation of step 2. In practice, the number of filters and stages can be variable.

[6] `https://github.com/Doremiska/M2_Internship`

[7] The number of parameters of a fully connected layer depends on the image size, while for a fully convolutional CNN, the image input does not matter. This is highly appreciated for our use case, where we want to train on small regions, but still perform inference on global images without altering the architecture.

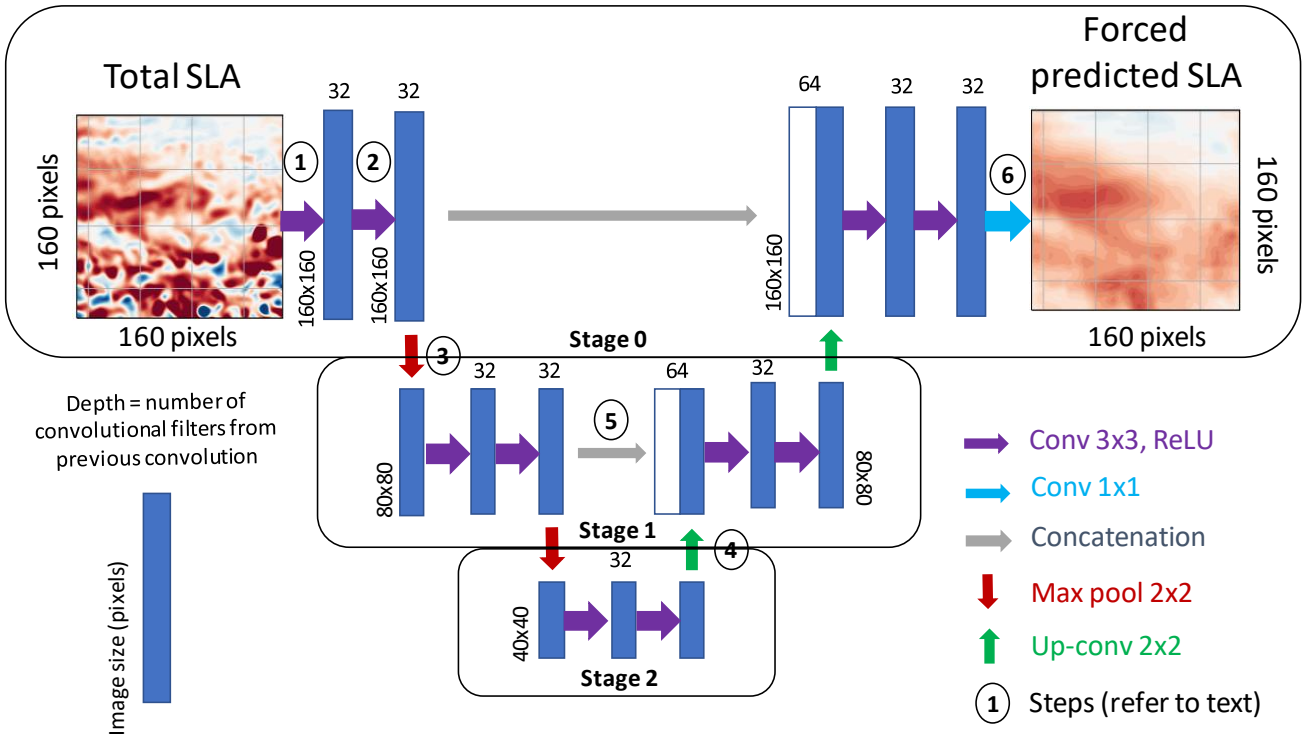


Figure 4: Adapted from Ronneberger et al. (2015): U-Net architecture used in this study. The input corresponds to one snapshot of total SLA and the output is the forced predicted SLA. Each blue box corresponds to a feature map (volume corresponding to the image size times the number of stacked images). White boxes represent copied feature maps. The arrows denote the different operations.

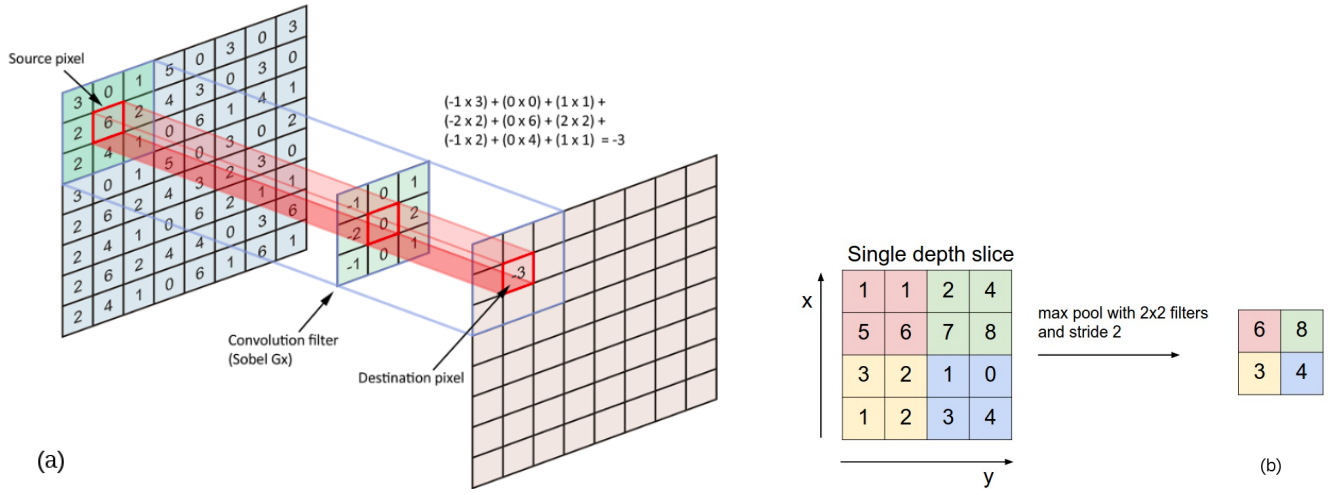


Figure 5: Example of (a) convolutional operation: the convolution filter weights are applied around the source pixel and summed to the destination pixel.—Source: `https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2`; (b) max pooling operation: takes the largest value.—Source: `http://cs231n.github.io/convolutional-networks/`.

After the encoding part of the U-Net architecture, the information is expanded back through upsampling operations (green arrows Up-conv 2x2). In our study, we used a bilinear interpolation followed by a convolution operation. As an example, step 4 of Figure 4, consists of expanding the 40x40x32 feature map to 80x80x32 with a bilinear interpolation on the image size dimension. Then, a convolution operation is applied as in step 2 (included in the green arrow), resulting in the blue box above the green arrow. Then, before applying the next convolutional operation (step 5), we concatenate the last feature map of the same stage from the encoding part of the U-Net with the actual feature map, resulting in a new feature map of dimensions 80x80x64 (white and blue boxes). The following operations of the decoding part are repeated, until step 6, which consists of applying a convolution operation with a single 1x1x32 convolution filter without applying the activation function (equivalent to a linear combination for each pixel), resulting in one snapshot of the predicted forced SLA.

U-Net architecture originated from the biomedical imaging community and quickly made its way to other applications. It is a baseline choice for image segmentation and other regression problems due to the use of skip connections, which have been proved to make the optimization easier, and ensure that not all the information from the encoding path is lost, especially fine-scale structures. The rationale behind the use of this architecture for our use case is that we expect that the encoding part will tend to remove the intrinsic variability from the total SLA and preserve the forced signal. The concatenation operation allows the preservation of some fine structures into the final result.

2.5 U-Net optimization

To optimize the U-Net parameters, we used a classical stochastic gradient descent variant: Adam (Kingma and Ba, 2014) with an adaptive learning rate starting from 10^{-5} to 10^{-6} (being divided by 2 every 20 iterations without improvements on the validation loss). The cost function is defined as the mean square error (MSE) between the true and predicted forced signals, summed over all training examples. Then, the training of the neural network iteratively adjusts the values of the weights (values inside the filters) from an initial random initialization. We use a batch size of 32 (number of training samples used at each iteration for the stochastic gradient method) and 300 epochs (number of times the algorithm pass through all data for the optimization) for the training. After 300 epochs, the gain becomes negligible.

We trained and implemented each neural network using Keras (Chollet, 2015) with the Tensorflow backend (Abadi et al., 2016). We worked directly with the SLA as an input and replaced the land areas by zeros. This implies that the algorithm will also learn the shapes of the coasts and some non-oceanic patterns. We used the first zone for choosing an architecture type that we applied for all this study. Figure 6 shows

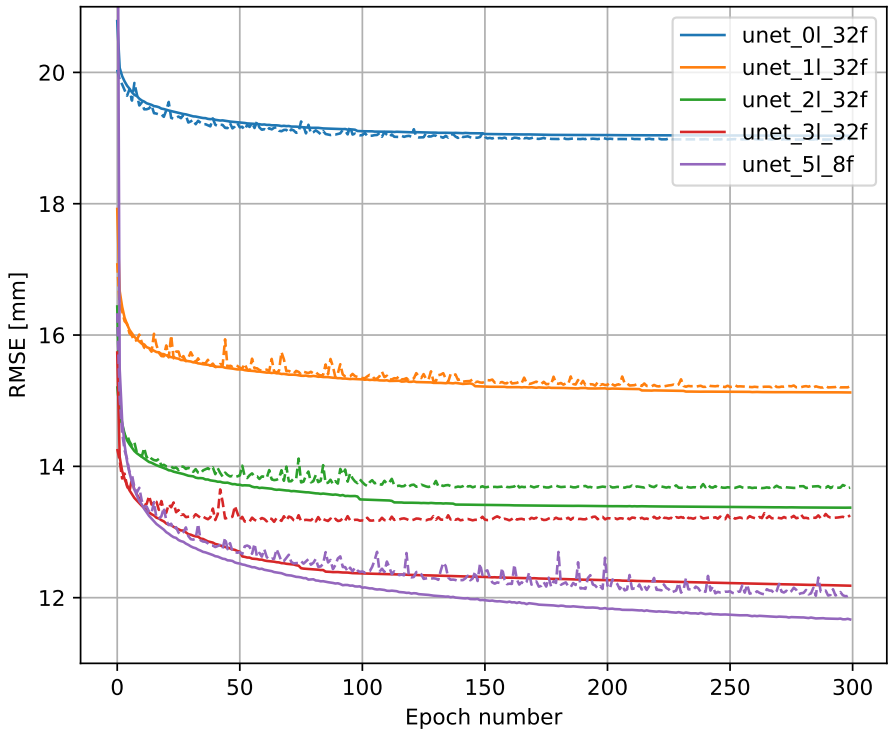


Figure 6: Loss functions for the training (solid lines) and validation (dashed lines) data set of different model architectures. Training data set: members 1 to 40, and validation data set: members 41 to 50 on the period 1979-2008. The labels correspond to: model name_number of stages_number of filters. For more details on architectures see Table 2. The loss in shown here with the RMSE, however the minimisation was done on the MSE.

Model name	**Number of parameters**	**Time per epoch [s]**	**Validation RMSE at epoch 300 [mm]**
unet_0l_32f	9,601	38	18.975
unet_1l_32f	59,937	110	15.220
unet_2l_32f	**110,273**	**130**	**13.677**
unet_3l_32f	160,609	138	13.254
unet_5l_8f	16,553	69	12.042

Table 2: Model comparison. The time per epoch corresponds to the second epoch (time after loading data) for 4 GPUs parallelization. The unet_2l_32f is the architecture we use for the results.

the learning curves for U-Net architectures we selected (for all results see Appendix A). We call "number of stages" the numbers of encoding/decoding steps. One stage thus corresponds, for example, to one encoding step, then one decoding step, and so on. The U-Net with 0 stages (blue line) corresponds to the simplest architecture without encoding/decoding steps.

We focused on the architecture with 32 filters on all convolutional layers that was a good compromise between the number of parameters, computation time and performance. Figure 6 shows the improvements when we increase the number of stages, until the limit of two stages (green curve). Beyond this, the architecture with 3 stages starts to overfit, meaning that the learning curve (solid line) diverges from the validation curve (dashed line). Another possible choice was the architecture with 5 stages and 8 filters everywhere, that considerably reduced the number of parameters and improved results, however, we had problems during the global generalization (see Appendix B). Thus, the U-Net architecture with 2 stages and 32 filters is a good compromise between performance, time per epochs, and the number of parameters. We also could further increase the complexity of the model and introduce some regularization (as we did in the preliminary study on GitHub); however, this was computationally more expensive for a non-significant improvement in the loss function, so we decided not to use it in this study.

3 Results

For all results, we call the U-Net algorithm, the architecture with 2 stages and 32 filters (unet_2l_32f of Table 2) trained from 1979 to 2008 with the 40 first members. We use the weights from the epoch 300.

3.1 Prediction on the first zone: South Atlantic

Figure 7 shows one snapshot example of the U-Net algorithm prediction of zone1 for member 50 (validation data set), trained in this zone. The forced predicted signal (d) is rather well reconstructed. The error map (c) shows slightly higher errors in the South, which is consistent with the fact that this area in more chaotic than the North (see the ratio of forced to total variance on Figure 2.a in the South Atlantic).

3.2 Global predictions

3.2.1 Zone 1

We then directly applied the U-Net that learned only from zone1 data to the global map (Figure 8). Surprisingly, this first algorithm trained only in the first zone performed quite well. However, the error

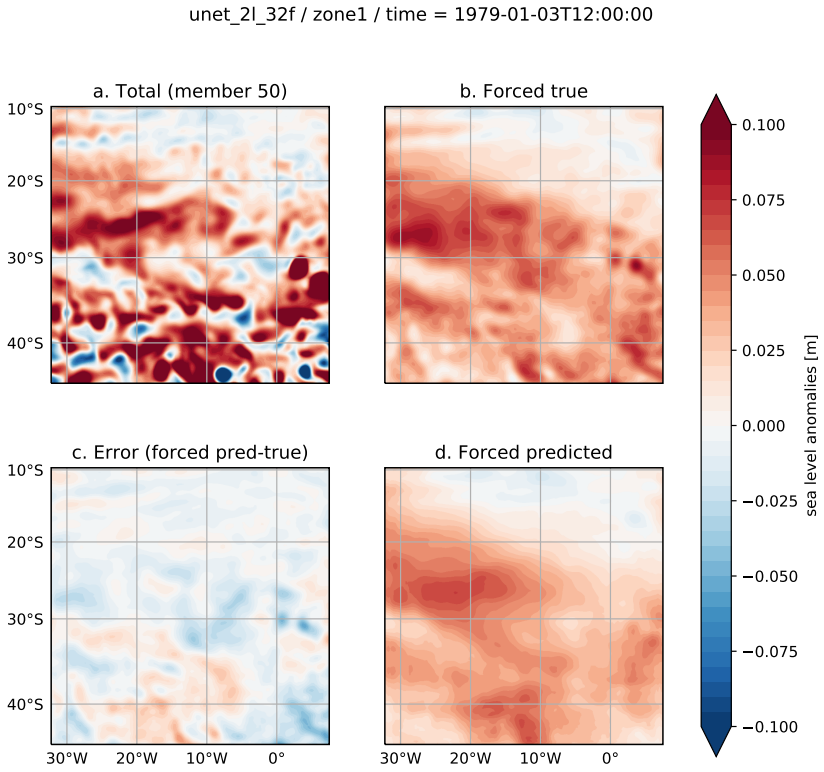


Figure 7: SLA prediction from the U-Net algorithm for one snapshot (January 3, 1979). (a) Total signal from the validation data set (member 50). (b) True forced signal (ensemble mean). (c) Absolute error between the predicted and true forced signal. (d) Predicted forced signal (from the unet_2l_32f trained on zone1).

map shows some differences, especially in different oceanic regimes (for example, in the Pacific equatorial area, Antarctic Circulation Current (ACC) or around the coasts).

Figure 9 shows the metrics used to evaluate the predictions over member 50[8] during the period 1979-2008. The RMSE confirms the intuition from Figure 8 showing the strongest errors around similar zones. The correlation map shows similar patterns. The standard deviation ratios show that the U-Net algorithm prediction generally yields weaker than observed variance, except around Antarctica, where the ratio is close to 100 %. The Figure 9 also reveals the best performance inside the training zone, as expected.

[8]We could also have used other members; however, the computation was already quite demanding. We checked some other members to increase our confidence in the robustness of these results. Generally, they gave the same global mean at 10^{-3} precision. It is not excluded that some members would nevertheless give slightly different results in some untested cases.

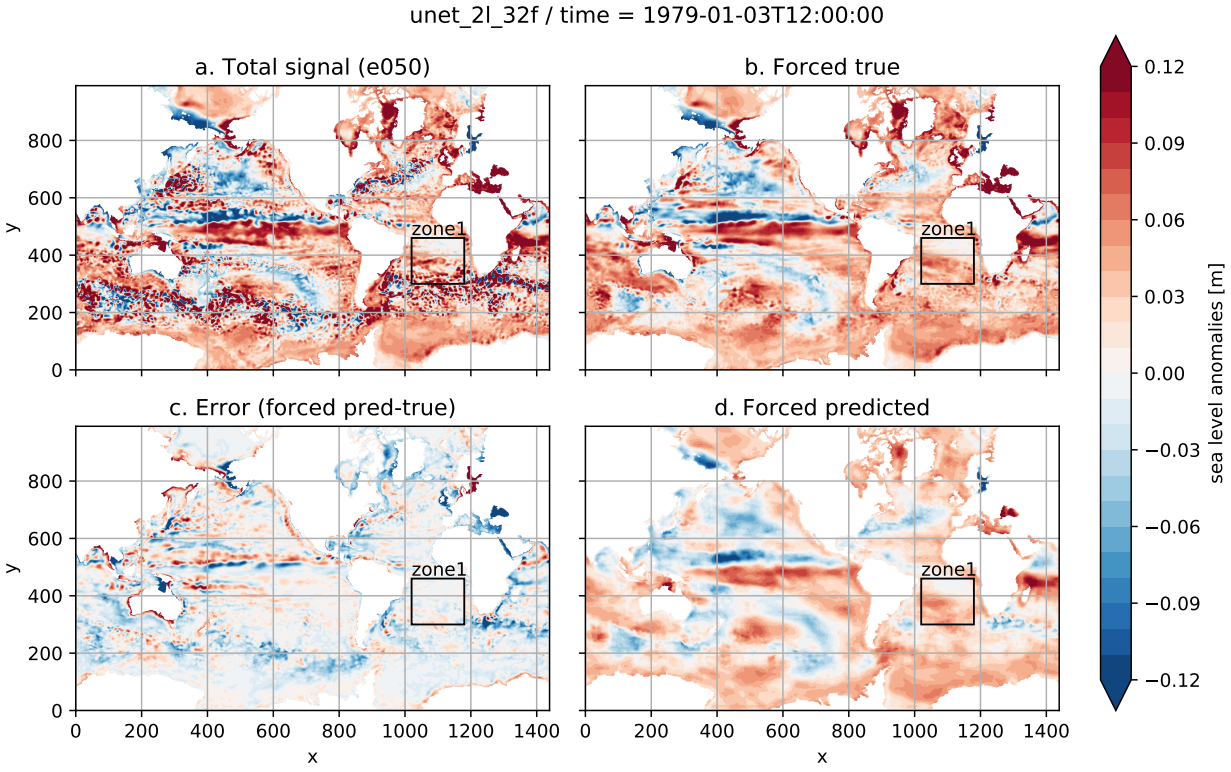


Figure 8: SLA global prediction from the U-Net algorithm trained on zone1 for one snapshot (January 3, 1979). (a) Total signal from the validation data set (member 50). (b) True forced signal (ensemble mean). (c) Absolute error between the predicted and true forced signal. (d) Predicted forced signal.

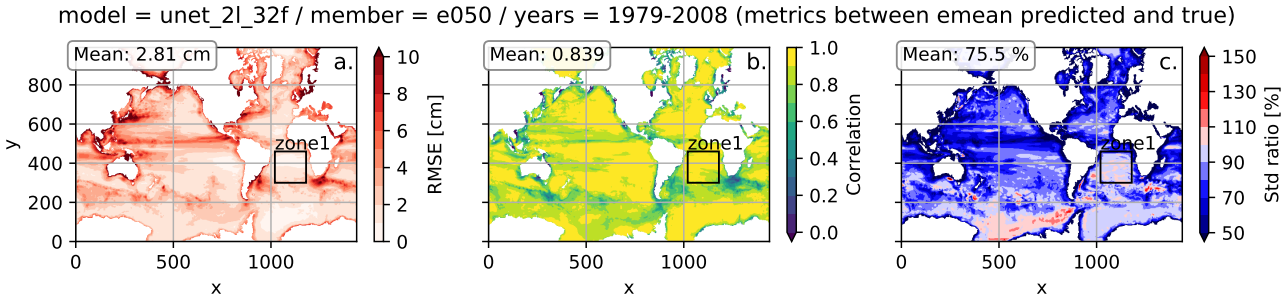


Figure 9: Global metrics computed between the SLA forced signal predicted and true (ensemble mean) with the U-Net algorithm trained on zone1 during the period 1979-2008 on the member 50: (a) root mean square error, (b) temporal Pearson correlation coefficient and (c) ratio of standard deviations ($\frac{\sigma(pred)}{\sigma(true)}$). Mean values are computed from the image without taking into account the latitude.

3.2.2 Zones 1+2

In order to improve the results, we added the "zone2", around Japan, to the training data set. It includes some highly intrinsic multiscale dynamics, such as the Kuroshio current, and some land areas that could decrease coastal errors[9]. Figure 10 shows an improvement for all metrics (first row) and the differences between the training on zones 1+2 and zone 1 alone (second row). The RMSE is decreased by 0.8 cm and displays large improvements between the north of Australia and Japan, including the equatorial dynamics of the Pacific. The coastal errors are also reduced in almost all basins, even in the Mediterranean Sea and North Sea where huge errors (larger than 10 cm of RMSE) were present. It improves the results in other chaotic areas, such as the Zapiola anticyclone (around 45°S, 45°W) where complex dynamics exist (Miranda et al., 1999) and in the Agulhas retroflection, south of Africa. A slight degradation can be noticed in the first zone, which is consistent with the fact the algorithm learns from both zones, and thus, "makes a compromise". The correlation map still has similar patterns and its global mean reaches almost 0.9. The standard deviation ratios are improved almost everywhere, despite some overestimation in the ACC and the Agulhas retroflection, where the correlation is also quite low.

Whilst adding the second zone improved most of the metrics, it added some "island artifacts". Figure 11 shows an example of some spurious white areas (SLA close to 0 m) predicted by the U-Net algorithm. These areas are randomly predicted in different places, depending on the snapshots and members. This problem could have been anticipated because, as described in the method, we replaced the land by zeros, and the algorithm thus learns lands and islands. This problem will be discussed in Section 4. No easy solution was found to avoid these artifacts while keeping similar performance. Thus, we decided to keep zone2 for this study. However, it needs to be kept in mind that this is an unsolved problem.

3.2.3 Zones 1+2+3

After the significant improvement of adding the second zone, we tried to add a third zone in the ACC. Figure 12 shows the results and improvements from the training on zones 1+2+3. Some slight improvements can be seen in the difference maps, in particular in the ACC, where the variance is better reconstructed. However, the RMSE is now slightly degraded around coastal areas. Furthermore, given the computation cost of adding other zones and the nonsignificant global improvements in the global metrics, there appears to be little value to be gained by including further zones.

[9] In practice, we tried some other zones before choosing this second zone. However, it is the one that most improved the global metrics.

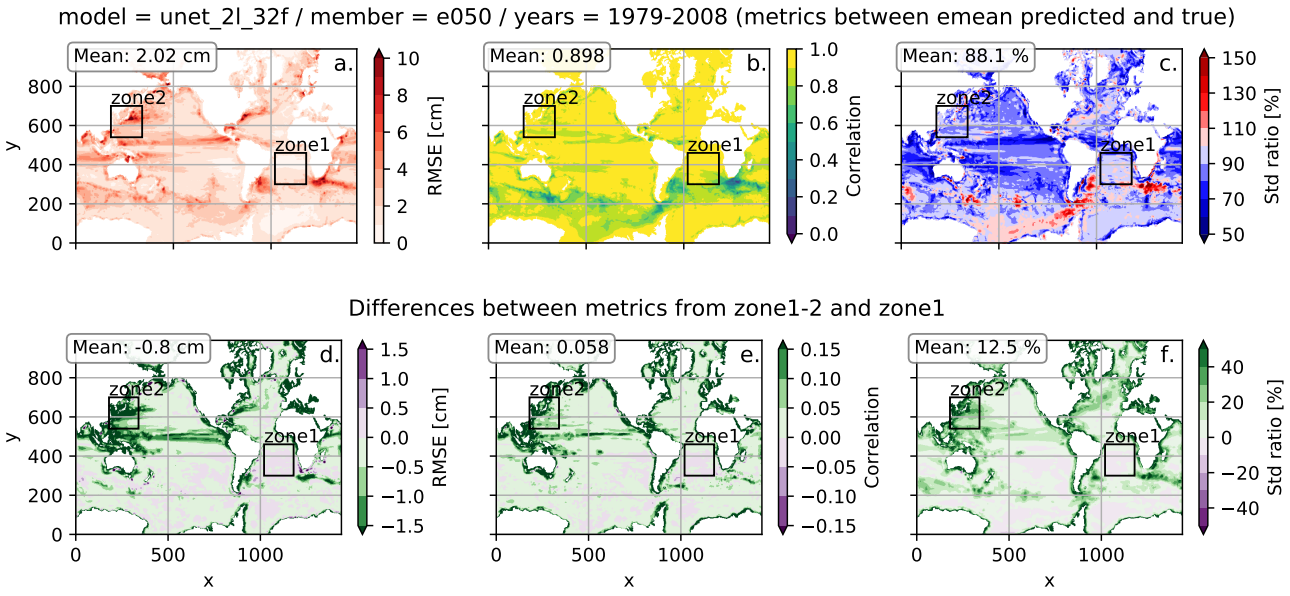


Figure 10: (a, b, c) Same as Figure 9 with the U-Net algorithm trained on zones 1+2 and (d, e, f) global metrics differences between training on zones 1+2 and zone1.

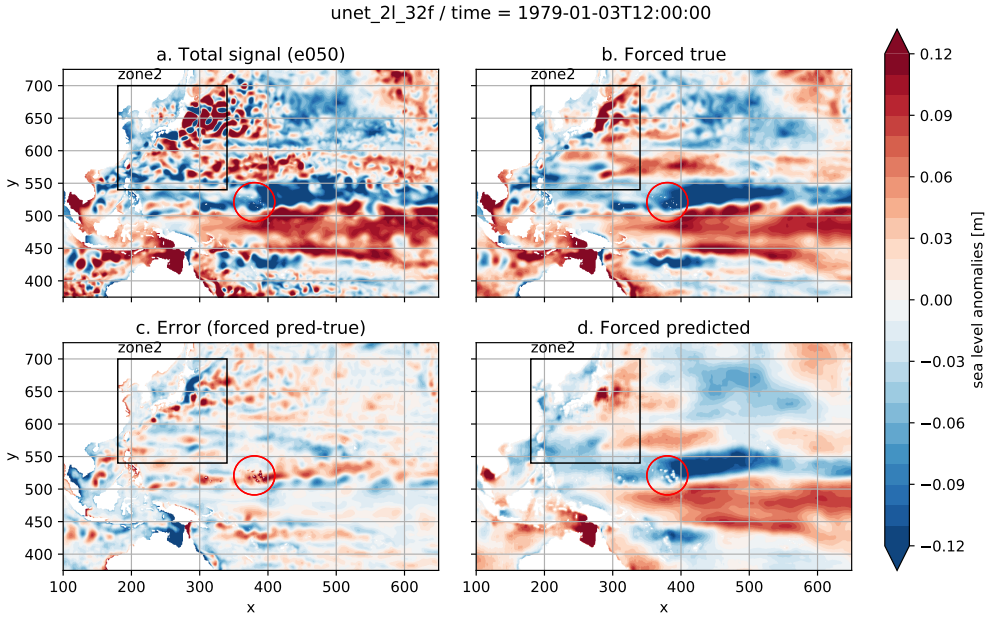


Figure 11: "Island artifact" (red circle) on SLA global prediction zoomed from the U-Net algorithm trained on zones 1+2 for one snapshot (January 3, 1979). (a) Total signal from the validation data set (member 50). (b) True forced signal (ensemble mean). (c) Absolute error between the predicted and true forced signal. (d) Predicted forced signal.

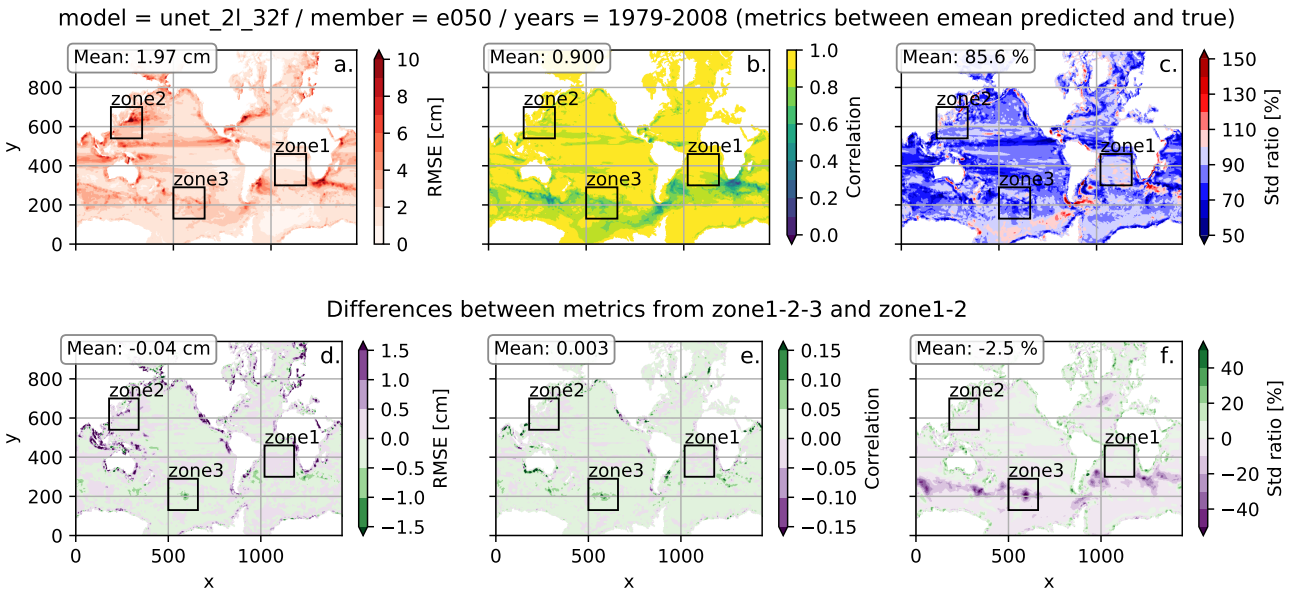


Figure 12: (a, b, c) Same as Figure 9 with the U-Net algorithm trained on zones 1+2+3 and (d, e, f) global metrics differences between training on zones 1+2+3 and zones 1+2.

3.3 Comparison to the bandpass Kaiser-Bessel filter

3.3.1 Global metrics

We compare the results obtained from the U-Net trained on the three zones with those obtained by application of the bandpass Kaiser-Bessel filter (KB filter) used in Close et al. (2019). Figure 13 shows (first row) the global metrics of the KB filter and (second row) the differences from the U-Net prediction (U-Net minus KB filter). The global performances are quite similar to the U-Net algorithm. The forced variance seems generally to be better reconstructed by the KB filter, except in the ACC where a large overestimation can be observed (the ratio between predicted and true emean stds can reach 300 %). Otherwise, the RMSE and correlation difference maps show slightly better global prediction using the U-Net algorithm: generally, the U-Net performs better around chaotic areas like the ACC, Kuroshio or the Gulf Stream (improvements of more than 1 cm in RMSE and more than 0.1 in correlation), while the U-Net performance drops around coastal areas.

3.3.2 Large scale intrinsic variability

The bandpass KB filter used in Close et al. (2019) uses cutoffs of 115 km and 800 km (intrinsic signal attributed inside these boundaries, and forced signal outside). Thus, we considered it interesting to compare, for example, the reconstructed intrinsic signals at large scales. Figure 14 shows an example

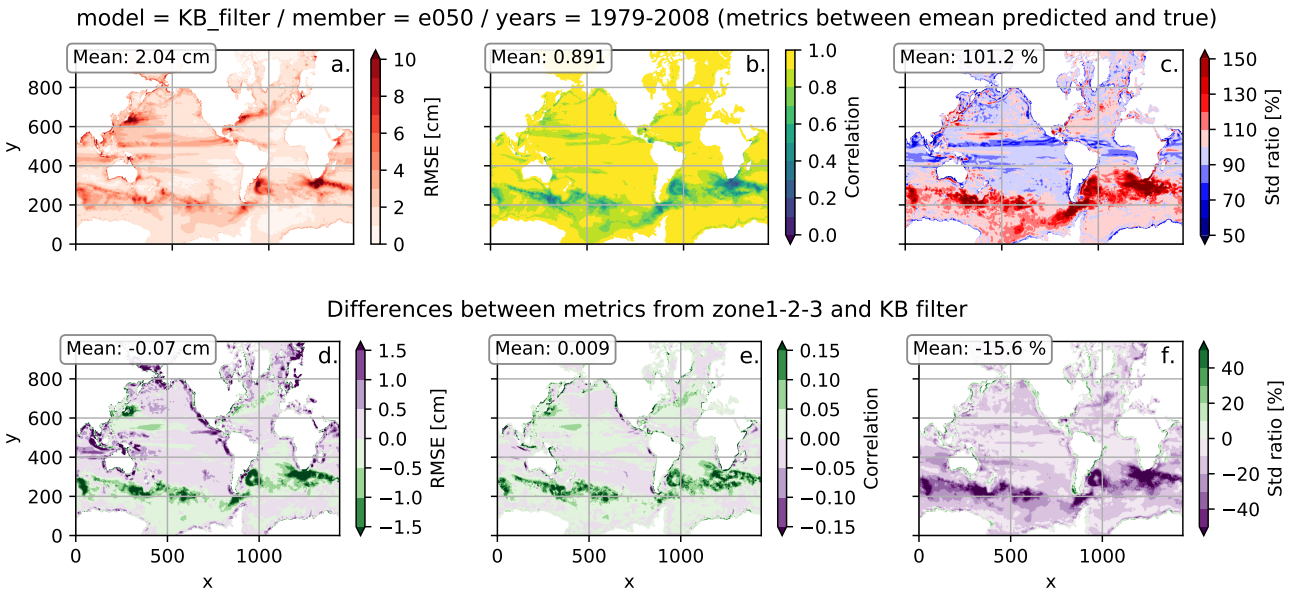


Figure 13: (a, b, c) Same as Figure 9 with the Kaiser-Bessel (KB) filter and (d, e, f) global metrics differences between the KB filter and U-Net algorithm trained on zones 1+2+3.

of the "true" intrinsic signal for one snapshot (first row), computed as the total SLA from one member minus the ensemble mean. The next two rows show the intrinsic signal predicted by the U-Net algorithm and the KB filter (computed as the total signal minus the forced predicted signal). Some signal is still present for the KB filtered-data at scales larger than 800 km that can be explained by the non-perfect filtering capacity of the KB filter; however no signal is present at scales larger than 2000 km as expected. The U-Net does succeed in reconstructing some intrinsic signal at both scales with similar amplitudes. However, the signal seems to diverge in some places, in particular around the equator, for example. The U-Net also predicts some intrinsic variability in the seas surrounding Europe, where no signal is present in the "truth".

Even if the U-Net algorithm succeeds in reconstructing some intrinsic signal at scales larger than 2000 km, Figure 15 shows that the global RMSE is lower for the KB filter (0.90 cm compared to 1.01 cm for the U-Net algorithm). This can probably be explained by the fact that the U-Net std ratio seems to be either too big or too small in areas where the correlation coefficient is quite low, yielding large errors. Nevertheless, the global correlation between the true and reconstructed intrinsic is improved from 0.18 for the KB filter to 0.30 for the U-Net algorithm. The global std ratio confirms that almost no intrinsic signal is reconstructed from the KB filter at large scales, while, the U-Net does reconstruct some of the intrinsic variances, but with some unrealistic patterns.

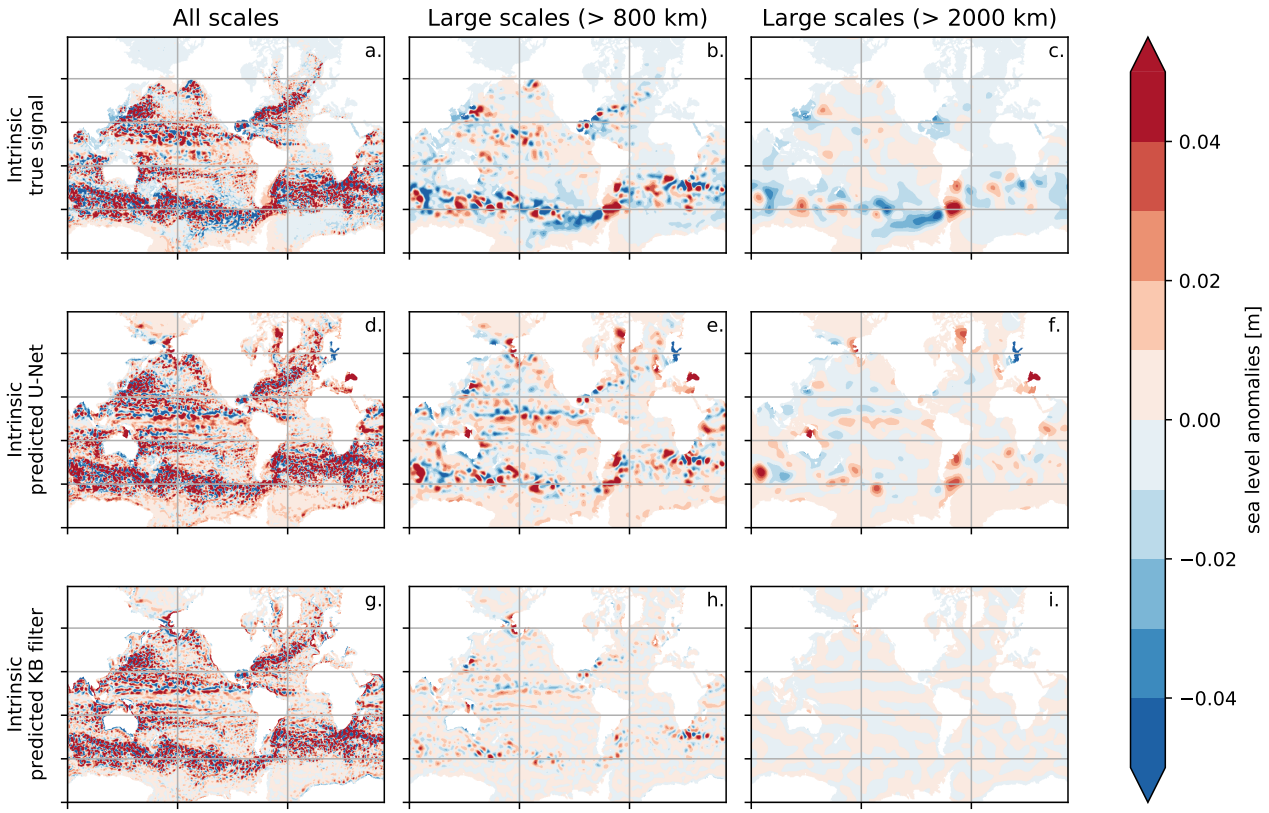


Figure 14: Intrinsic signal (total-forced) of SLA at different scales for one snapshot (January 3, 1979) and member 50: (a,b c) true values, (d, e, f) prediction with the U-Net algorithm, and (g, h, i) prediction with the Kaiser-Bessel (KB) filter. The same KB filter was used for selecting large scales.

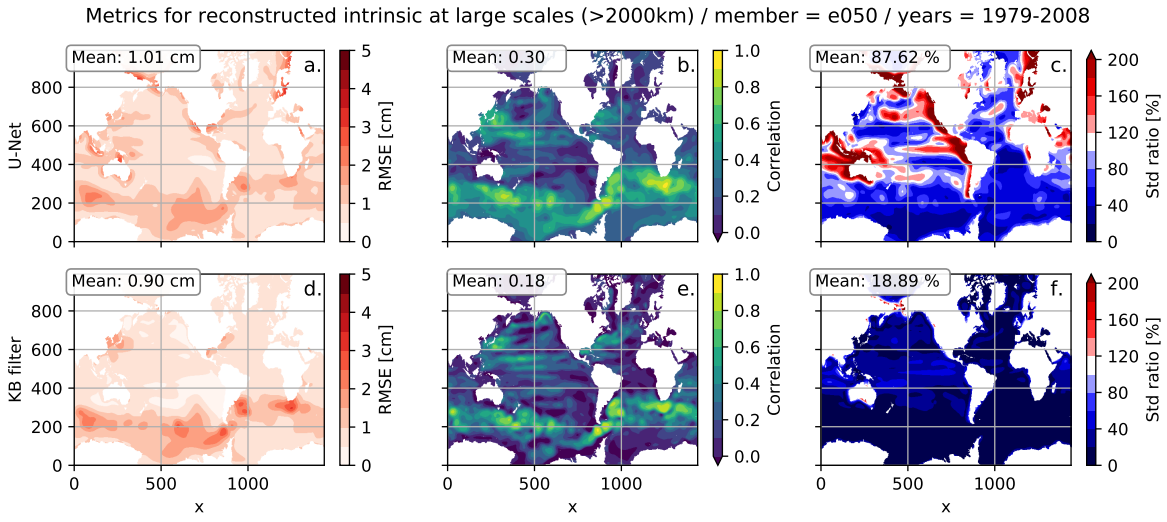


Figure 15: Same as Figure 9 for the large scales (>2000km) intrinsic reconstructed signal for (a, b, c) the U-Net (trained on the zones 1+2+3) prediction and (d, e, f) the KB filter prediction. The prediction is made on all scales and then filtered to large scales.

3.3.3 Spectral analysis

Further investigation of the transfer functions of both methods (U-Net and KB filter) between the total SLA and the forced predicted signal is carried out. One of the main interests of this analysis is to identify whether the U-Net can adapt its transfer function to the different zones, while we expect the bandpass filter to behave the same on all different zones.

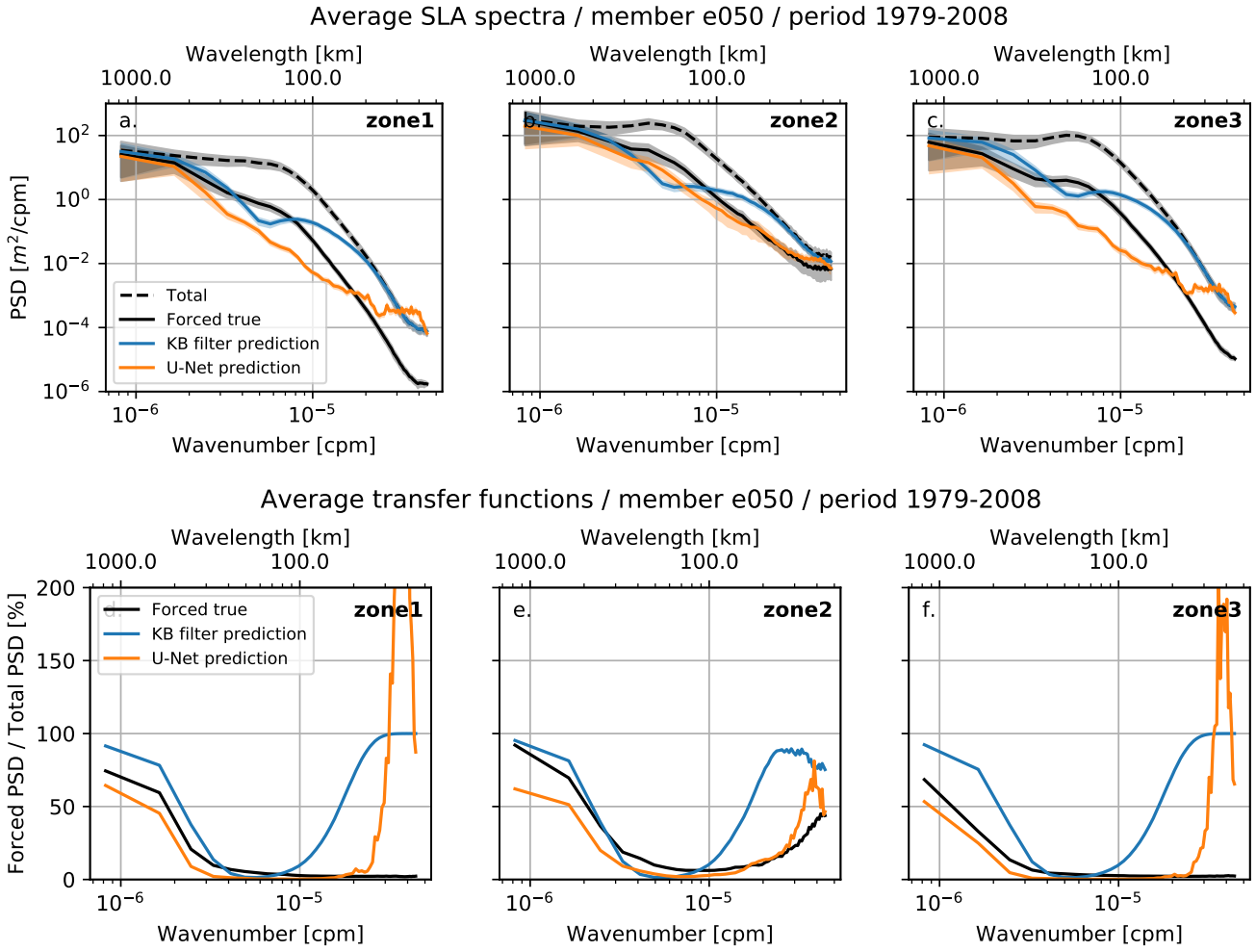


Figure 16: (a, b, c) Averaged estimation of the wavenumber spectrum of two-dimensional SLA of different zones (spectrum for each time step are averaged on the period 1979-2008 on the member 50), the shading envelopes represent plus or minus one standard deviation. Before the spectrum estimate, the data set is interpolated (any NaN value is replaced with interpolated value) and the mean is removed. The 2D spectrum obtained after application of the FFT is radially averaged to a 1D spectra. The PowerSpec tool developed by Adekunle Ajayi and Julien Lesommer is used (`https://github.com/adeajayi-kunle/PowerSpec`). (d, e, f) Transfer functions are estimated as the average forced spectrum divided by the average total spectrum.

Figure 16 (first row) shows the spectral analysis of the total SLA (dashed black line), the ensemble mean (solid black line), the predicted forced signals from the KB filter (blue line), and the U-Net algorithm (orange line). Both predictions seem to more or less follow the true signal. However, the KB filter prediction has generally lower energy than the true signal on spatial scales between 100 km and 800 km, which is consistent with the fact that it attributes these spatial scales range to the intrinsic signal. On the other hand, the U-Net prediction is usually lower than the true signal at almost all scales except at fine scales (smaller than 80 km) where the energy is higher than the true signal. The KB filter has similar behavior at small scales. However, the energy becomes negligible in this range.

The transfer function (ratios of forced-to-total signals, Figure 16 second row) shows complementary results: both methods reconstruct the forced signal quite well at scales greater than 100 km. However, the U-Net algorithm tends to fit better the true signal at fine scales (lower than 100 km), until a strange peak is found. This might be due to edge effects, or may arise from the concatenation step in the U-Net, but these ideas have not been tested. As expected, the KB filter shows a similar transfer function for all three zones, while the U-Net algorithm shows more differences (other zones, especially non-trained zones, were also tested, and the results found to be similar), confirming the ability of the U-Net to adapt its transfer function depending on the location.

3.4 Application to observations

3.4.1 Global prediction

Estimating the forced signal from observed total SLA fields can yield very valuable information, particular with respect to the detection of atmospherically-forced (deterministic) SLA patterns. We thus applied the U-Net algorithm to AVISO observations over the period 1993 to 2017. Knowing that we only worked in the model "world", it is possible that some structures or physical processes are not well resolved in the model but present in altimetric observations.

Since the U-Net algorithm was trained on the original model grid, we interpolated all AVISO data to the ORCA025 grid before prediction. Figure 17 shows one prediction snapshot of the U-Net algorithm from the total observed signal. No metric can be directly computed to evaluate the accuracy of this prediction since the "true" forced signal is unknown in observations.

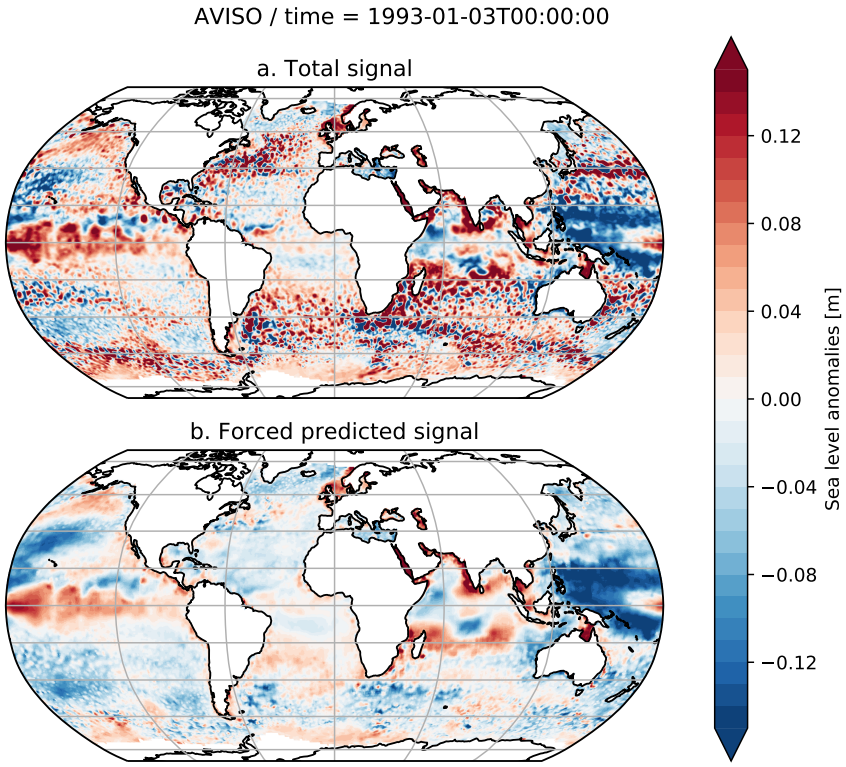


Figure 17: Prediction of the forced SLA from the AVISO observation data on one snapshot (January 3, 1993): (a) total SLA AVISO, (b) forced predicted SLA with the U-Net algorithm. Data were linearly interpolated to the ORCA025 grid before prediction and put back into the AVISO grid.

3.4.2 North Atlantic Oscillation impact on the real ocean

In order to evaluate the U-Net prediction indirectly, the link between the North Atlantic Oscillation (NAO) index and the SLA is further investigated. Indeed, the wind forcing is known to influence the SLA variability with a usual delay of one month. We chose to reproduce the result from Close et al. (2019) to evaluate the ability of the machine learning filter (U-Net) to reproduce a physical phenomena. Figure 18 (first row) represents the temporal Pearson correlation coefficient between the winter averaged (DJF) NAO index and the winter averaged (JFM, delayed of one month) SLA of the total and recreated forced for the KB filter and U-Net. The winter averaged NAO index is computed from monthly values and the SLA from 5-day average fields. The winter season is chosen because the influence of the NAO is the strongest at this time (Close et al., 2019). Figure 18 (second row) is computed as the square of the correlation coefficient, corresponding to the percentage of SLA variance explained by the NAO.

All subfigures display a tripolar pattern (also shown in Marshall et al. (2001) between NAO and SST); however, this pattern is partly hidden for the total SLA. The sparse correlation for the AVISO observation

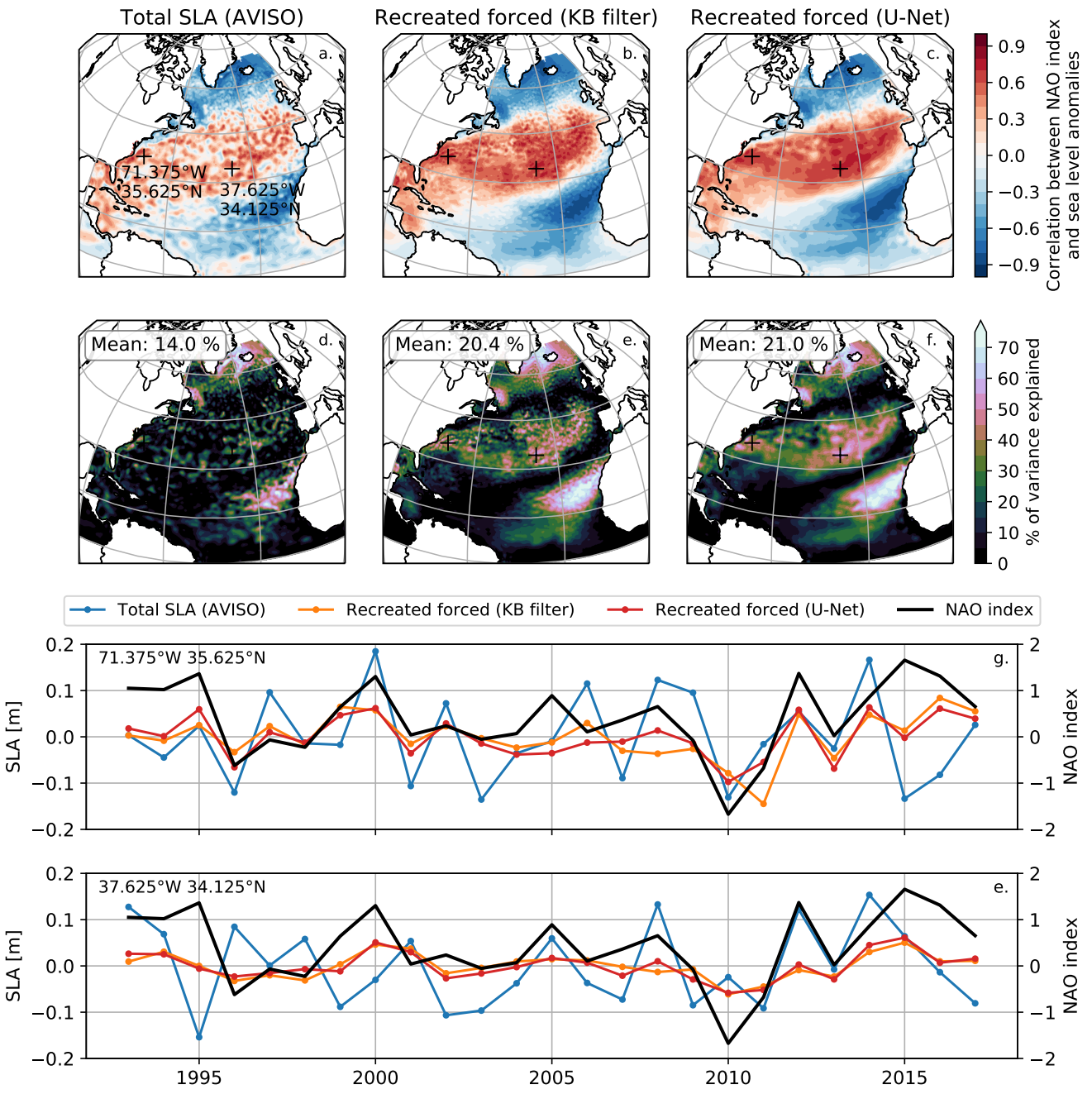


Figure 18: (a, b, c) Correlation between DJF NAO index and JFM SLA from AVISO observations for the period 1993 to 2017: (a) total AVISO, (b) reconstructed forced signal with the KB filter and (c) with the U-Net algorithm. (d, e, f) same as (a, b, c) but for the percentage of SLA variance explained by the NAO (square of correlation); the mean value was computed directly from the image without taking into account the latitude. The black cross represents the two points were the time series (g and e) were extracted. Source of NAO index: `https://www.ncdc.noaa.gov/teleconnections/nao/`.

can be explained by the strong chaotic behavior emerging from the ocean itself above all around the Gulf Stream. By filtering this chaotic part, we recover a stronger correlation between the NAO and the SLA. Both methods (KB filter and U-Net) yield similar results, confirming the ability of the U-Net to filter the chaotic signal in observational data. Figure 18 (g, e) shows two time series extracted from the 2 points shown on Figure 18 (a). In these two areas, the forced recreated fields are indeed better correlated with the NAO index.

3.5 Sensitivity to generalization

To evaluate the ability of the U-Net algorithm to generalize to other data or periods, we used member 1 (training data set) and member 50 (validation data set) on the period from 2002 to 2008 where the U-Net has learned, and the period from 2009 to 2015 where the algorithm has not seen any data. Both periods include seven years for comparison.

Table 3 shows the global image average metrics. There are no significant differences for any metrics and both periods between members 1 and 50, suggesting that the U-Net algorithm can generalize to other data for global predictions. For the two different periods, the results are similar with the exception of the global correlation, which is slightly lower for the new period (2009-2015). Similar results can be found for other members.

Years	**2002-2008**		**2009-2015**	
Member	**e001**	**e050**	**e001**	**e050**
RMSE [m]	1.95	1.95	1.94	1.95
Correlation	0.908	0.909	0.896	0.897
Ratio of std [%]	86.5	84.5	85.7	85.7

Table 3: Global averaged metrics (average is computed on the image ORCA025 grid without taking into account the latitude and lands) of the U-Net SLA predictions for the member 1 (training data set) and 50 (validation data set) on two different periods: 2002-2008 (training period) and 2009-2015 (new period).

3.6 Data reduction

Another question raised by our study is: how much data do we need to feed the U-Net algorithm? Knowing that we almost reached the computation limits with the three zones, it is a doubly relevant question. Table 4 shows a preliminary result of different possible methods for reducing the training data

set. The results do not show significant differences between using the full data set or taking only half of it, encouraging the use of the reduced data set for further study.

	Full	**Period/2**	**Time steps/2**	**Members/2**
RMSE [cm]	1.97	1.95	1.97	1.98
Correlation	0.900	0.902	0.901	0.901
Ratio of std [%]	85.6	86.4	85.2	86.1

Table 4: Same as Table 3 but for the member 50 on full training period 1979-2008 (30 years). First column summarize the global metrics of Figure 12 corresponding to the training of the U-Net algorithm with the full data set of the three zones. Then the second, third and fourth columns respectively correspond to: training on period 1979-1993 (15 years), training with only one time step per two on the full period and using only the first 20 members.

4 Discussion

4.1 General insights

The U-Net algorithm is able to remove the chaotic signal from modelled and observed total SLA fields. Its performance is quite similar to that of the KB filter used by Close et al. (2019) and does slightly better regarding the global metrics, transfer functions, and large-scale prediction of the intrinsic signal. It is encouraging to get these results with a data-driven approach. However, many things could be improved in the present state of the proposed method.

First, the model choice was made on only the first zone, and other models or architectures with a different number of parameters could perform better on global predictions. A different choice of zones could also improve results. The grid dependence of our method could be a limitation, even if the multiple stages of the U-Net allows the results at different scales to be combined, and thus may mitigate this effect. Trying to include latitudes and longitudes in the algorithm could be a possible solution to generalize the learning to any grid.

We note that we kept the annual cycle during the training. The U-Net algorithm must thus have learned the annual cycle. However, because it learns on some particular zones, a bias might be introduced to the global results. Performing predictions on data without the annual cycle in our case could result in a decrease in performance.

Inclusion of the temporal dimension could also be another way to improve the U-Net algorithm. Indeed, Close et al. (2019) showed that the intrinsic and forced components are easier to filter on spatial scales with a bandpass filter. However, the machine learning algorithm could extract relevant information in the temporal dimension too. It was computationally impossible in our study to add the time dimension with the defined zones. Learning on smaller zones and on a few months on the temporal dimension could be explored. It is possible to use the same architecture with 3D convolutional layers instead of 2D. Nevertheless, it might be necessary to look for better architectures for adding the temporal dimension.

It can be noted that, generally, the U-Net predictions have lower amplitudes (std ratios smaller than 100 %), it might be interesting to try to constrain the algorithm to conserve the variance, for example. Adding another variable to the cost function, like vorticity for instance, could also be a possibility.

4.2 Data reduction

Deciding on the amount of data needed for training a deep learning algorithm is not easy. Table 4 suggests that, in our case, it is not necessary to feed to the algorithm with the full data set. Further investigation could help clarify the best way to reduce the training data set without too much loss on the global metrics. A limit could be found before the general performance drops. Reducing the training data set could allow more zones to be added, larger zones to be used, or the time dimension to be included in the training data set for a cheaper cost. This might be useful for either improving the global predictions or, if no better performance is found, to at least make the training step cheaper in terms of computation cost.

4.3 Island artifacts

The choice to include a zone with some land and islands was innovative; no study in our knowledge tried ConvNets on regions with islands for geophysical data. The straightforward method we used to replace the lands by zero has some drawbacks. The results (e.g. Figure 11) show artefacts in chaotic regions and close to the coasts. It is consistent with the fact that we trained the algorithm including a zone around Japan, where many small islands are present. There are probably two options to reduce or avoid these artifacts: either preprocessing the data to find a better way than adding zeros over lands and/or looking for different learning methods that would not take into account the masked areas. For the first option, it might be wiser to interpolate data instead of just replacing the masked area by zeros. However, this is not a perfect solution since some non-physical information will be still introduced and learned by the algorithm.

We explored some ideas about the second option to utilise different learning methods. However, we did not go far enough to present the results in this study. A first idea is to exclude the masked area in the cost function by passing the land mask to the model, and multiplying by it before computing the loss. However, we did not see significant improvements, and some artifacts were smoothed but still present. Another idea that was explored was to add bathymetry as another variable. Knowing that the depth differentiates coastal and non-coastal dynamics, it was believed that this might improve the performance in coastal regions. However, adding the bathymetry strongly degraded the results. This solution might be re-tried with a different model with more parameters. The last idea that we investigated was to use partial convolutional layers, as developed in Liu et al. (2018)[10]. Their full model did not work in our case, but we tried to replace the first two convolutional layers of the U-Net model by the partial convolutional layers. This method could be the solution; otherwise, the way we applied it did not yield good results either (some smoothed artifacts were still present, and a significant loss in the global metrics occurred).

We cannot conclude in this study which method could be best for including lands in the training of CNNs models without degrading the physical information. However, each of the methods cited before could be a potential solution. Dealing with masked areas in geophysical data remains a challenge for machine learning.

5 Summary and conclusions

The main objective of this study was to estimate the forced variability in global SLA observations (1993-2017) through a deep learning algorithm trained on an existing ensemble ocean simulation.

We used the SSH output of the ensemble experiment OCCIPUT during the period 1979-2015 with a temporal resolution of 5 days and a spatial resolution of about $1/4°$. The SSH was preprocessed by removing the mean for each snapshot and removing the long-term trend. The 50 members were used for computing the ensemble mean, defined here as the atmospherically-forced signal.

A machine learning algorithm, U-Net, was used to extract the intrinsic variability of the ocean and to retrieve the forced variability from a single member of the ensemble. The U-Net algorithm was trained on the 40 first members over the period 1979-2008. Three different zones were used for the training to improve the global predictions of the forced SLA signal. The best global prediction reaches a correlation coefficient of 0.9 with the true signal and reconstructs 85.6 % of its variance.

Compared to the band-pass filtering performed by Close et al. (2019), the U-Net performance is quite

[10]We used the Keras implementation of it: `https://github.com/MathiasGruber/PConv-Keras`

similar. It shows some potential to retrieve some scales not reconstructed by the bandpass KB filter, but with some unrealistic patterns. The U-Net transfer functions are closer to those obtained using the ensemble values, and can adapt to different zones. Furthermore, the results on the NAO showed the potential for the U-Net to be applied to observation data. However, the performances of the U-Net algorithm could drop over a different period.

This study shows the potential of CNN to perform a complex task and go beyond what can be done by hand on geophysical data. However, it also shows some limitation that needs to be investigated in the future. First, our method depends on the grid, restricting its application to different resolutions. Second, it requires significant computation resources, that may limit further development. However, this field of study is quite young, and many solutions for going further, improving the method, and trying to understand how the algorithm works could be explored.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P. A., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zhang, X.: TensorFlow: A system for large-scale machine learning, CoRR, abs/1605.08695, URL `http://arxiv.org/abs/1605.08695`, 2016.

Bernard, B., Madec, G., Penduff, T., Molines, J.-M., Treguier, A.-M., Le Sommer, J., Beckmann, A., Biastoch, A., Böning, C., Dengg, J., Derval, C., Durand, E., Gulev, S., Remy, E., Talandier, C., Theetten, S., Maltrud, M., McClean, J., and De Cuevas, B.: Impact of partial steps and momentum advection schemes in a global ocean circulation model at eddy-permitting resolution, Ocean Dynamics, 56, 543–567, doi:10.1007/s10236-006-0082-1, 2006.

Bessières, L., Leroux, S., Brankart, J.-M., Molines, J.-M., Moine, M.-P., Bouttier, P.-A., Penduff, T., Terray, L., Barnier, B., and Sérazin, G.: Development of a probabilistic ocean modelling system based on NEMO 3.5: application at eddying resolution, Geoscientific Model Development, 10, 1091–1106, doi:10.5194/gmd-10-1091-2017, 2017.

Bolton, T. and Zanna, L.: Applications of Deep Learning to Ocean Data Inference and Subgrid Parameterization, Journal of Advances in Modeling Earth Systems, 11, 376–399, doi:10.1029/2018MS001472, 2019.

Brankart, J.-M., Candille, G., Garnier, F., Calone, C., Melet, A., Bouttier, P.-A., Brasseur, P., and Verron,

J.: A generic approach to explicit simulation of uncertainty in the NEMO ocean model, Geoscientific Model Development, 8, 1285–1297, doi:10.5194/gmd-8-1285-2015, 2015.

Chollet, F.: Keras, URL https://keras.io, 2015.

Church, J., Clark, P., Cazenave, A., Gregory, J., Jevrejeva, S., Levermann, A., Merrifield, M., Milne, G., Nerem, R., Nunn, P., Payne, A., Pfeffer, W., Stammer, D., and Unnikrishnan, A.: Sea Level Change, book section 13, p. 1137–1216, Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA, doi:10.1017/CBO9781107415324.026, 2013.

Cleveland, W. S.: Robust Locally Weighted Regression and Smoothing Scatterplots, Journal of the American Statistical Association, 74, 829–836, URL http://www.jstor.org/stable/2286407, 1979.

Close, S., Penduff, T., Speich, S., and Molines, J.-M.: A means of estimating the intrinsic and atmospherically-forced contributions to sea surface height variability applied to altimetric observations., In revision, Progress in Oceanography, 2019.

Dong, C., Loy, C. C., He, K., and Tang, X.: Image Super-Resolution Using Deep Convolutional Networks, IEEE Transactions on Pattern Analysis and Machine Intelligence, 38, 295–307, doi:10.1109/TPAMI.2015.2439281, 2016.

Dussin, R., Barnier, B., and Brodeau, L.: The making of Drakkar forcing set DFS5, URL https://www.drakkar-ocean.eu/publications/reports, 2016.

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

Jiang, G.-Q., Xu, J., and Wei, J.: A Deep Learning Algorithm of Neural Network for the Parameterization of Typhoon-Ocean Feedback in Typhoon Forecast Models, Geophysical Research Letters, 45, 3706–3716, doi:10.1002/2018GL077004, 2018.

Keller, C. A., Evans, M. J., Kutz, J. N., and Pawson, S.: Machine learning and air quality modeling, in: 2017 IEEE International Conference on Big Data (Big Data), pp. 4570–4576, doi:10.1109/BigData.2017.8258500, 2017.

Kingma, D. P. and Ba, J.: Adam: A method for stochastic optimization, URL https://arxiv.org/abs/1412.6980, 2014.

Krasnopolsky, V. M. and Fox-Rabinovitz, M. S.: Complex hybrid models combining deterministic and machine learning components for numerical climate modeling and weather prediction, Neural Networks, 19, 122 – 134, doi:https://doi.org/10.1016/j.neunet.2006.01.002, 2006.

Krizhevsky, A., Sutskever, I., and Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks, in: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, NIPS'12, pp. 1097–1105, Curran Associates Inc., USA, URL http://dl.acm.org/citation.cfm?id=2999134.2999257, 2012.

Leroux, S., Penduff, T., Bessières, L., Molines, J.-M., Brankart, J.-M., Sérazin, G., Barnier, B., and Terray, L.: Intrinsic and Atmospherically Forced Variability of the AMOC: Insights from a Large-Ensemble Ocean Hindcast, Journal of Climate, 31, 1183–1203, doi:10.1175/JCLI-D-17-0168.1, 2018.

Liu, G., Reda, F. A., Shih, K. J., Wang, T., Tao, A., and Catanzaro, B.: Image Inpainting for Irregular Holes Using Partial Convolutions, CoRR, abs/1804.07723, URL http://arxiv.org/abs/1804.07723, 2018.

Llovel, W., Penduff, T., Meyssignac, B., Molines, J.-M., Terray, L., Bessières, L., and Barnier, B.: Contributions of Atmospheric Forcing and Chaotic Ocean Variability to Regional Sea Level Trends Over 1993–2015, Geophysical Research Letters, 45, 13,405–13,413, doi:10.1029/2018GL080838, 2018.

Madec, G.: NEMO ocean engine, Note du pôle de modélisation, 27, 2012.

Marshall, J., Johnson, H., and Goodman, J.: A Study of the Interaction of the North Atlantic Oscillation with Ocean Circulation, Journal of Climate, 14, 1399–1421, doi:10.1175/1520-0442(2001)014<1399:ASOTIO>2.0.CO;2, 2001.

Miranda, A. P., Barnier, B., and Dewar, W. K.: On the dynamics of the Zapiola Anticyclone, Journal of Geophysical Research: Oceans, 104, 21 137–21 149, doi:10.1029/1999JC900042, 1999.

Nair, V. and Hinton, G. E.: Rectified Linear Units Improve Restricted Boltzmann Machines, in: Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML'10, pp. 807–814, Omnipress, USA, URL http://dl.acm.org/citation.cfm?id=3104322.3104425, 2010.

Penduff, T., Juza, M., Barnier, B., Zika, J., Dewar, W. K., Treguier, A.-M., Molines, J.-M., and Audiffren, N.: Sea Level Expression of Intrinsic and Forced Ocean Variabilities at Interannual Time Scales, Journal of Climate, 24, 5652–5670, doi:10.1175/JCLI-D-11-00077.1, 2011.

Penduff, T., Barnier, B., Terray, L., Bessières, L., Sérazin, G., Grégorio, S., Brankart, J., Moine, M., Molines, J., and Brasseur, P.: Ensembles of eddying ocean simulations for climate, CLIVAR Exchange, Special Issue on High Resolution Ocean Climate Modelling, 2014.

Penduff, T. P., Sérazin, G. S., Leroux, S., Close, S., Molines, J.-M., Barnier, B., Bessières, L., Terray, L.,

and Maze, G.: Chaotic Variability of Ocean Heat Content: Climate-Relevant Features and Observational Implications, Oceanography, 2018.

Ronneberger, O., Fischer, P., and Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation, CoRR, abs/1505.04597, URL http://arxiv.org/abs/1505.04597, 2015.

Simonyan, K. and Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition, CoRR, abs/1409.1556, URL http://arxiv.org/abs/1409.1556, 2014.

A Full models comparison

Table 5 shows the results for all different models we tried in this study. The code is available in the Appendix C. The better performances on global predictions motivate the choice of the unet_2l_32f in this study. However, the validation RMSE shown here is only computed on zone1 and does not represent a good estimator for the global predictions. Indeed, many models with lower parameters perform better on zone1. However, all the models that we tried with fewer parameters than the unet_2l_32f tend to drop their performances on global predictions (see Appendix B for an extreme example).

A better validation estimator during the training would be needed. However, it is not possible to give to Keras different image sizes for training and validation. Giving for the validation data set random subsampled areas from the global map could be an idea.

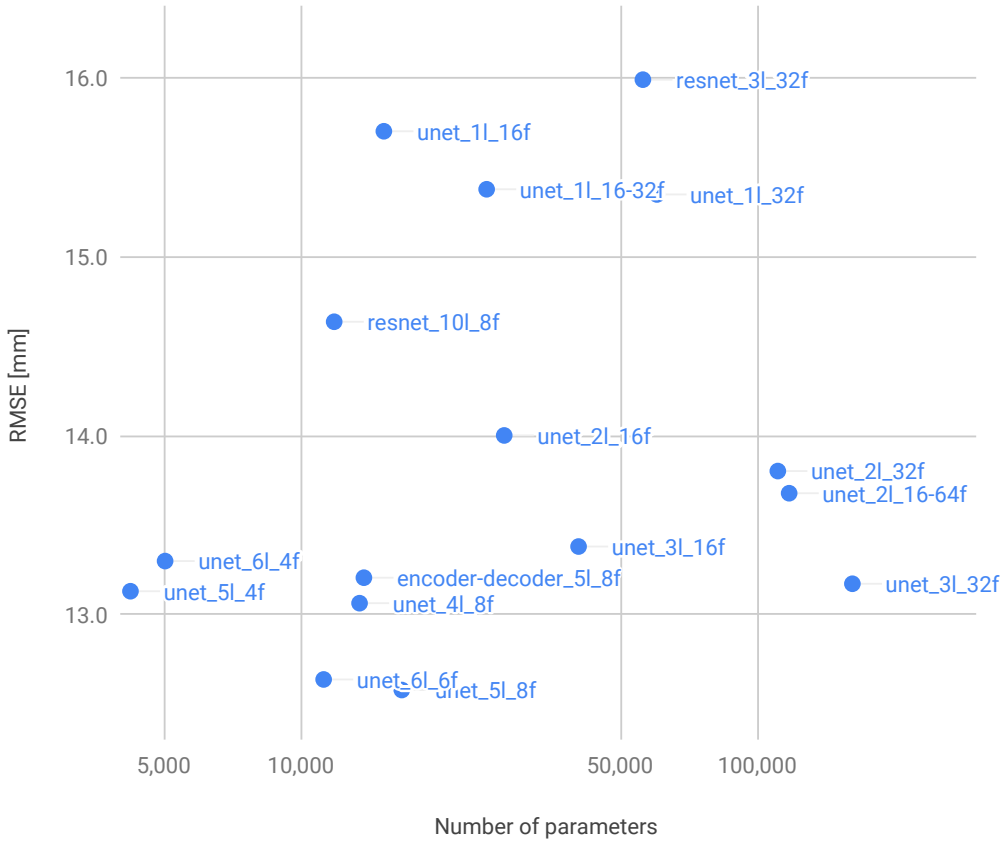


Figure 19: Error (RMSE at epoch 100) versus the logarithm of the number of parameters (unet_0l_32f out of range). Same data as Table 5.

Model name	**Number of parameters**	**Time per epoch [s]**	**RMSE [mm]**
unet_0l_32f	9 601	38	19.041
unet_1l_16f	15 121	66	15.703
unet_1l_16-32f	25 393	76	15.378
unet_1l_32f	59 937	110	15.349
unet_2l_16f	27 745	75	14.002
unet_2l_32f	110 273	130	13.802
unet_2l_16-64f	116 753	100	13.678
unet_3l_16f	40 369	86	13.380
unet_3l_32f	160 609	138	13.172
unet_4l_8f	13 377	65	13.063
unet_5l_4f	4 213	59	13.130
unet_5l_8f	16 553	69	12.578
encoder-decoder_5l_8f	13 673	67	13.206
unet_6l_4f	5 017	61	13.299
unet_6l_6f	11 161	69	12.637
resnet_3l_32f	55 841	180	15.991
resnet_10l_8f	11 769	128	14.637

Table 5: Full model comparison. The time per epoch corresponds to the second epoch (time after loading data) for only 1 GPU. The RMSE is shown for the epoch 100. The training is performed on zone1 for the members 1 to 40 on the period 1979-2008 as for the subsection 2.5. The RMSE corresponds to the validation data set (members 41 to 50 on the same period 1979-2008).

B Translation invariance

As we explained in the report, we did not choose the model unet_5l_8f (or other models with fewer parameters) because they do not perform well on the global maps and they also create different results depending on the way the image is scanned. Indeed, the CNN are supposed to be translation invariant; however, the max pooling operation can introduce different results. For models with few stages, there is negligible impact; however, for the models with more stages of downscaling and upsampling, the results can be significantly different. Figure 20 shows two different predictions using two different subsampled areas, including zone1. The prediction is different depending on whether the prediction starts (top left corner of the image) from South America or directly from zone1.

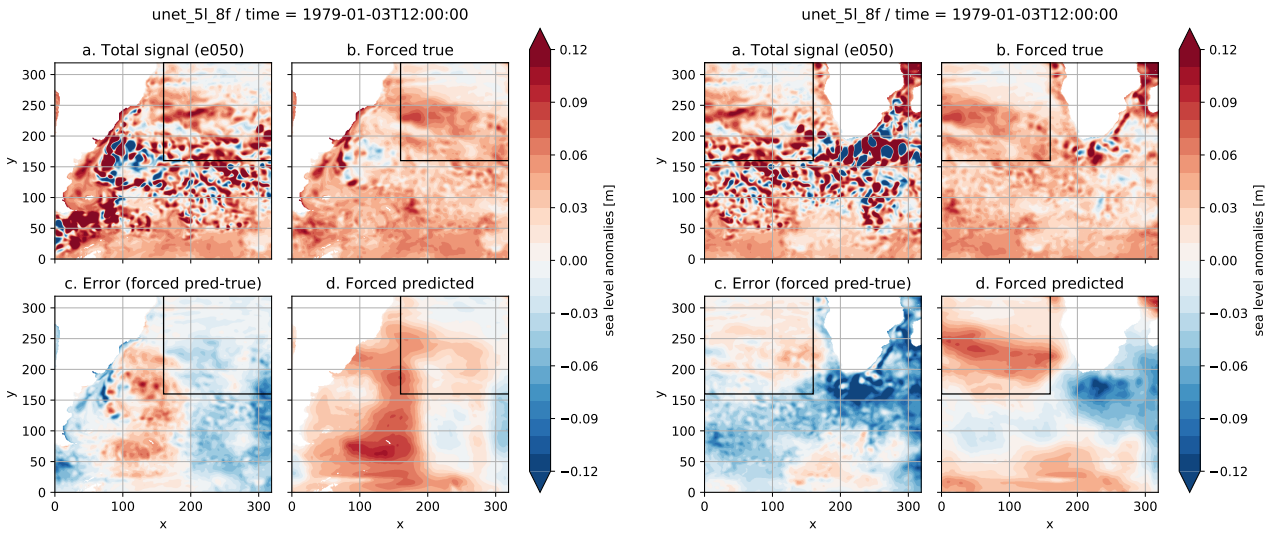


Figure 20: Same as Figure 7 but the prediction is done with the unet_5l_8f. The left and right panels are given as is to the algorithm for the prediction (only the subsampled area shown). The black box correspond to the zone1.

The max-pooling reduces the image size without overlapping pixels, thus, depending on which maximum pixel is picked, it can lose information. As Geoffrey Everest Hinton (an English Canadian cognitive psychologist and computer scientist, most noted for his work on artificial neural networks, nowadays working for Google Brain and the University of Toronto) said[11]:

> *The pooling operation used in convolutional neural networks is a big mistake and the fact that it works so well is a disaster. If the pools do not overlap, pooling loses valuable information about where things are. We need this information to detect precise relationships between the parts of an object. Its true that if the pools overlap enough, the positions of features will be*

[11] `https://www.reddit.com/r/MachineLearning/comments/2lmo0l/ama_geoffrey_hinton/`

accurately preserved by "coarse coding" (see my paper on "distributed representations" in 1986 for an explanation of this effect).

A solution might be to instead use the max pooling operation to reduce the image size, using a stride of 2 for the convolutional layers. Alternatively, an average pooling (instead of taking the absolute maximum value out of 4 pixels, taking the average) might keep more information and be more invariant by translation. The U-Net is quite a complicated machine learning algorithm, and, for an oceanographic application, many questions remain unsolved.

C Models source code

```
import numpy as np
from keras.models import Model
from keras.layers import Input, Conv2D, MaxPooling2D, concatenate, Add
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
from keras.optimizers import Adam
from keras.utils import multi_gpu_model
from keras.layers import Lambda
import tensorflow as tf

def UpSampling2DBilinear(size):
    return Lambda(
            lambda x: tf.image.resize_bilinear(x, size, align_corners=True)
            )

# Model names exemple: unet_3l_64-512f_r
# - unet: model name
# - 3l: 3 layers of pool/up
# - 64-512f: 64 to 512 features/filters
def unet(input_shape, n_layers=2, n_filters=32, n_filters_const=True):
    inputs = Input(input_shape)
    conv   = [None] * (2*n_layers+1)
    pool   = [None] * (n_layers)
    up     = [None] * (n_layers)
```

```
merge  = [None] * (n_layers)
if n_filters_const:
    n = [n_filters] * (2*n_layers+1)
    str_filters = str(n_filters)
else:
    n = np.concatenate((n_filters*2**np.arange(n_layers+1),
                        n_filters*2**np.arange(n_layers-1,-1,-1)))
    str_filters = str(n_filters) + '-' + str(n_filters*2**(n_layers))
# Down
for l in range(n_layers+1):
    if l == 0:
        conv[l] = Conv2D(
                n[l], 3, activation='relu', padding='same',
                kernel_initializer='he_normal')(inputs)
    else:
        conv[l] = Conv2D(n[l], 3, activation='relu', padding='same',
            kernel_initializer='he_normal')(pool[l-1])
    conv[l] = Conv2D(n[l], 3, activation='relu', padding='same',
        kernel_initializer='he_normal')(conv[l])
    if l != n_layers:
        pool[l] = MaxPooling2D(pool_size=(2, 2))(conv[l])
# Up
for i, l in enumerate(np.arange(n_layers+1,2*n_layers+1)):
    upsize_y = int(input_shape[0]/2**(n_layers-i-1))
    upsize_x = int(input_shape[1]/2**(n_layers-i-1))
    up[i] = Conv2D(n[l], 2, activation='relu', padding='same',
      kernel_initializer='he_normal')(
            UpSampling2DBilinear((upsize_y,upsize_x))(conv[l-1])
            )
    merge[i] = concatenate([conv[2*n_layers-l],up[i]], axis=3)
    conv[l] = Conv2D(n[l], 3, activation='relu', padding='same',
        kernel_initializer='he_normal')(merge[i])
    conv[l] = Conv2D(n[l], 3, activation='relu', padding='same',
```

```
            kernel_initializer='he_normal')(conv[l])
    outputs = Conv2D(1, 1)(conv[2*n_layers])
    model = Model(inputs, outputs)
    model.name = 'unet_'+str(n_layers)+'l_'+str_filters+'f'
    return model

def resnet(input_shape, n_layers=2, n_filters=32):
    inputs = Input(input_shape)
    conv0 = Conv2D(n_filters, 3, activation='relu', padding='same',
                   kernel_initializer='he_normal')(inputs)
    conv  = [None] * (2*n_layers+2)
    add   = [None] * (2*n_layers+1)
    for l in range(n_layers):
        if l == 0:
            conv[l] = Conv2D(n_filters, 3, activation='relu', padding='same',
                kernel_initializer='he_normal')(conv0)
        else:
            conv[l] = Conv2D(n_filters, 3, activation='relu', padding='same',
                kernel_initializer='he_normal')(add[l-1])
        conv[l] = Conv2D(n_filters, 3, activation='relu', padding='same',
            kernel_initializer='he_normal')(conv[l])
        if l == 0:
            add[l] = Add()([conv[l], conv0])
        else:
            add[l] = Add()([conv[l], add[l-1]])
    outputs = Conv2D(1, 1)(add[l])
    model = Model(inputs, outputs)
    model.name = 'resnet_'+str(n_layers)+'l_'+str(n_filters)+'f'
    return model

def run_model(model, x_train, y_train, x_val, y_val, lr=1e-4, batch_size=32,
              epochs=1, patience=30, path=None, n_gpus=1, past_history=None):
    if n_gpus > 1:
```

```
        parallel_model = multi_gpu_model(model, gpus=n_gpus)
        batch_size = batch_size*n_gpus
    else:
        parallel_model = model
    # Callbacks
    checkpointer = ModelCheckpoint(
            filepath=path+'best_weights.hdf5', monitor='val_loss', verbose=0,
            save_best_only=True, save_weights_only=True, mode='auto', period=1)
    earlystopper = EarlyStopping(
            monitor='val_loss', min_delta=0, patience=patience, verbose=1,
            mode='auto', baseline=None, restore_best_weights=False)
    reduce_lr = ReduceLROnPlateau(
            monitor='val_loss', min_delta=0, factor=0.5, patience=30,
            min_lr=1e-5)
    # Compile model
    parallel_model.compile(optimizer=Adam(lr=lr), loss='mse', metrics=['mse'])
    # Run model
    history = parallel_model.fit(
            x_train, y_train, batch_size=batch_size, epochs=epochs,
            validation_data=(x_val, y_val),
            callbacks=[checkpointer, earlystopper, reduce_lr])
    return history
```

The encoder/decoder model is the same as the U-Net without the concatenation operation.