Improving Subseasonal-to-Seasonal Weather Forecasting : Leveraging Statistical Post-Processing

Maya Janvier

Mention Sciences des Données de l'Information
Filière Recherche
Double Master Mathématiques, Vision et Apprentissage

May - October 2024

Tuteurs école :
Gilles Faÿ
Vincent Mousseau

Tuteur entreprise :
Claire Monteleoni
David Landry

Table des matières

1 Introduction

Subseasonal-to-Seasonal (S2S) weather forecasts are global or local weather predictions (classically 2meter temperature and total precipitation) beyond 2 weeks but less than 3 months ahead. White et al. (2022) [39] demonstrated the socioeconomic importance of S2S forecasts across many fields, such as agriculture or public health. It can be used to predict monsoons or droughts, allowing stake-holders to anticipate and improve food security and water resources management. S2S forecasts also allow for better planning of the use of renewable energies [39]. Furthermore, these types of forecasts are essential for climate change adaptation, with an increasing number of extreme meteorological events (floods, cold-waves) threatening infrastructures and human health.

S2S predictions are historically done using Numerical Weather Predictions (NWP) models. These models solve the physical equations driving the atmospheric or oceanic processes (e.g. Navier-Stokes for fluid dynamics) one time step at a time. This is done on a discretized version of the Earth sphere, using numerical mathematical methods such as finite elements [40].

In between short to medium-range weather (up to 15 days) and climate predictions (30 years to centuries), S2S weather forecasting appears as a challenging intermediate task. In S2S, predictability stems from both initial conditions as well as slowly evolving coupled components, such as the long-term el Niño Southern Oscillation (ENSO) or the Madden-Jullian oscillation (MJO) [13]. NWP models are medium-range and suffer from low predictive skills at S2S time ranges, that are too far from the initial conditions. On the other hand, predicting several weeks ahead is too close a time range for climate models to take into account coupled processes. In this situation, stakeholders are forced to restrict themselves to medium-range or seasonal predictions [39] because of the S2S predictability gap. Figure 2 by Merryfield et al. [13] shows the different prediction ranges and their sources of predictability.

The particularity of these coupled processes - and consequently S2S weather - is that their own predictability is not permanent and global but rather intermittent and regional. Mariotti et al. [27] defines this spatiotemporal possibility of skillful forecasts beyond two weeks as "windows of opportunity". Depending on the presence or absence of coupled processes, on their linear or non-linear dependencies and their potential interactions, weather forecasting is made possible and Mariotti advises targeting such windows for optimal performance. As an example, forecasts of temperature and precipitation over North America are now solely based on the linear climatological trend of these variables associated with the initial state of MJO and ENSO (Johnson et al. 2014 [20]).

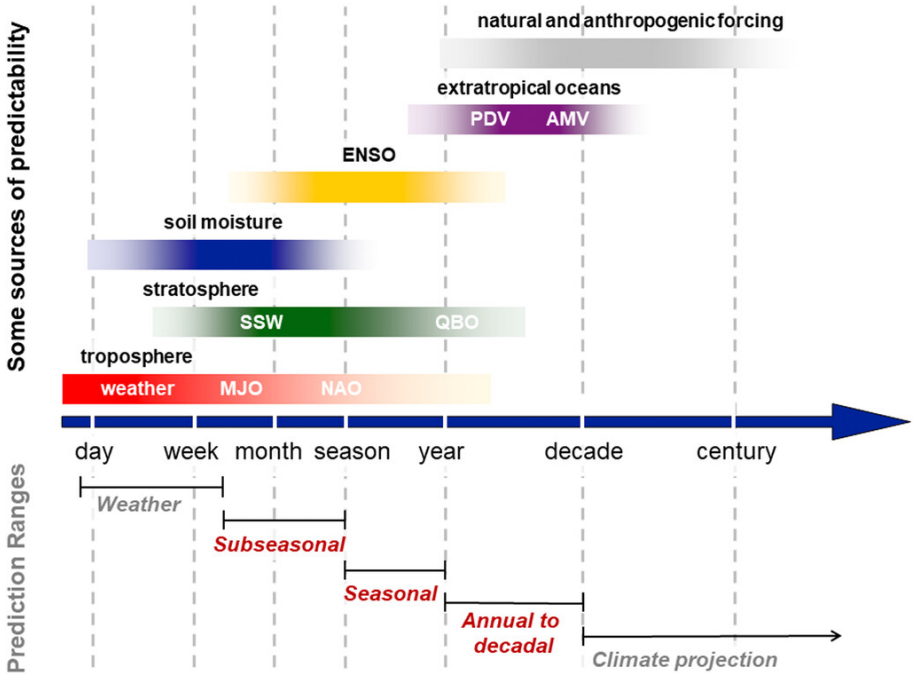


FIGURE 2 – Schematic depiction of (bottom) temporal ranges and (top) sources of predictability for weather and climate prediction. The subseasonal range encompasses the S2S time scales.

However, if NWP are not skillful at S2S time ranges, they can still produce forecasts that are useful. Another strategy to fill the S2S predictability gap is to consider S2S post-processing instead of direct forecasting. The idea is to make use of the numerous existing NWP as prior predictions and to post-process them to correct systematic biases, as it is commonly done for medium-range weather forecasts (Vannitsem et al. 2021 [37]). Figure 3 illustrates 10m wind speed post-processing as an example : such model takes the original forecast from the numerical model, some input features (a mix of predictors including the NWP output itself) and provides a newly corrected forecast.

Most of existing S2S post-processing techniques are statistical and aim at producing a probabilistic forecast from an ensemble of deterministic NWP forecasts. Having distributions is particularly interesting at S2S timescales, since they quantify uncertainties. Some methods choose to model them using a parametric distribution such as the Ensemble Model Output Statistics (EMOS) by Gneiting et al. (2005) [17], while others do not use a particular distribution and model their Cumulative Distribution Function (CDF) directly, for example Quantile Regression (Bremnes 2004 [4]).

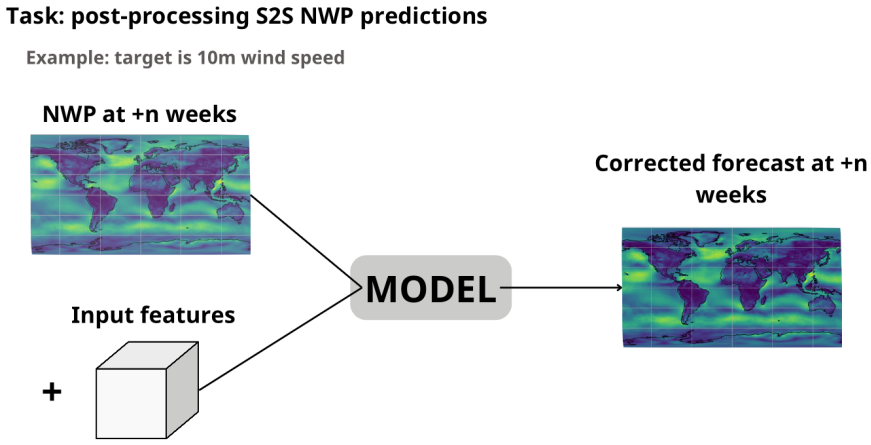


FIGURE 3 – Post-processing task

The newer postprocessing techniques leverage deep learning architectures to produce probabilistic forecasts from simple Neural Networks that work grid-cell wise (Fan et al. 2021 [15]) to spatial architectures based on Convolutional Neural Networks. Such models can focus on categorical S2S predictions (below normal, near normal and above normal conditions, Horat 2023 [18]) or provide explicit distribution of the variables of interest (Pic et al. 2024 [31]), in the spirit of EMOS.

S2S post-processing, supposedly easier than direct forecasting, is still scarcely investigated as highlighted by Horat et al. [18]. In this study, we will explore and compare the different techniques to **post-process weekly aggregated NWP at various lead times : 7, 14, 21, 28, 35 and 39 days.** The recent publication of the **WeatherBench2 dataset** (2023) [33] provides easily accessible NWP data to this end : we will work on the predictions of the **Integrated Forecasting System (IFS) from the European Centre for Medium-Range Weather Forecasts (ECMWF), at 1.5 degrees resolution.**

We will focus on **post-processing 2 meter temperature**, typical in this type of studies ([12] [25] [18] [15]) and **10 meter-wind speed** [1] which is less studied but no less interesting for stake-holders (for instance predicting wind turbine power output). We post-process these variables **globally**, and focus on **distributional-based approaches** (see section 2.2) as they can easily incorporate deep learning. Formally, we first make an **assumption on the prior distribution** $\mathcal{D}_\theta$ of a variable, parametrized by a set of parameters θ that we **derive from NWP predictions**. Then, using a set of input features X and a post-processing model M, we obtain **corrected parameters** $\hat{\theta} = M(\theta, X)$ for the post-processed distribution $\hat{\mathcal{D}}_{\hat{\theta}}$.

In this paper, we will first dive into the existing forecasting and post-processing techniques for NWP, especially for subseasonal-to-seasonal predictions (section 2). We will then present the WeatherBench2 dataset (section 3.1) and its formatting for the S2S post-processing task (section 3.2). Section 4 introduces the post-processing models we compare to the forecasting baselines, from historical statistical methods with EMOS [17] to Deep Learning with a Distributional Regression U-Net [31]. Their respective pre-processing pipelines are described in section 5.1, followed by the performance metrics (5.2) and results in section 5.3. Section 6 explores some possibilities for improvements. Finally, section 7 gives some hindsight on this internship experience at INRIA.

2 Previous and Related Work

Post-processing relies first and foremost on direct forecasts. They can be done in many ways, numerically or using deep learning (section 2.1). Similarly, S2S post-processing techniques are progressively integrating deep learning to their usual pipelines (section 2.2).

2.1 Forecasting

Many entities produce NWPs (section 2.1.1), such as Meteo France or the European Centre for Medium-Range Weather Forecasts (ECMWF). In the past few years, their state-of-the-art predictions have been challenged by hybrid and fully based deep learning models (section 2.1.2), in particular for medium-range weather forecast.

2.1.1 Numerical Weather Predictions

As presented in the introduction, most S2S predictions are produced by NWP models. In such models, the Earth sphere is horizontally discretized into a grid at its surface with latitude/longitude coordinates, and the atmosphere is vertically divided into a number of distinct layers that can be defined in terms of pressure (called **pressure levels**) or altitude. The resolution of the surface grid is described in terms of degrees : a resolution of $1^\circ \times 1^\circ$ degree means each grid cell spans 1 degree of latitude and 1 degree of longitude.

Average values of variables such as wind and temperature are predicted for each grid box by numerically solving equations that describe fluid dynamics, the balance of atmospheric pressure forces on a rotating planet and the redistribution of heat. These computations constitute the **dynamical core** [14]. Small-scale processes such as radiation or turbulence cannot always be resolved by the model and are simulated independently, in so-called **parametrization schemes** [14]. Some variables

that interact in the physical equations can also be fixed (albedo of snow, etc). Figure 4 illustrates NWP components.

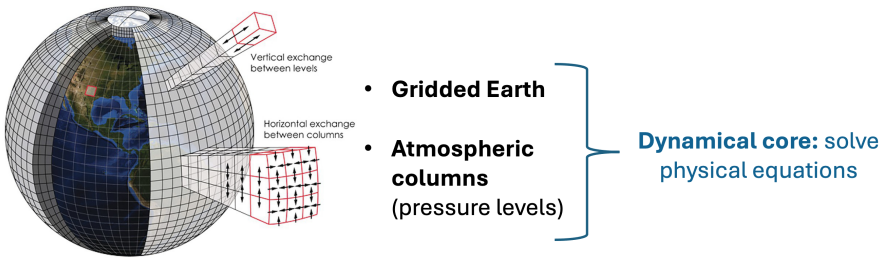


FIGURE 4 – NWP model components

The NWP models thus provide weather predictions based on this type of modeling. Forecasts are usually produced in **ensembles**. An ensemble forecast can be built by running multiple simulations with slight changes in the initial conditions or with slightly perturbed weather models. Each of these simulations is called a **member** of the ensemble. Ensembles aim to reflect a broader range of possible outcomes and introduces uncertainties instead of a single deterministic output [14]. We are interested in quantifying them as NWP models have systematic biases, and the analysis grid from which we get initial conditions as well.

Ensemble models can be taken a step further with **multi-model ensembles** : the idea is to combine forecasts not only from the same ensemble but also from different models. The SubX experiment (2019) assembled seven global models to build a dataset for studying subseasonal predictability as well as post-processing techniques [30]. Building multimodel ensembles is a challenging task as models can be initialized on different dates, which calls for a particular ensemble protocol called **lagged average ensemble** as in [36].

2.1.2 Data driven models

Weather forecasting is a domain that generates a lot of data which makes it well suited for **deep learning** usage. Their applications to weather forecasting have been increasing in the past few years, successfully leveraging deep learning architectures.

In the medium-range context, **fully deep-based** models such as PanguWeather (2023) [2] or GraphCast (2023) [24] outperforms the ECMWF, both in accuracy and running time, which has been the best in this field for years. Both use recent advances in deep learning : Pangu Weather employs a 3D Earth-Specific Transformer based on [26] whereas GraphCast leverages Graph Neural Networks [21] and

transformers.

The **ensembling strategy** still applies to newer machine learning based forecasting systems : Weyn et al. (2021) [38] designed an ensemble of U-Net, thats takes only three minutes on a single GPU to produce a 320-member set of six-week (S2S) forecasts at 1.4 degrees resolution. However this ensemble still performs worse than the ECMWF ensemble at these time scales.

The rise of **generative models** is also of interest in producing ensembles as their outputs are no longer deterministic but probabilistic. They also help tackle the "blurry" aspect of deterministic forecasts that produce the mode of the plausible weather forecast instead of producing a full on member of the distribution. GenCast by Google (2023) generates global 15-day ensemble forecasts at 0.25 degree resolution that, for the first time, are more accurate than the top operational ensemble system, ECMWF's ensemble [32].

But when it comes to predicting ahead of 2 weeks, these models fall short to NWP. **Hybrid models** using both numerical solvers and deep learning seem promising for such ranges. NeuralGCM, released in 2024 by Google, is competitive with the ECMWF in the medium-range and can also accurately track climate metrics such as global mean temperature for multiple decades [22].

2.2 Post-processing

Post-processing weather forecasts is a common technique in many national meteorological services to enhance model performances, as they often lack robustness. Errors in the initial conditions, in the boundary conditions, bad modelling of certain physical processes or inaccuracies in the numerical scheme can quickly degrade the model performances due to its chaotic nature [37], all the more at S2S timescales. Post-processing forecasts helps correcting these random or systematic biases in the models.

2.2.1 Historical statistical techniques

Vannitsem et al. [37] did a very complete review of statistical postprocessing for weather forecasts in 2020. They classified the numerous methods in two main categories : "distribution-based" and "distribution-free". In the annex A, Figure 22 from the review summarizes them and compare their accessibility and flexibility.

Distribution-based approaches specify a parametric model $\mathcal{D}_\theta$ for the forecast distribution depending on the variable of interest [37]. The corrected parameters $\hat{\theta}$ are regressed from NWP predictors by optimising an adapted loss. EMOS, one of

the oldest distribution-based method (2005), is detailed in section 4.

Bremnes et al. (2004) [4] proposed to use Quantile Regression (QR) to post-process precipitation. It is a much older approach (Koenker [23], 1978) where the quantiles are estimated separately instead of estimating the conditional mean of the outcome given predictors, avoiding any distributional assumption.

Taillardat et al. [35] also took the **distribution-free approach** in 2016, using decision trees from random forests for the postprocessing of temperature and wind speed forecast. They showed that it was not only better than the original ensemble forecast, but also more skillful than EMOS.

2.2.2 Deep Learning based post-processing

Many more methods are presented in Vannitsem study, in particular the **integration of deep learning in the former methods**. Bremnes improved QR using Neural Networks to estimate quantile functions' parameters, modelled as Bernstein polynomials (BQN) [5] (2020). Bouallègue et al. [1] (2022) studied different ML methods (random forest, Neural Networks) to post-process 2m temperature and 10m wind speed up to two days ahead, as well as the features' importance for each method.

Landry et al. [25] (2024) compared these historical methods (EMOS, QR) and their deep learning versions for medium-range post-processing, showing their advantages in terms of performances. In the S2S field, Fan et al. (2021) [15] also concluded the Neural Networks being more advantageous, as they are able to model linear and nonlinear relationships, but also make accurate global predictions in a flexible setting.

As for forecasting (section 2.1), **multimodel approaches can be useful for post-processing**. In 2021, the World Meteorological Organization (WMO) organised the S2S Artificial Intelligence challenge [12]. One of its successes was a multimodal approach combining EMOS, Convolutional Neural Networks (CNN) and past observations. Mouatadid et al. (2023) [28] also developed a similar multimodel setup to perform adaptive bias correction using ECMWF forecasts, past and present observations. They also provided a workflow to identify windows of opportunity from specific climate conditions.

Recently, **more complex deep learning architectures** are also leveraged for post-processing medium-range ensemble. Bouallègue et al. (2024) [3] used hierarchical transformers to this purpose and succeeded in improving 2m temperature skill by 20% globally and by 2% for precipitation forecasts from the ECMWF. U-Nets are particularly appreciated in S2S post-processing, for their spatial treatment of

the data : Horat et al. (2023) [18] and Pic et al. [31] (2024) are the more recent examples of its use for temperature and precipitation post-processing.

3 Data

Weather forecasting is a domain that generates a lot of data, from observations to forecasts produced by the numerous numerical and deep learning models built over the years. To take advantage of this, Rasp et al. [33] released WeatherBench2 (WB2) in 2024, a framework for evaluating and comparing data-driven and traditional numerical weather forecasting models. We introduce here the WB2 data we used for our study (section 3.1), as well as its formatting for our experiments (section 3.2).

3.1 WeatherBench2

There are two main components of WeatherBench2 that we need for our post-processing task : outputs of NWP for training data, and reanalysis data for ground truth. In this section, we present these types of data as well as the global structure of WB2 data.

3.1.1 Data types

WeatherBench2 provides outputs of numerical ensemble models such as the **ECMWF Integrated Forecasting System (IFS)**. The IFS is the ECMWF system responsible for producing NWP. It integrates an Earth System model, in particular an ocean and a land-surface models coupled to the atmospheric one, as it is crucial to correctly represent the interactions between these components in order to get accurate weather predictions. Atmospheric chemistry, climate forcings like greenhouse gases and aerosols are all represented within the Earth System model [14].

The ECMWF IFS generates two types of predictions that we can find in WeatherBench2 :

- **Forecasts :** predictions produced by the IFS in the past with the model versions available at the time.
- **Reforecasts (also known as hindcasts) :** more recent versions of the IFS are run for past weather with past initial conditions. It is a costly operation, thus it is only run twice a week.

Both forecasts and reforecasts are produced in ensembles in the ECMWF, with 50 members for forecasts, while reforecasts are made of only 10 [14].

The ECMWF not only provides medium-range weather predictions (up to 15 days ahead) but also **extended-range** (up to 39 days ahead) and long-range predictions (up to seven months ahead). The extended range dataset, which corresponds to **subseasonal to seasonal** predictions, is integrated into the WeatherBench2 dataset and will serve as **training data** in our experiments.

WeatherBench2 also contains the global climate and weather **reanalysis data ERA5** from the ECMWF. Climate and weather variables can be observed on Earth using satellites and in-situ means such as weather stations, buoys, ships or aircraft. However, these observations are not evenly distributed across the globe and the further we go back in time, the scarcer they are. Reanalyses aim to generate consistent historical time series of multiple climate and weather variables by combining past observations with today's NWP models [7].

Such dataset is built using statistical techniques known as **data assimilation**, from optimal interpolation to variational methods introduced by Courtier and Talagrand in the 1990's [9] and perfected ever since within the ECMWF with methods like four-dimensional variational data assimilation (4D-Var) [10].

ERA5 is the fifth generation and latest version of ECMWF reanalysis, providing hourly data on many atmospheric, land-surface and sea-state parameters together with estimates of uncertainty [7]. Data is available from 1940 onwards, with the years 1958 to 2023 being available in WeatherBench2. Satellites and in-situ data assimilated into ERA5 [8] come from several organisations such as the European Space Agency (ESA) or the NASA, coordinated under the Copernicus European Earth Observation program. Reanalyses represent a very reliable support to study climate change, and ERA5 is often used as **ground truth** for Machine Learning (ML) applications.

3.1.2 Dataset structure

Figure 5 summarizes the structure of WeatherBench2 and illustrates the richness of this benchmark. Outputs of data-driven models are also integrated into WeatherBench2 : Pangu Weather and GraphCast (see 2.1.2) are available, as well as older models like the Spherical CNN (2018) [6]. All of these models could also be used in a **data augmentation** setting. We will not use them in our experiments as we work on the post-processing task of the IFS outputs.

One specificity of weather forecasting datasets is their several temporal dimensions :

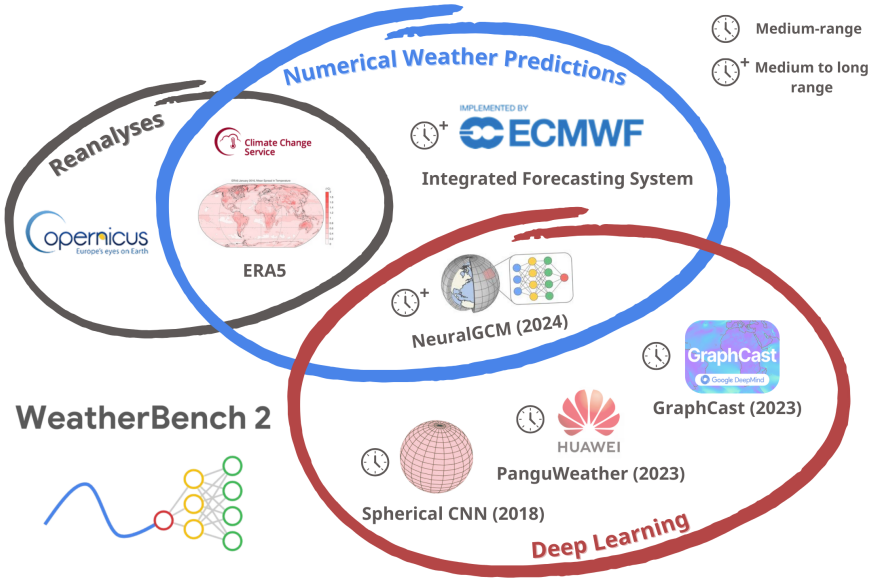


FIGURE 5 – The WeatherBench2 dataset (figure by author)

— **forecast time :** moment corresponding to the initial grid used to launch the forecast (initial conditions date).
— **lead time :** how far the model is predicting (a lead time of 2 weeks means we are predicting 2 weeks ahead).
— **valid time :** moment in time expected to match the conditions of a model state

The relationship between these time variables is :

$$\text{valid time} = \text{forecast time} + \text{lead time}$$

All of WeatherBench2 data comes in zarr files at **different longitude/latitude resolutions** : 1440x721 (with poles, 0.25degrees), 240x121 (1.5 degrees), 64x32. Many **temporal aggregations** were done as well (using rolling-windows) to create hourly, daily, and weekly data.

3.2 Data setup for post-processing task

This section describes the general data setup of this article : the division of the training samples, the choice of the input features, and how we formatted this large amount of WB2 data.

3.2.1 Temporal organisation

Our training data comes from **IFS extended range dataset** within WeatherBench2. We use the **weekly** temporal aggregations as for S2S, the predictions are only relevant at week-scale.

IFS reforecast data in WeatherBench2 cover the time period 1996-2022 while forecasts only cover 2016-2023. Even if our goal is to be able to post-process forecasts, we train our models on reforecasts and evaluate them on forecasts in order to get a larger number of training samples. This is standard practice, even if it poses issues as they usually have a different number of members [18], here 10 for hindcasts and 50 for forecasts.

We separate our samples based on **the year of their valid times** : 1996 to 2016 are reserved for training, 2017 for validation, and 2018 to 2022 for testing. Figure 6 summarizes the general temporal division of our dataset for training and testing.

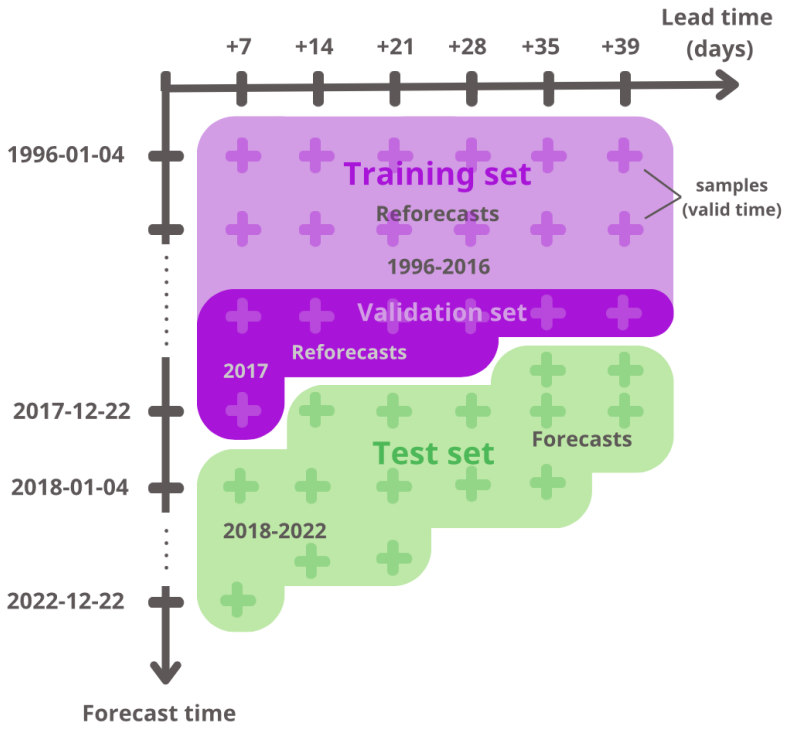


FIGURE 6 – Temporal division of our dataset

3.2.2 Input features

We want to correct the IFS forecasts of 2m temperature and 10m wind speed. The latter variable is derived from the latitudinal "u" and longitudinal "v" components of wind for both IFS and ERA5, using the Euclidean norm :

$$v_{\text{wind}} = \sqrt{u^2 + v^2}$$

Following [1], our models use input features X of different types to this end :
— physical variables, denominated **state-dependent predictors** by [1] : there are outputs from the IFS
— geographical/spatial data or **static predictors** [1] that describe constant characteristics of the surface model
— **time indicators** that specify the temporal context of the predictions

The IFS extended range data has a **1.5 degrees resolution** (240x121 grid) in WeatherBench2. We use all the physical variables available as state-dependent predictors. There are **18 single level variables** : minimal, maximal and mean 2m temperature, snow albedo, depth and density, soil moisture and temperature, sea surface temperature, sea ice cover, skin temperature (of the ground), total precipitation, cloud cover, etc.

There are 5 other variables available at 10 pressure levels (10, 50, 100, 200, 300, 500, 700, 850, 925, 1000 hPa) : geopotential, specific humidity, temperature, "u" and "v" components of wind. We consider each variable at each pressure level as an independent feature. This results in **47 features** after discarding 3 levels of specific humidity with only missing data. We also add the derived 10m wind speed at 1bar pressure level as an input feature.

Bouallègue et al. (2023) [1] found static predictors to improve the performances of their models, especially for 10m wind speed post-processing. We thus include some of these spatial features in our input features, such as the land sea mask (defined as the ratio of land per grid point, between 0 and 1), the latitude and longitude maps.

We also included temporal indicators for some models : the lead time of the prediction, the day of the year (doy) encoded using sine and cosine, following [1] and [25] :

$$(x, y)_{\text{doy}} = (\cos(\frac{\text{doy}}{366}), \sin(\frac{\text{doy}}{366}))$$

where doy represents the position of the day in the year (between 0 and 366 with February 29th included). This two-dimensional representation ensures unique coordinates for each day of year.

Finally, our data in WeatherBench2 is ensemble data, with several members. From these members, we can easily derive **means and standard deviations**, for all our lead times. The base input features presented above are all mean features. The same 67 base features are used for all models (66 NWP predictors and the land sea mask).

We only use the rest of the static predictors and time indicators for our deep approach. The standard deviations can be added as input features or serve as **priors** as we will see in Section 4. Appendix B gives the list of all features as well as the files they were extracted from.

3.2.3 Dealing with a large dataset

As we have seen in the previous section, the data we need from WeatherBench2 is quite large in terms of memory usage, with 10 members for reforecasts and 50 for forecasts.

The first challenge is to efficiently download such a large amount of data : 150G of pressure levels and 500G of single level forecast (test) data, 2T of pressure levels and 3T of single level reforecast (train) data. Instead of downloading the entire zarr file, it is faster to **divide it into chunks of data** : we take chunks of 20 forecast times for test data and 20 forecast times and 1 hindcast year for train data.

Downloading the data with this strategy and processing it (computing means and standard deviations) took approximately a week. We end up with **49G of test data, 943G of training data and 4G of observational data (ERA5)**. Such amount calls for a particular building of the dataset method in Pytorch. Indeed, it is simply **not possible to load everything** at once to access the elements. Furthermore, all of the train and test data is dispersed in **several files**, which makes their access even trickier.

This challenge can be overcome by using an **indexing strategy**. The idea is to constitute a list of information of where to find a specific piece of data. In this index, we store the paths of mean and std files, the positional index within the file, the forecast time, lead time and valid time of the sample. This way, we just have to go through this index to be able to recover the data we need.

The train and test index can be built only once and then stored. However, the data we load with this technique is not prepared at all (formatting, normalization, see Section 5.1), thus not ready for a direct usage during training. **The preparation of the data still needs to be done**, for example directly in the *getitem* method of the dataset class.

Unfortunately, this solution **slows the dataloader** because contrary to a classic *getitem*, it does just not only get the data but has to process it. One can choose to build a second dataset on top of this indexing one, to prepare and save the prepared data again in order to access the processed elements directly. We did not make this choice due to the **time consumption** it represents (downloading the raw data already took a week) and to **save memory space** (raw data already represent 1 Terabyte).

4 Models

In the spirit of [25], we want to compare different S2S post-processing methods and their "deep learning" versions. Because they can easily incorporate deep learning, we focus on "distributional-based" approaches in the terminology of [7], where we model our target variables with a parametric distribution at each grid point.

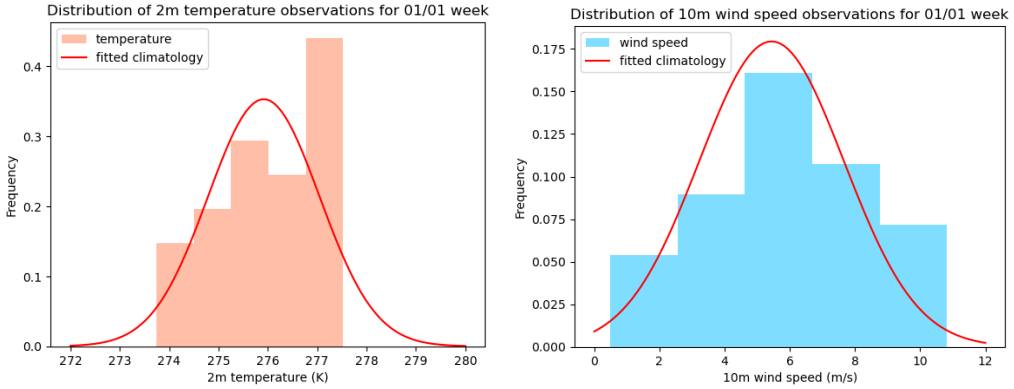


FIGURE 7 – Distributions of 2m temperature (left) and 10m wind speed (right) observations (ERA5) in Paris grid point on first week of January (doy=1), with the fitted climatology in red (based on 27 samples)

Figure 7 depicts examples of the distributions of our weekly aggregated target variables (ERA5 data) for the Paris grid point, as in S2S we model weekly variables. A common choice for temperature and wind speed is normal distribution : examples of Gaussian fits are plotted in red lines in Figure 7. We see that this assumption seems more correct for wind speed than temperature. We finally keep the normal distribution as it is a classical choice for temperature [35]. As both variables are always positive (temperature in Kelvin, speed above zero), other choices can be made such as a log-normal distribution or the more complex generalized truncated/censored normal distribution as in [31].

4.1 Distributional forecasting baselines

The raw data from ERA5 and the IFS can already be used to build simple models, especially the IFS with its 10 members.

Climatology The first baseline of all weather forecasting task is the climatology that we derive from the reanalyses, here weekly aggregated ERA5. A climate is defined over several years. As ERA5 is not an ensemble dataset, we get the mean and standard deviation per day of year, across the training years (1996-2016), that we use as the parameters of a Gaussian distribution. The climatology is thus constituted of 366 (February 29^{th} included) parameterized normal distributions, each built on the samples available for the corresponding day of year in the training set. An example is shown in Figure 7. This baseline contains weekly information even though we have one distribution per day of year as ERA5 data is aggregated weekly : doy 1 contains the mean of the 7 days from it (one week rolling-window).

RAW model The raw model is derived from the IFS ensemble data. We simply compute the mean and standard deviation of our target variables across the members, and plug it into a Gaussian distribution. The raw model plays a special part in the design of the next methods : both EMOS (section 4.2) and DRUNet (section 4.3) models use the raw model means and stds as **priors** for their predictions. Formally, our prior distribution $\mathcal{D}$ is the normal distribution $\mathcal{N}(\theta)$ with $\theta = (\mu, \sigma)$ derived from the IFS ensemble. The post-processing task consists in learning a model M such that $M(\mu, \sigma, X) = (\hat{\mu}, \hat{\sigma})$ by optimising a certain criteria (see section 5.2).

4.2 Statistical baseline : EMOS

Ensemble Model Output Statistics (EMOS) model by Gneiting et al. (2005) [17] is a popular baseline in S2S post-processing ([37] [25] [31]). This variant of multiple linear regression linearly corrects the prior mean and standard deviation from an ensemble forecast. If $(\mu_1, ..., \mu_n)$ be the predictions of the members of the ensemble, and σ their standard deviation, the original EMOS aims to learn $a_1, ..., a_m, b, c$ and d parameters such that the corrected distribution $\hat{\mathcal{D}}$ follows :

$$\hat{\mathcal{D}} \sim \mathcal{N}(a_1\mu_1 + ... + a_n\mu_n + b, c\sigma + d) \tag{1}$$

In order to determine the correcting parameters, we simply use a matrix formulation to combine our input features X. In the case of the EMOS models, we use the 67 base features presented in 3.2 (normalised, see 5.1.1, with temperature detrended, see 5.1.2) so $X \in \mathbb{R}^{67}$.

In our case, we want to post-process grid point-wise, for each variable, each lead time and each day of year, so have a different set of EMOS trainable parameters

for each. We have $n = 10$ members, so it would make in total 121 lat $*$ 240 lon $*$ 2 var $*$ 6 lead $*$ 366 doy $*$ 10 members $*$ 67 features i.e approximately 10^{10} trainable parameters to learn. In order to reduce this amount, we first decide to only learn 4 corrective parameters a, b, c, d for each single EMOS, such that the corrected distributions follows :

$$\hat{\mathcal{D}} \sim \mathcal{N}(a\bar{\mu} + b, c\sigma + d) \tag{2}$$

using only the mean of the members $\bar{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mu_i$ instead of all the individual predictions (reduction by 10). Then, here $\theta = (\bar{\mu}, \sigma)$ and we can write M for one grid-point as :

$$M(\theta, X) = (\beta X)^T \begin{pmatrix} \bar{\mu} & 0 \\ 1 & 0 \\ 0 & \sigma \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} a\bar{\mu} + b \\ c\sigma + d \end{pmatrix} = \hat{\theta} \tag{3}$$

with $\beta \in \mathcal{M}_{4,67}(\mathbb{R})$ the corrective matrix, and $\beta X = \begin{pmatrix} a & b & c & d \end{pmatrix}^T \in \mathbb{R}^4$.

In practice, we also need the new predicted standard deviation $\hat{\sigma} = c\sigma + d$ to remain strictly positive during training, as the linear transformation can make it negative. A simple trick seen in [25] is to use the exponential :

$$\hat{\sigma} = \exp(c \log(\sigma) + d) \tag{4}$$

with the logarithm helping keeping the new value to the original range (equal when $c = 1, d = 0$). However, the introduction of this exponential can make the model unstable and will need a strong regularization of the input features used to determine the parameters (see section 5.1.1).

As our number of parameters to learn is still significant, it is not efficient to train each model separately, both in computational and organisational terms (difficult to store, label and evaluate). Instead, we had to do a bit of data engineering in order to train for all grid points, for the days of one month, for one lead time, for one variable, leading to 12 month $*$ 2 var $*$ 6 lead making only 144 training and evaluation sessions for the 10^9 single models. This optimisation is made possible by the use of Pytorch tensors, with the individual models staying independent, just trained in parallel, as in [25]. Figure 8 summarizes the architecture of one of the 144 models, with its dimensions.

4.3 Going deep : DRUNet

EMOS is a simple model to implement but lacks flexibility [37], all the more when working on a grid point basis. Deep learning for S2S leverages more complex architectures but that are more flexible and allow to work globally, ie on the whole

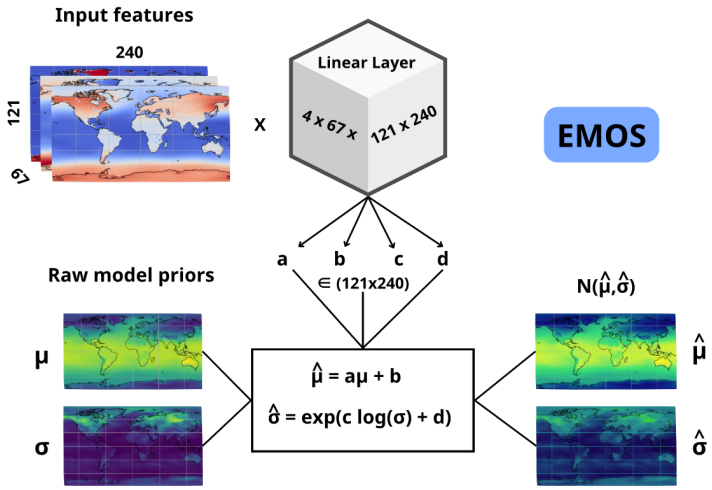


FIGURE 8 – EMOS architecture : The parameters a, b, c, d are linear combinations of the input features. They are used to correct the raw model priors using equation (2).

world with one model. Fan et al. [15] used Neural Networks at different scales (local, regional and global), while Horat et al. [18] also played with the scale of their U-Net architecture. Both have shown the **advantage of global models versus regional ones**. Therefore, we decided to build a single model for the whole world. In particular, **U-Nets make use of spatial information using convolutional filters**, instead of treating each grid point independently as in EMOS, which could improve our performance, while the **skip connections allow to preserve high resolution details.**

A recent work by Pic et al. (2024) [31] proposes a global U-Net architecture to post-process S2S precipitation ensemble forecasts using distributions : they call it Distributional Regression U-Nets (DRUNet). This model takes input features and outputs the parameters of the distribution of our choice to model the variable, similarly to EMOS. Although they worked only on the Mediterranean area, their 112x192 resolution is similar to ours (121x240) and allows us to reuse their architecture as it is. Because the U-Net architecture needs to divide by two the spatial dimensions several times, we get rid of the -90° latitude to have an even resolution : 120x240.

As input features, the DRUNet needs more information apart from the 67 base physical features. Indeed, as it is trained for all lead times and all doy at the same time, we add time indicators : normalized lead time and encoded doy (see section 3.2). The convolution layers of the U-Net make it spatial, so we provide additional

static predictors such as latitude and longitude data. We also add the prior standard deviation of wind speed and temperature.

Indeed, we can use the DRUNet in two settings : predicting a correction using raw priors as in EMOS (see section 4.2), or directly predicting the variable of interest (i.e. direct forecasting). We call these models respectively **DRUNet+prior** and **DRUNet**. Adding the prior standard deviations of wind speed and temperature as input features is thus necessary without priors, and can also improve DRUNet+prior performances. **All of our DRUNet then have 74 input features (69 NWP predictors, 3 time indicators and 2 static indicators)**, i.e. $X \in \mathbb{R}^{74}$.

DRUNet+prior is not exactly following the EMOS equation (2). Indeed, after several experiments we found that it was more beneficial to only learn the bias of the raw prior, without scaling parameter :

$$\hat{\mathcal{D}} \sim \mathcal{N}(\bar{\mu} + b, \exp(\log(\sigma) + d)) \tag{5}$$

as if $a = c = 1$. We found this formulation to be more stable for training and leading to better results. DRUNet without prior, that is to say **direct forecasting**, can be written as :

$$\hat{\mathcal{D}} \sim \mathcal{N}(b, \exp(d)) \tag{6}$$

as if $a = c = 0$. It could be interesting to try if our architecture is also fit for direct forecasting.

Even though the input features X is of higher dimension than for EMOS, due to the weight-sharing (convolutional filters) in UNet architecture, the reduced number of corrective parameters (b,d only) and as we have only one model for all lead times and all day of year, **the number of trainable parameters in DRUNet is much lower than EMOS**, around $6 * 10^5$ at most (versus 10^9).

Finally, the DRUNet architecture allows for an interesting coupling in the prediction of variables' distributions. Contrary to EMOS where variables can only be post-processed separately, we can use the same DRUNet to produce the parameters of our two variables, denoted **DRUNet both**. Sharing the weights across these two tasks could enhance our performances according to Stein's phenomenon in statistics : it is better to estimate several Gaussian random variables (n $>$ 2) altogether rather than separately, even when the Gaussians are independent [34]. Training two variables together acts as a regularization. It also has its set of challenges, as optimizing a net for 2 objectives - the loss being the sum of temperature and wind losses - is a harder problem than a single objective.

Figure 9 illustrates the general DRUNet architecture . We focus on the following combinations : DRUNet both, DRUNet+prior both, DRUnet+prior single (for single

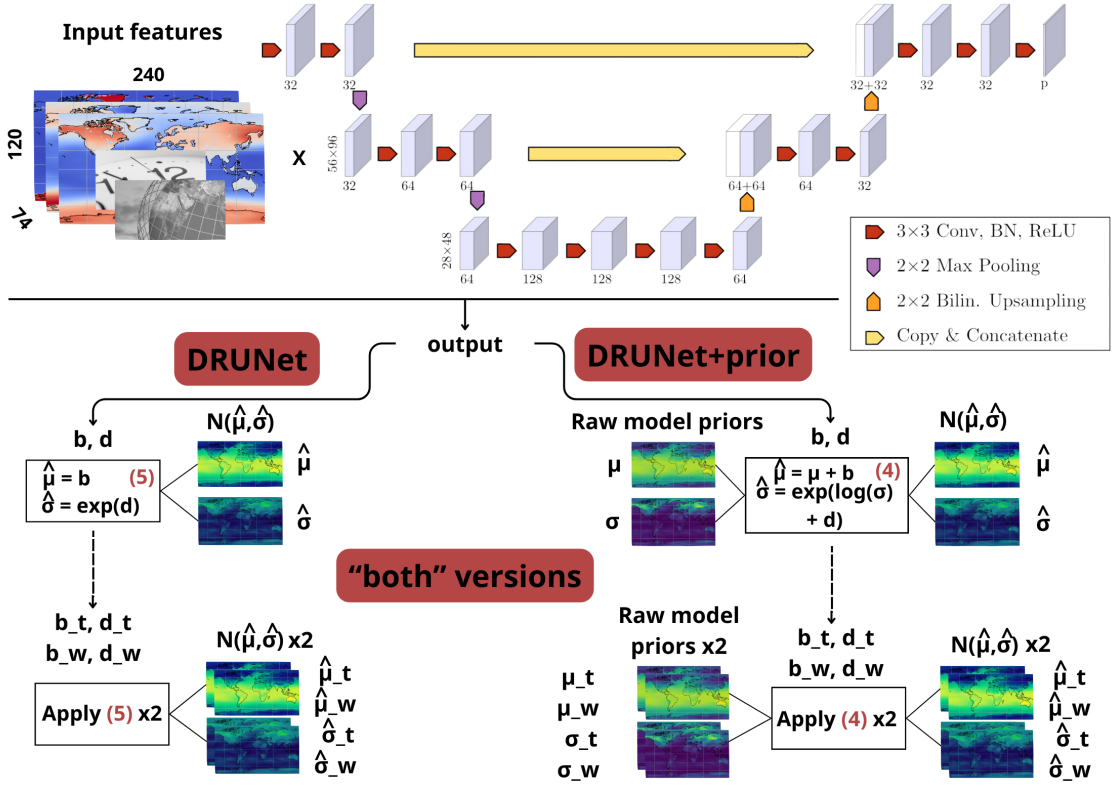


FIGURE 9 – DRUNet architectures derived from Pic et al. It takes extended input features compared to EMOS, and has different versions : DRUNet with no priors following (6), DRUNet+prior following (5) and their "both" versions that forecast both temperature (t index in the figure) and wind speed (w index in the figure) using the same equations.

target variable). An additional model, DRUNet single+, is designed for direct wind speed forecasting by adding orography, high and low vegetation cover as they are of particular interest for wind speed according to Bouallègue et al. [1].

5 Final experimental pipeline and Results

Now that we have defined most of the elements we need for this study, we are left with associating them in a final experimental pipeline. To this end, we first design the data preparation or **pre-processing** for our models (section 5.1). We then define the different **metrics** involved during training and evaluation (section 5.2) before presenting the **results** of our experiments in section 5.3.

5.1 Pre-processing

In section 3.2 we have defined the input features and general setup we use to access the data. However, our different models presented in section 4 need different pre-processing and data formatting. These different data preparation pipelines will share the same global steps :

1. Choice of the temporal aggregation (section 5.1.1)
2. Detrend the training and ground truth temperatures using the learned X_{trend} and Y_{trend} learned on the training years of respectively IFS data and ERA5 data (section 5.1.2)
3. Normalize the input features individually and grid point-wise using learned min-max normalizations (*detrended* temperature is normalized). We don't normalize our target variables. (section 5.1.1)

The different pre-processing pipelines are summarized in the Figure 10 below. Target temperature (ERA5) is detrended using Y_{trend}, while both prior and input mean temperatures are detrended using X_{trend}. We then perform a min-max (MM) normalization with either monthly and lead aggregation, or a general aggregation (see section 5.1.1). These are then the input features for EMOS, and we extend DRUNet input features with temporal and static indicators as explained in section 4.3.

In this section, we explain the choice of the min-max normalization strategy instead of normal standardization based on the characteristics of our physical variables (section 5.1.1), and explain why 2m temperature needs a detrending (section 5.1.2).

5.1.1 Normalization strategies

As the physical variables we use as input have very different ranges, it is better to **normalize** them, that is to say shift them to the same magnitude. This operation stabilizes the training of the model and brings balance to the influence of each variable in the network. A normalization protocol for S2S post-processing is the combination of 3 components : a temporal aggregation, a spatial aggregation and a normalization type or method.

Temporal aggregation The temporal aggregation used to build the normalization depends on the structure of the technique. As EMOS has one model per lead time and per month, we perform the normalization on elements with the same lead time and from the same month (24 different normalizations). We denominate this methodology as **[Month Lead Agg]**.

For the DRUNet, we can also use the **[Month Lead Agg]** by checking the lead and month of the sample and applying the corresponding normalization. However,

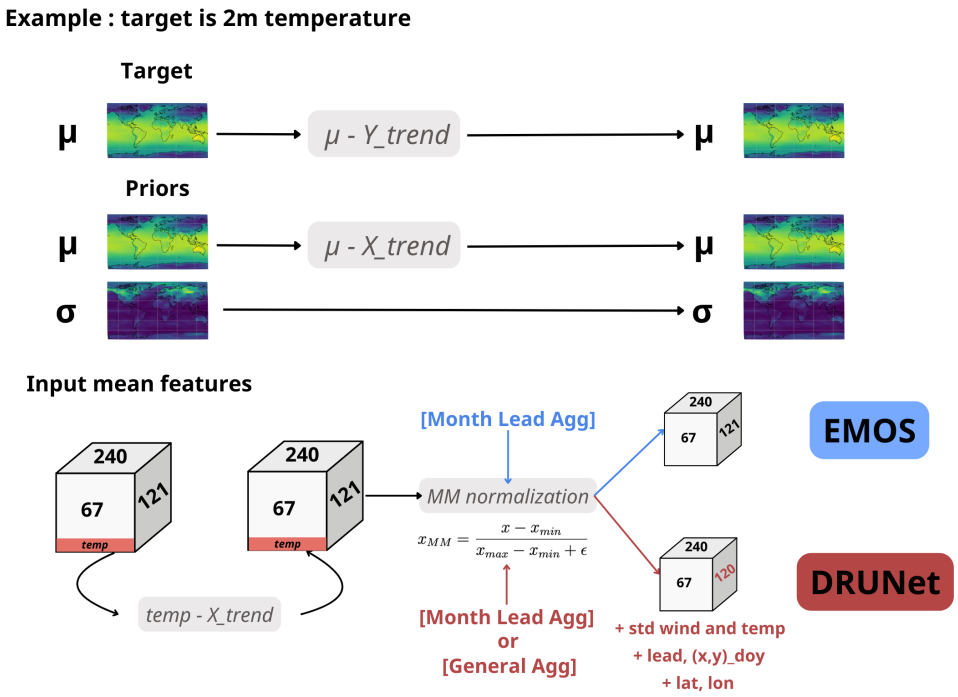


FIGURE 10 – Data preparation pipelines of EMOS and DRUNet models, example with target 2m temperature.

as we have one model for all lead times and months, we can also try to use the same normalization, namely **[General Agg]**, built on all the available samples. This could enhance the stability during training and lead to better performances, as the model is seeing samples from different leads and months contrary to EMOS.

Spatial normalization Spatially, we want to perform a normalization **per grid point** of the Earth map.

Normalization method for physical variables With at least 67 physical variables, it is no easy task to find the appropriate normalization strategy. The most common methods are :

— **normal standardization :** If the underlying distribution of our data can be considered Gaussian, we can shift it to a normal distribution of null mean and unit variance $\mathcal{N}(0, 1)$. It can be done removing the mean μ and dividing each sample by the standard deviation σ of the dataset :

$$x_{norm} = \frac{x - \mu}{\sigma}$$

This normalization preserves the original distribution (especially the extreme

values distribution) while compressing its ranges.

— **min-max (MM) normalization :** If the data does not have a known underlying distribution or is sparse (lot of zeros), we can compress it to $[0, 1]$ with min-max normalization with :

$$x_{MM} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

This normalization has the disadvantage to compress based on the extreme values and can degrade the original data, but it ensures that the data norm is under one.

We explain in the following paragraphs the process we went through to choose between these two strategies to **normalize our mean maps** (mean over members of the ensemble, see section 3.2). As stated in section 3.2.3, one major difficulty is that it is impossible to load all the data at the same time. We could only work on parts of it and this lack of visibility complicated the elaboration of the normalization protocol.

When working on **[Month Lead Agg]** with EMOS in the beginning of the internship, our first strategy was to standardize (using mean/std) all variables together. However, it was not optimal for a good representation of the data, as variables within similar ranges of values ended up looking the same, losing information. We thus decided to normalize **each variable independently**.

With **[Month Lead Agg]**, it was still possible to load the training data entirely. However when we developped the **[General Agg]** for the DRUNet, we needed to build mean/std or min/max parameters for normalization **incrementally**, as we could only open the files one by one. We first tried to implement the normal standardization, by building incremental means and variances. It appeared difficult, in the management of missing values as well as divisions and squares in the variance formula causing an accumulation of errors. Building an incremental min/max is much easier as we just have to update the current min/max with each new file.

But the choice of the normalization strategy mostly relies on the characteristics of our physical variables. We took the time to study all of them, as they were very diverse. Their diversity constrained a lot what was possible or not for normalizing them with a single method, for the normalization protocol to be easily to generalize when adding new input features.

Among the different characteristics there are (see Figure 11) :

— Variables defined everywhere, with coherent mean and enough variability gridpoint-wise to define a variance, such as **2m temperature**. These variables can be normalize using normal standardisation.

— Variables not defined everywhere by definition : for example **sea surface temperature** is NaN (Not a Number) everywhere on land. This means that we cannot have a variance on land grid points, and that min=max on them as well.

— Variables defined everywhere but with a very high variance, non gaussian such as **snow depth**. Its extreme values are still high after normal standardisation, which lead to instabilities in our architecture due to the exponential in the standard deviation post-processing.

— Variables defined everywhere but always 0 in some grid points such as **total precipitation**. Again, we have no variance information on these grid points.

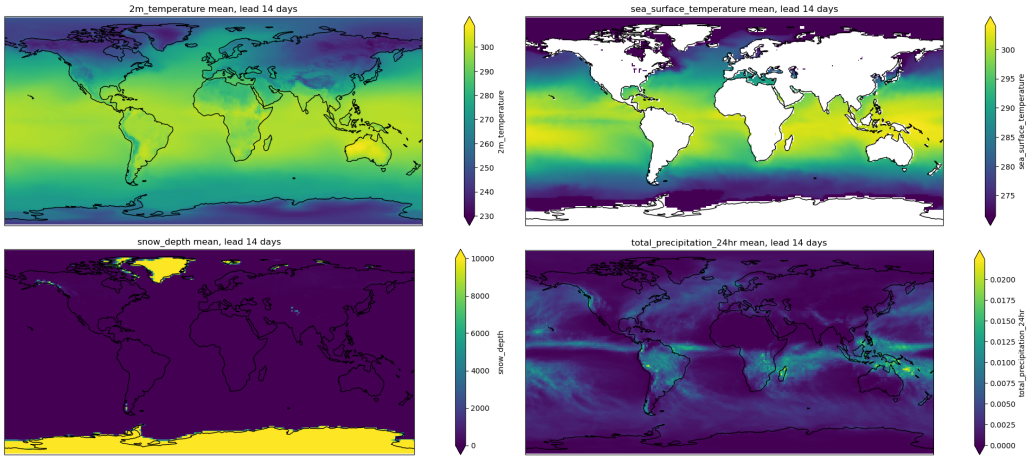


FIGURE 11 – Diversity of variables (mean train data, lead +14 days, on 1996-01-18) : from left to right, top row : 2m temperature and sea surface temperature ; bottom row : snow depth and total precipitation

We see that building mean and std parameters for normalization becomes tricky with inconvenient processing needed for many variables. We had to deal with both intrinsic and fortuitous NaNs at the same time, and some variables would still produce large outliers such as snow depth. On the contrary, MM normalization was better suited to handle both outliers and "no variance" variables, ensuring values between 0 and 1 to avoid any numerical instabilities during training and dealing with min=max grid points with only a simple trick during normalization :

$$x_{MM} = \frac{x - x_{min}}{x_{max} - x_{min} + \epsilon}$$

with ϵ small (10^{-3} in practice).

For both **[Month Lead Agg]** and **[General Agg]**, we thus decided to apply MM normalization per variable, per grid point. When the normalized mean features or

mean priors (from RAW) contained *missing values*, we replaced them by 0. Missing or ill-defined std priors (from RAW) were replaced by 1.

5.1.2 Detrending temperature

As our models use priors, it is very important to take into account their physical characteristics, as we have done during normalization. In particular, 2m temperature has a specificity 10m wind speed does not. Besides the internal variabilities (seasonality etc), temperature presents a **warming trend** over the years, due to climate change. In Figure 12, we can see this trend in both training and observational data, as well as the bias we are trying to correct between the truth and RAW model. We see that the trend is not the same in the forecast and analysis datasets for 2m temperature.

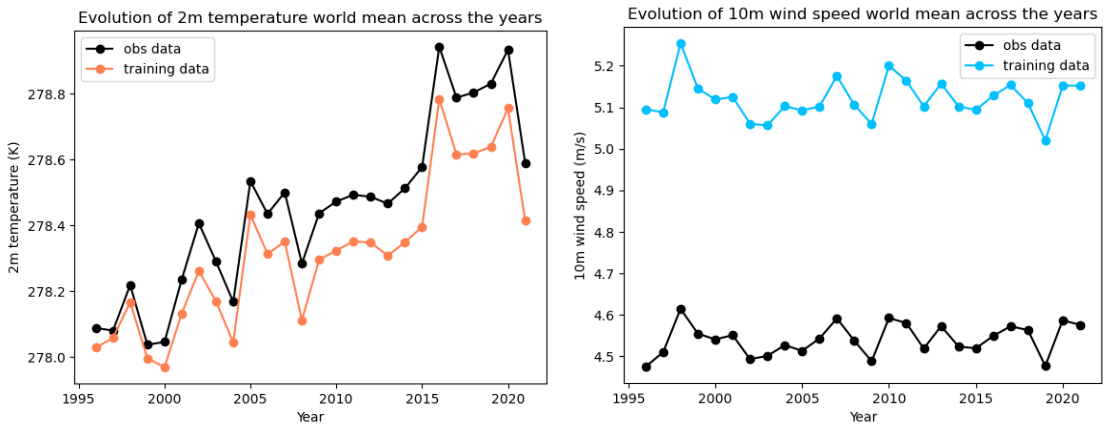


FIGURE 12 – Target variables trends (temperature left, wind speed right) of observations (ground truth) and training data

This misaligned trends issue that wind speed does not have can hinder our ability to generalize on unseen years. Indeed, our models see the 1996-2016 years and will correct the biases learned over these years, but because analysis and forecast data have different trends, the bias in the test years will be different and the model will not be able to perform well (see Figure 13 left).

In order to avoid this issue, we need to **detrend** the temperature data, that is to say modelling the trend and remove it. Many methods exist like linear (the one we use) or quadratic models. We need to learn both the training and ground truth data trends, respectively X_{trend}, Y_{trend}, on the training years only to avoid data leakage. During training, we just remove the trends and at inference time, we need to add back Y_{trend} to the model's predictions. Figure 13 (right) shows the learned Y_{trend}

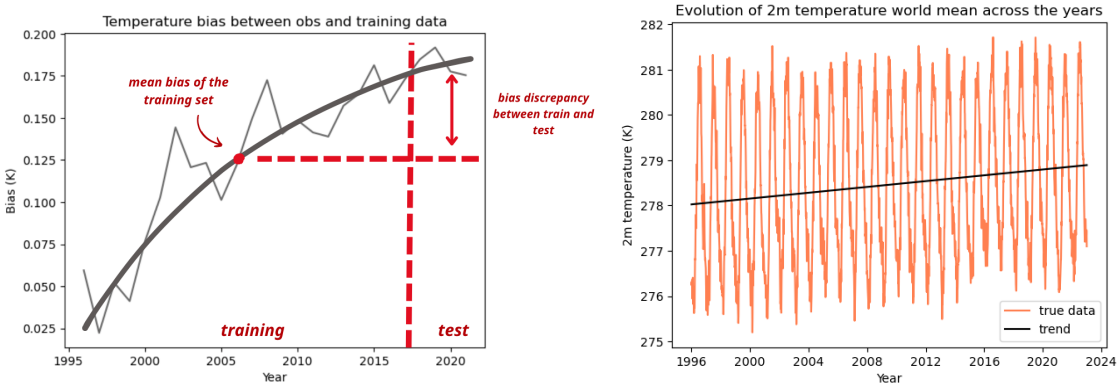


FIGURE 13 – Left : Trend issue, Right : Y_{trend} fit on truth data

using a simple linear regression. The oscillations we observe show the seasonality of temperature.

5.2 Metrics and Losses

This section introduces the different metrics we use to train our models as well as to evaluate them. A down arrow ↓ (respectively up arrow ↑) means that the lower (the higher) the metric, the better the performances.

CRPS ↓ One of the common metrics in S2S is the Continuous Ranked Probability Score. Let X be a random variable (our predicted distributional forecast), let F be the Cumulative Distribution Function (CDF) of X i.e. $F(x) = \mathbb{P}(X \leq x)$. Let y be the observation (ground truth), then the CRPS is defined as :

$$CRPS(F, y) = \int_{\mathbb{R}} (F(x) - 1_{\{x \geq y\}})^2 dx \tag{7}$$

with 1 denoting the Heaviside step function (1 if the argument is positive, 0 otherwise).

The CRPS has the same unit as the observed variable. It reduces to the Mean Absolute Error (MAE) if the forecast is deterministic i.e. just a single prediction $\hat{y}$:

$$\begin{aligned} CRPS(1_{\{x \geq \hat{y}\}}, y) &= \int_{\mathbb{R}} (1_{\{x \geq \hat{y}\}} - 1_{\{x \geq y\}})^2 dx \\ &= \int_{\min(y,\hat{y})}^{\max(y,\hat{y})} 1 dx \\ &= |y - \hat{y}| \end{aligned} \tag{8}$$

This property allows to easily compare the performances of ensemble and deterministic forecasts [38]. In the case of particular distributions for X like the normal distribution, the CRPS also has a literal expression :

$$CRPS(\mathcal{N}(\mu, \sigma^2), y) = \sigma \left(\omega \left(2 \cdot \Phi(\omega) - 1 \right) + 2 \cdot \phi(\omega) - \pi^{-1/2} \right) \tag{9}$$

with $\omega = \frac{y-\mu}{\sigma}$, and ϕ being the Probability Density Function (PDF) of the standard normal distribution, Φ its CDF. The computation of this CRPS integral is actually non trivial and requires several integrations by parts [35].

Figure 14 by Faran I. [16] gives a visual representation of the CRPS metric : we can see it as an area between two curves, the CDF F of our prediction and the Heaviside CDF of the deterministic ground truth. The CRPS score penalizes overly confident and wrong predictions (small σ, inaccurate μ) as well as too uncertain predictions (large σ), even if the μ is accurate [38].

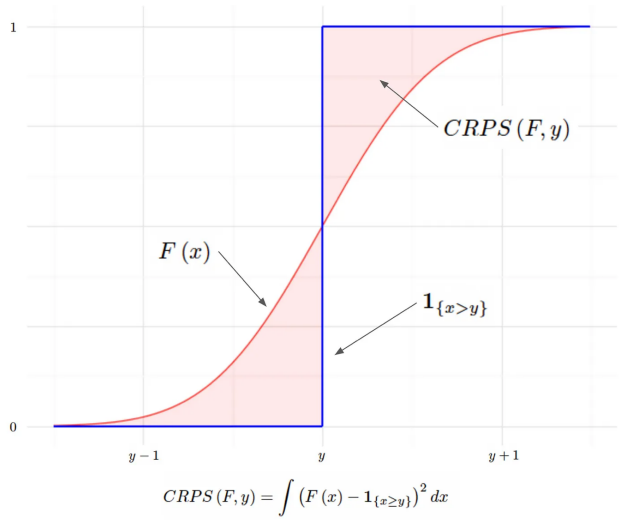


FIGURE 14 – The predicted distribution is marked in red, and the ground truth's degenerate distribution is marked in blue. The CRPS is the (squared) area trapped between the two CDFs. Image from [16]

As in the original EMOS [17], the **CRPS will be used as our loss function during training**.

CRPSS ↑ In S2S, it is important to know if a model is performing better than the baseline, most of the time the climatology (see 4). We use a metric called **skill**,

built with the scores of our model and the climatology. The CRPSS, for Continuous Ranked Probability Skill Score, is defined as :

$$CRPSS_{model} = 1 - \frac{CRPS_{model}}{CRPS_{climatology}} \tag{10}$$

The higher the CRPSS, the better the model performs compared to climatology. The skill is not reserved for CRPS and can be defined for other scores like the ranked probability score (RPS) and corresponding RPSS. We can also define the CRPSS with another baseline than climatology, for example the RAW model.

We will use **CRPSS (vs climatology)** as the **evaluation** metric of our study. We are only interested on its values on **land** as in the WMO challenge [12] so we will use a land sea mask. Its values are the ratio of land in the grid point, so between 0 and 1. We use a 0.5 threshold to turn it into a mask (see Figure 15, top). We will also look at its mean scores at different latitude regions.

When evaluating our performances, it is important to take into account the real size of our gridded map. Indeed, not all grid points represent the same area due to the curvature of the Earth, the grid points at the poles cover smaller areas than the ones at the Equator. In order to model this, when doing any kind of mean, we weight it using the cosine of the latitude that is proportional to the area. This way, we put more weight on the grid points accounting for larger areas, as you can see in Figure 15 (bottom). However, training DRUNet using this weighted CRPS caused the 90° latitude weights to be put to zero and not trained, disrupting the entire zone due to the spatial sharing of the model weights. We decided not to apply latitude weighting during training, but we did for validation to monitor the CRPS.

5.3 Experiments and Results

Both EMOS and DRUNet were trained in the same setup, using **Adam** optimizer with a **learning rate 0.01** and a **batch size 128**. We trained for several epochs, saving a model checkup every 5 epochs. We select the model version from the epoch with the lowest training loss and before a steady increase in the validation loss, indicating the model is overfitting. It results in an optimal number of 9 epochs for EMOS 2m temperature, and 15 epochs for 10m wind speed. The DRUNet usually converged quicker, in only 5 epochs for both variables.

We present here the results of the models yielding the best post-processing outcomes : EMOS, DRUNet+prior both [Month Lead Agg], DRUNet+prior single [Month Lead Agg] and best direct forecasting outcomes : DRUNet both [General Agg] and DRUNet single+ for wind speed only. DRUNet both [Month Lead Agg] performed similarly than its [General Agg] version and is not presented here.

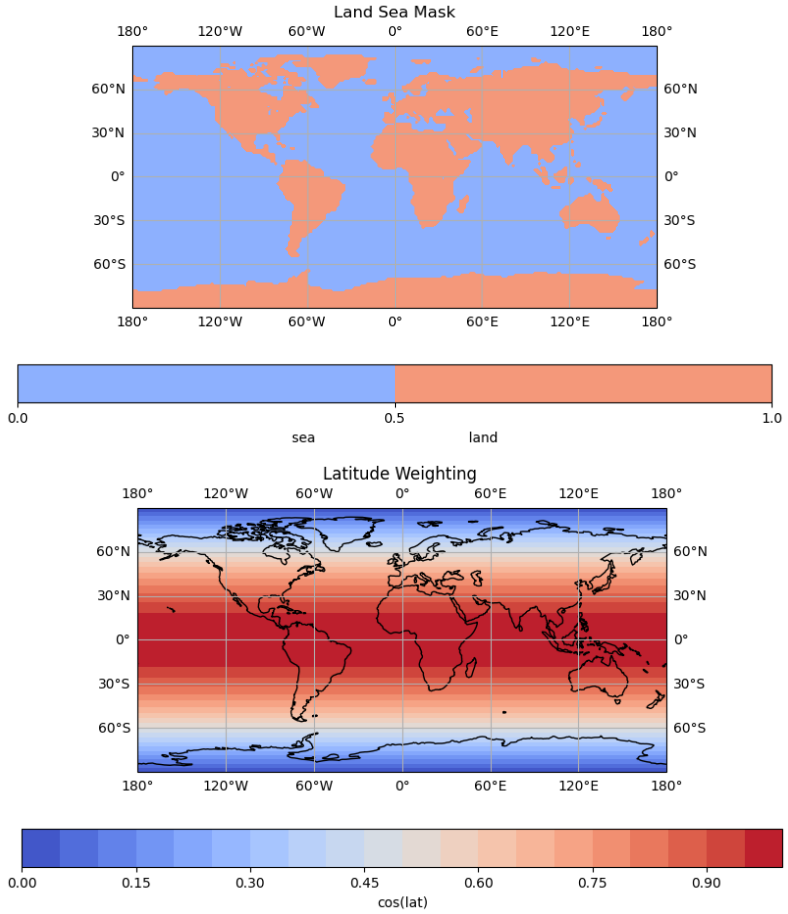


FIGURE 15 – Land Sea Mask (top) and Latitude weighting (bottom)

5.3.1 Global performances

Figure 16 shows the evolution of the land CRPSS (vs climatology) of the models across lead times for 2m temperature (left) and 10m wind speed (right), to give a first picture of our global performances. DRUNet both is not showed on the temperature panel as it performs to badly compared to climatology (-2 CRPSS). As explained in section 5.2, if the CRPSS is above 0, the method performs better than climatology, under it performs worse and 0 is the climatology (grey dotted lines).

On both panels, we can observe the decrease in performance when the lead time increases in all methods. The raw CRPSS curves (in orange) illustrates the difficulty of S2S forecasting, with the forecast skill declining abruptly after 7 days. The ECMWF raw model only performs better than climatology at medium-range (7 days) for temperature and its wind speed predictions all have negative CRPSS, showing the need of some post-processing.

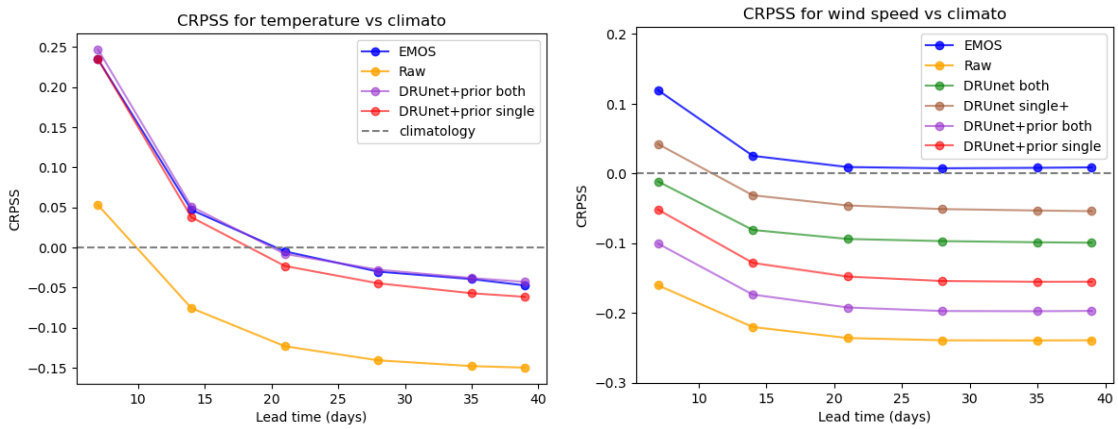


FIGURE 16 – Global land CRPSS (vs climatology) weighted by latitude across lead times : 2m temperature (left) and 10m wind speed (right). DRUNet both is not showed for temperature as it performs too badly. The grey dotted lines represent the climatology.

We see that most of our post-processing methods are successful in improving raw predictions, but are not always doing better than climatology. The S2S post-processing task also gets harder with greater lead times : EMOS and DRUNet+prior single are able to improve temperature predictions by 24% and DRUNet+prior both even reaches 25% at 7 days, 5% at 14 days, but then no model performs better than climatology after 3 weeks. For 10m wind speed, EMOS is the only model outperforming the climatology at all lead times, respectively improving +7 and +14 days by 12% and 2%, although it quickly plateau after 3 weeks around 0.8 % improvement only. In direct forecasting, only DRUNet single+ (brown line) manages to improve for lead time 7 days but has negative CRPSS for bigger lead times.

5.3.2 Optimizing for a single or both variables

Overall, EMOS is the most effective and stablest method among the ones tried here, having some success in both temperature and wind speed. On the contrary, we observe in Figure 16 that the DRUNet-based methods perform very differently depending on the variable.

DRUNet both (green lines), which does not use any prior and predict both variables' distributions, has the worst CRPSS for temperature but one of the best for wind speed, although still worse than climatology. Still on the "both" strategy, adding the prior in DRUNet+prior both (purple lines) greatly helps the model for temperature that even equates EMOS in performance, while decreases it for wind

speed compared to DRUNet both. This seems to indicate the existence of a competition between our variables in this multi-objective task, that cannot find an optimal solution for both variables at the same time.

On the other hand, DRUNet+prior single optimizes the model for each variable separately. Again, the results of Figure 16 are quite surprising : when trained only for temperature, the model does worse than the one trained for both variables. This is somewhat in agreement with the findings of Bouallègue et al. [1] : wind speed forecasts play a key role in estimating 2m temperature systematic error, thus correctly modelling wind speed within the model can enhance the performances on temperature. Training DRUNet+prior single for wind speed is more advantageous, however this could be linked to an imbalance in the "both" training setup.

This is why it could be interesting to improve our multi-task training. I investigated some techniques explained in Xiao B. [41] to balance my multiple loss functions, such as normalizing the two terms of my loss by their initial values, in order for both terms to be in the same ranges and thus considered equivalently by the model during training, but it did not change the results too much. More complex approaches could be investigated such as Multiple-gradient descent algorithm (MGDA) by Désidéri J.A. [11], famously and successfuly leveraged for Auto-Encoders and more by Sener et al. (2019) [34].

5.3.3 Discussing spatial strategy for wind speed post-processing

A disappointing result are the performances of the DRUNet-based methods on wind speed, that do not seem to be adapted. We believe that the convolutional model has difficulties modelling sharp changes in wind speed, which is a more local variable, strongly dependent on spatial discontinuities like orography as found in Bouallègue et al. [1]. Wind speed values are under 4m/s on land but much stronger on the sea, whereas temperature is more homogeneous across the Earth and has a continuity above the oceans as we can see in Figure 17.

Such discontinuity, likely due to the abrupt change in surface type, can be hard to learn. We see on Figure 18 (top row) that the raw model indeed struggles on the coasts, whereas a gridpoint wise approach like EMOS (top left) is free of these considerations, and thus performs well for this type of variable.

This issue is even worse for a UNet-like architecture where the weights are shared across neighbouring cells. DRUNet both has negative CRPSS in mountainous areas such as the Andes cordillera, at the western and northern mountains in Congo and Ethiopia surrounding Victoria Lake, or in Himalaya in Asia. These regions all have abrupt changes in topography or vegetation cover, making it challenging for spatial-informed models (see annex C).

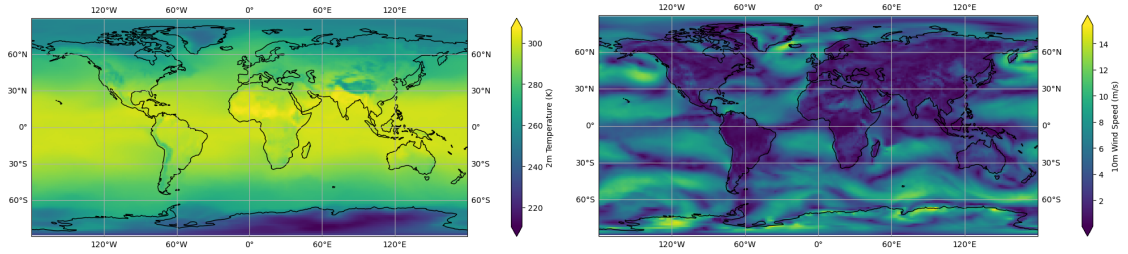


FIGURE 17 – ERA5 2m temperature (left) and 10m wind speed (right) on the 1996-04-13. Temperature is a homogeneous field over land and sea, whereas wind speed is strongly dependent of the surface type.

Adding these important static predictors identified in [1] (low and high vegetation cover, standard deviation of subgrid orography, available in ERA5 WB2) in DRUNet single+ improves the results but does not outperform EMOS (Figure 16). We see in Figure 19 that it still struggles in the same mountainous areas as DRUNet both, although performances on low lands in northern Asia and Antarctica are improved. It proves that these indicators are indeed essential to wind speed forecasting, although not sufficient here within a UNet.

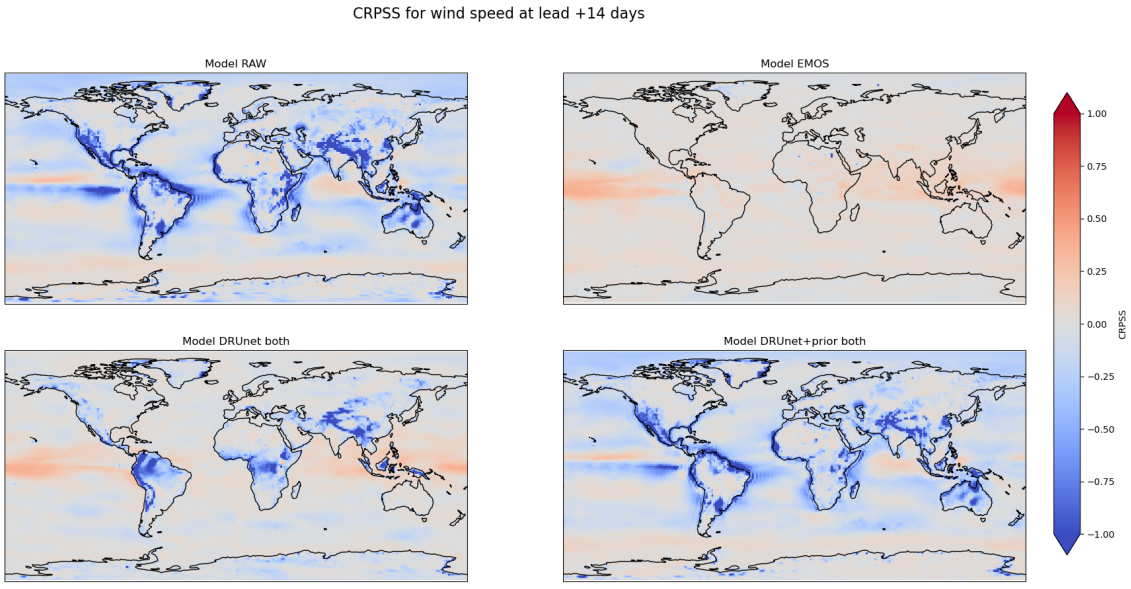


FIGURE 18 – CRPSS of 10m wind speed at lead 14 days for top row : raw (left), EMOS (right) ; bottom row : DRUNet both (left), DRUNet+prior both (right).

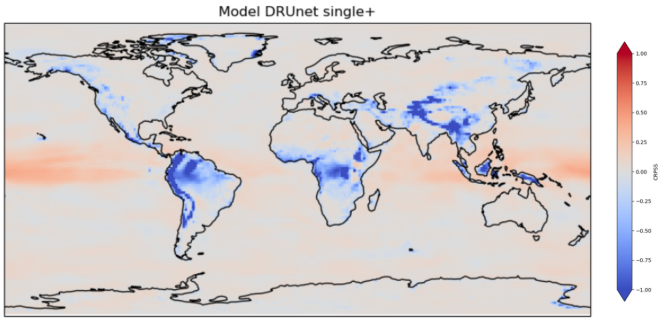


FIGURE 19 – CRPSS of 10m wind speed at lead 14 days for DRUNet single+

The hypothesis of the spatiality of the UNet being a weakness for wind speed post-processing is also being corroborated by the decrease in performance when we add priors to DRUNet both. As we know, these priors from the raw model are bad, and with the UNet convolutions being too conservative, it leads to poorer results than without. We indeed observe the same negative CRPSS patterns in the coasts for raw and DRUNet+prior both (Figure 18 bottom right).

5.3.4 Discussing multi-model approach

Figure 20 also confirms our doubts on a UNet architecture being very conservative. Model RAW is mostly struggling on grid points at high altitude, and both DRUNet+prior both or single present some remnants of these bad predictions in South America, West Africa and in the Himalaya mountains. On the contrary, EMOS does not present the same negative CRPSS patterns and actually improves most of them, even in altitude. However, we see that EMOS degrades the CRPSS in Greenland or western Asia compared to the raw priors, whereas the DRUNet architectures keep the raw model patterns.

We then understand the power of multi-model strategies, such as the winning model of the WMO challenge [12], called an "opportunistic mixture model". It is a weighted ensemble of models composed of climatology, raw, EMOS and a CNN (ResNet). As we have seen in Figure 16, the climatology is hard to bet after 3 weeks, so its weighting in the mixture model increases with lead time. Our models could probably benefit from this strategy for 2m temperature as well, although it is more costly to run several models rather than one. Similarly, the mixture model can also be spatial, based on topography and that would use EMOS in mountains and DRUNet+prior in other places.

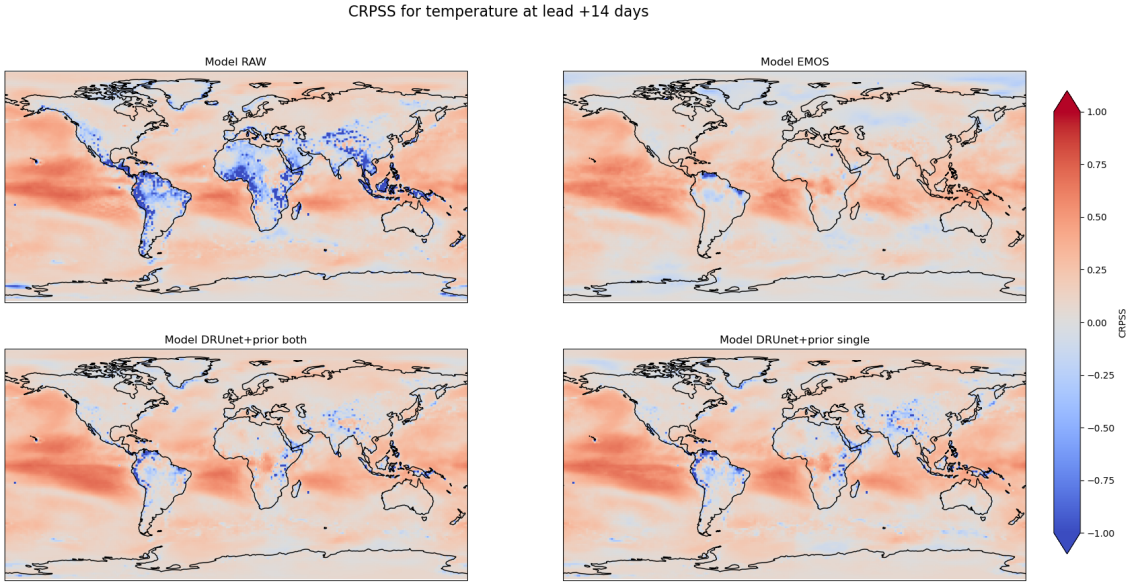


FIGURE 20 – CRPSS of 2m temperature at lead 14 days for top row : raw (left), EMOS (right) ; bottom row : DRUNet+prior both (left), DRUNet+prior single (right).

5.3.5 Spatial performances

For more precise results, Table 1 references the CRPS values of our different architectures for 2m temperature and 10m wind speed on land. The results are aggregated into different lead time categories as it is classicaly done in such studies [12] : week 1+2 for medium-range and week 3+4, week 5+6 for S2S range. We look at the global CRPS and also by latitude zones (90°–30°N, 30°N–30°S, 30°S–90°S). Our baseline, climatology, is steady across lead-times, thus only filled once in the table. CRPS values are in bold when they beat the climatology, and when no model outperforms climatology, the second best is underlined.

For 2m temperature, the CRPS go from 0.17 to almost 1K for EMOS and the 2 DRUNet+prior methods, whereas it is almost 2K at best for DRUNet both. We see that northern latitudes (90°–30°N) predictions are the hardest to improve in general, with no model outperforming climatology after 2 weeks. Central and southern latitudes can be improved a little at week 3+4 (respectively by EMOS and DRUNet+prior architectures). At week 5+6, models are at best either really close to (EMOS for 30°N–30°S) or equal to climatology (DRUNet+prior single for 30°S–90°S).

For 10m wind speed, the values are smaller, between 0.09 and 0.25 m/s for EMOS while the other models span between 0.1 and 0.3 m/s. We also observe poorer results

Lead time	Model	2m temperature (K)				10m wind speed (m/s)			
		90°N–30°N	30°N–30°S	30°S–90°S	Global	90°–30°N	30°N–30°S	30°S–90°S	Global
All	Climatology (baseline)	0.927	0.206	0.288	0.407	0.252	0.097	0.122	0.142
Week 1 + 2	RAW	**0.855**	0.266	**0.261**	0.412	0.290	0.131	0.128	0.169
	EMOS [Month Lead Agg]	**0.795**	**0.170**	**0.261**	**0.350**	**0.238**	**0.087**	**0.116**	**0.132**
	DRUNet both [General Agg]	1.901	0.339	0.430	0.744	0.264	0.104	0.124	0.149
	DRUNet+prior both [Month Lead Agg]	**0.792**	**0.177**	**0.250**	**0.346**	0.281	0.121	0.124	0.161
	DRUNet+prior single [Month Lead Agg]	**0.808**	**0.181**	**0.249**	**0.352**	0.273	0.114	**0.121**	0.155
Week 3 + 4	RAW	0.994	0.279	0.291	0.461	0.301	0.135	0.134	0.175
	EMOS [Month Lead Agg]	0.956	**0.202**	0.294	0.415	**0.250**	**0.095**	**0.122**	**0.141**
	DRUNet both [General Agg]	1.982	0.352	0.450	0.775	0.273	0.111	0.130	0.155
	DRUNet+prior both [Month Lead Agg]	0.953	0.216	**0.287**	0.415	0.294	0.128	0.131	0.169
	DRUNet+prior single [Month Lead Agg]	0.974	0.220	**0.286**	0.421	0.287	0.121	0.128	0.163
Week 5 + 6	RAW	1.019	0.280	0.293	0.468	0.302	0.135	0.134	0.176
	EMOS [Month Lead Agg]	0.986	0.207	0.297	0.425	**0.250**	**0.096**	**0.121**	**0.141**
	DRUnet both [General Agg]	1.993	0.354	0.451	0.779	0.273	0.112	0.130	0.156
	DRUnet+prior both [Month Lead Agg]	0.977	0.222	0.289	0.424	0.295	0.128	0.131	0.170
	DRUnet+prior single [Month Lead Agg]	1.004	0.225	**0.288**	0.432	0.288	0.122	0.129	0.164

TABLE 1 – Land CRPS weighted by latitude for 2m temperature and 1m wind speed forecasts for different models, lead times and latitude zone. Bold : better than climatology. Underlined : best across models (excluding climatology). All means are built using the latitude weighting presented in section 5.2.

at northern latitudes than central or southern ones. As discussed in section 5.3.3, the DRUNet architecture is not performing well, although outperforming raw model predictions.

To understand the origin of these poor performances at these latitudes, we can examine the direct predictions of our models. Figure 21 shows the corrected standard deviations for 2m temperature, in day in January (left) and August (right) for EMOS. We observe that the area with the more uncertainty (yellowish values) is 90°–30°N, where it covers the entire area, followed by 30°S–90°S where the 60°S–90°S part is mostly affected, and finally 30°N–30°S with very few high standard deviations. This is consistent with our previous comments, as larger standard deviations induce larger CRPS.

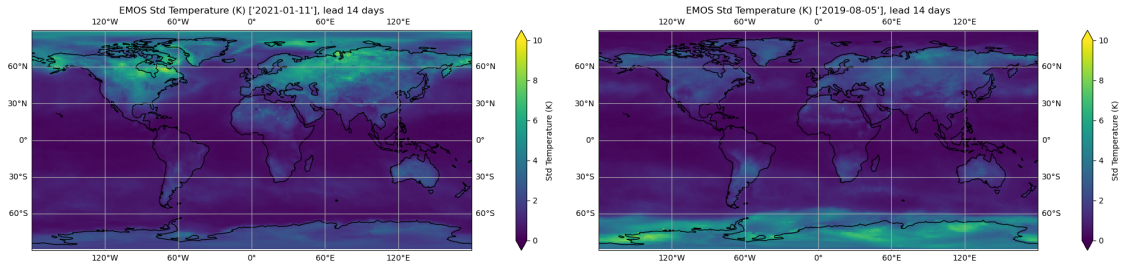


FIGURE 21 – Corrected standard deviations of 2m temperature using EMOS on the 2021-01-11 (left) and 2018-08-05 (right)

It is also interesting to notice in Figure 21 that these uncertainties are rooted in

seasonality : in winter, the northern latitude are harder to predict and have larger uncertainties (left panel) whereas during the summer, it is the South Pole that is subject to high uncertainties (right panel).

6 Conclusion and Future Work

In this study, we implemented several different post-processing methods for 2m temperature and 10m wind speed. We built reusable pipelines for WeatherBench2 data, from downloading to pre-processing, which are relatively new as the dataset was released in 2023. We also discussed and implemented normalization strategies and metrics fit to our problem and weather post-processing in general.

For subseasonal-to-seasonal post-processing, the traditional EMOS is our most successful method, improving results for both variables while DRUNet was only competitive for temperature. S2S remains a challenge, especially after 3 weeks where we do not perform significantly differently (or worse) than climatology. However, we have to remind ourselves that ERA5 from which we derive the climatology is no easy nor naive product but the result of a lot of scientific effort, in both collecting observations and assimilating them (3.1.1). Post-processing strategies remain faster methods in providing corrected predictions, which can have a strong socio-economical impact on communities.

Our experiments also shed light on essential differences between the variables we are working with. We reached similar results for temperature with EMOS and DRUNet. Our experiments showed that post-processing 2m temperature can be enhanced by jointly predicting other variables such as 10m wind speed, as suggested by Bouallègue et al [1]. This multi-objective problem will certainly need some adjustments in the training setup and we could dive into the findings on multi-task learning [34] to further improve our performances. We also observed the efficiency of mixture model approaches in this context, that we could leverage for better performance. In particular, integrating climatology when other models are too unsure about their predictions could improve them, in the spirit of "windows of opportunity" discussed by Mariotti et al. [27].

On the other hand, wind speed is revealed as strongly surface-dependent and could benefit from a better representation and treatment of the static predictors such as topography or vegetation cover. UNet-like architectures showed weaknesses in the post-processing context, the convolutional filters being too conservative of the priors, but also in the forecasting context where it does not outperform EMOS. We believe it could be related to the weight-sharing across neighbouring cells, impeding the learning of sharp changes in wind speed in the mountains. We could imagine building a topography-aware UNet architecture, by cutting some links between cells

when the topography is too different, or leveraging other deep learning architectures to this end.

We did not have time to analyse the impact of our learned temperature trends across the years, which we suspect could be critical in the quality of our predictions. Indeed, one flaw of our method is that they actually rely a lot on these trends, that we know are changing due to climate change. Our results will probably be degraded the further we look in time. This would call for better modeling of the trends, powered or not by deep learning.

In conclusion, there is still a lot to do in research for S2S post-processing, each different variable bringing its share of challenges. More approaches should be designed to fill this predictability gap in weather-to-climate forecasting, which could benefit in correctly modelling the influence of the ocean on S2S weather, such as the MJO, or taking into account soil moisture and the troposphere as pointed out by Merryfield et al. (Figure [13]).

7 Internship hindsights

This research project was done during an internship from May to October at INRIA Paris, in the AI Research for Climate Change and Environmental Sustainability (ARCHES) team. This relatively new team, that started in 2023, is slowly growing and gaining members. Being a part of it for 5 months helped me develop not only technical but also professional skills that will be useful for my career (section 7.1).

Since my gap year internship in Machine Learning for climate at the research laboratory LOCEAN IPSL in 2023, I wanted to keep working in this field. Indeed, climate change is one of the biggest challenges of our generation, and as a researcher and engineer you can help communities by predicting future climate, weather or building tools to mitigate and adapt to climate change. The other revolution of our time, deep learning, is a great tool to this end but comes with some ethical dilemmas as well (section 7.2).

7.1 Professional skills

In the 2023-2024 school year, I took the Centrale Supélec "Sciences des Données de l'Information" master, combined with the "Mathématiques Vision et Apprentissage" master from ENS Paris Saclay in order to gain technical skills in mathematics and deep learning useful for my professional project. I then wanted to put this theoretical knowledge at use, and found interest in being part of the ARCHES team of Claire Monteleoni.

Our team is multinational, with PhD students and researchers from the United States, Canada, France ... with many different backgrounds : deep learning, operational weather forecasting, physics, etc. Research is a multicultural world, it is all about sharing knowledge, both within your team and out of it and this professional experience taught me a lot about it. We had weekly seminars about papers, work of team members or outside researchers and overview of conferences people went to. We always spoke in English, which is really important to master for presenting your work to the research community, as well as for writing papers.

I learned a lot from the seminar presentations, on the scientific content as well as on how to design and deliver scientific presentations : it is important to adapt to your audience. It helped me when I presented my previous work in LOCEAN that was accepted at the Climate Change AI workshop at NeurIPS 2023 [19], to the ARCHES team and Science and Climate week 2024 organized by the LSCE IPSL in Saclay. These experiences allowed me to confront my previous work to critics and questions, which gave new ideas for research. I also gained confidence in myself and in my ability to answer questions in front of an audience, which is really important when doing research.

During the internship, I was looking for a thesis opportunity in deep learning for climate change. Thanks to the many connections of my advisor, I had the opportunity to get an interview for a Google DeepMind CIFRE. The process was a bit rushed so I did not have time to properly prepare for it, but this experience gave me an insight on the standardized recruitment protocols of big companies, perhaps for future opportunities. I also had an opportunity at LOCEAN with my previous advisor Julie Deshayes, and finally opted for their subject on hybriding deep learning and climate models.

One of the big challenges I faced during this internship was to deal with a large quantity of weather data. In school projects, the data is usually already curated and ready to use for us to focus on the methods. However in real life problems, and even more in physics, processing data is not that simple : terabytes of data, many dimensions, missing values ... Working with WeatherBench2, I learnt how to manage this large amount of data on a server (section 3.2.3), which will definitely be useful for my next research projects in deep learning. Weather forecast and its data were totally new to me and I learnt to quickly get familiar with new dataset structures. I also gained methodology in processing physical data (section section 5.1), and learnt the importance of visualizing your data regularly in the process.

All of my work was done on the INRIA server called CLEPS, as I could obviously not store my data on my computer, and I had access to CPUs and GPUs for my experiments. Unfortunately, in July, the server underwent a maintenance. It

was scheduled to last a week, but turned into 2 during which I could not run any experiments nor access the totality of my data. I had anticipated and downloaded a minimal part of my data as I was working on data processing at that time. It forced me to design new ways to work, and actually helped me reflect on the state of my project. With only several files, you dive more into the details and can unveil potential errors in your method. The unexpected second week forced me to adapt and diversify my activities apart from coding : I read papers and started to write my bibliography for this report. This incident taught me to be flexible and take time to reflect on my research, which I believe will be very important to do during my thesis.

Finally, I created bonds in a new team and work daily with all of its members. A caring and interested team is a chance and I learnt to ask for help as well as to help others. Asking questions and sharing knowledge is the best way to do research in a lab.

7.2 Ethics

One of the great paradoxes in climate informatics, especially in deep learning, is that we use computing resources such as CPUs and GPUs to train models that have a huge impact on carbon emissions : we contribute to climate change while trying to mitigate it. AI models, particularly large-scale ones, require vast amounts of energy, often powered by fossil fuels, which increases their carbon footprint. Data centers housing these models and data consume enormous amounts of electricity and water for cooling, further contributing to environmental degradation.

While these models hold immense potential to help us understand, predict, and respond to climate challenges—by forecasting extreme weather events, optimizing resource allocation, or monitoring biodiversity—they simultaneously exacerbate the very problem they aim to solve. This creates an ethical dilemma : how can we justify deploying such energy-intensive technologies in the fight against climate change without addressing their own environmental costs ?

Addressing this issue requires a push for more sustainable AI practices, and the development of smaller, energy-efficient models with high impact. During my internship, I tried to make a reasonable use of CPUs and GPUs, by designing and thinking my experiments through in order to run the ones I needed. Post-processing instead of forecasting is also a form of low cost practice, as it should require less complex models while making use of existing forecasts.

Similarly, ChatGPT [29] is a powerful tool for data scientists, but hides a very high environmental price : a request on ChatGPT needs ten times as much energy as a Google search. I try to be reasonable in its use by first searching on the in-

ternet (Wikipedia, StackOverflow, Medium), keeping it for coding tasks I already know how to do but that it will do quicker than me. It is also better to do certain things on your own, even if they take time, like literature review : this is how you understand the scope of your research and can better define your project.

I think we also need to be more transparent about the environmental impact of AI, especially when working on its use for climate change mitigation. I did not have time to estimate the CO2 emissions of this project, but many tools exist (such as the codecarbon Python package) and I know it will be required to estimate it during my future thesis at LOCEAN.

On the other hand, AI is a powerful tool with positive socio-economical impacts when used for weather and climate. The successes of AI based medium-range weather models (section 2.1.2) allow for quicker and better forecasting, which is crucial to anticipate and protect civilians during extreme weather events such as the recent Milton hurricane in Florida. Deep learning can also help reduce the costs of current climate models. Indeed, they take up to several weeks to produce forecasts, leading to high energy cost and carbon emissions. Replacing components of climate models by AI models has already been proved to accelerate inference while providing good quality forecasts, for example with NeuralGCM [22]. This hybridisation is one of the goal of my future thesis and can yield positive impacts on the costs of climate models.

Data availability WeatherBench2 dataset is available at `https://console.cloud.google.com/storage/browser/weatherbench2`.
This project is on GitHub at `https://github.com/mayajanvier/S2S_weather`.

Acknowledgements I would like to thank Claire Monteleoni and David Landry for the many opportunities and daily tutoring of this research project. I also would like to thank the ARCHES team as a whole, for always being helpful, caring and for bringing interesting topics and research insights everyday.

A Statistical Postprocessing methods

Vannistem et al. built Figure 22 by assessing the implementation difficulty of each method (amount of training data, choice of hyper-parameters, model size) as well as their adaptability (switching target variable, complexity of input-output relationship). As explained in [37], this figure is bound to evolve and is not an absolute truth but rather a first overview.

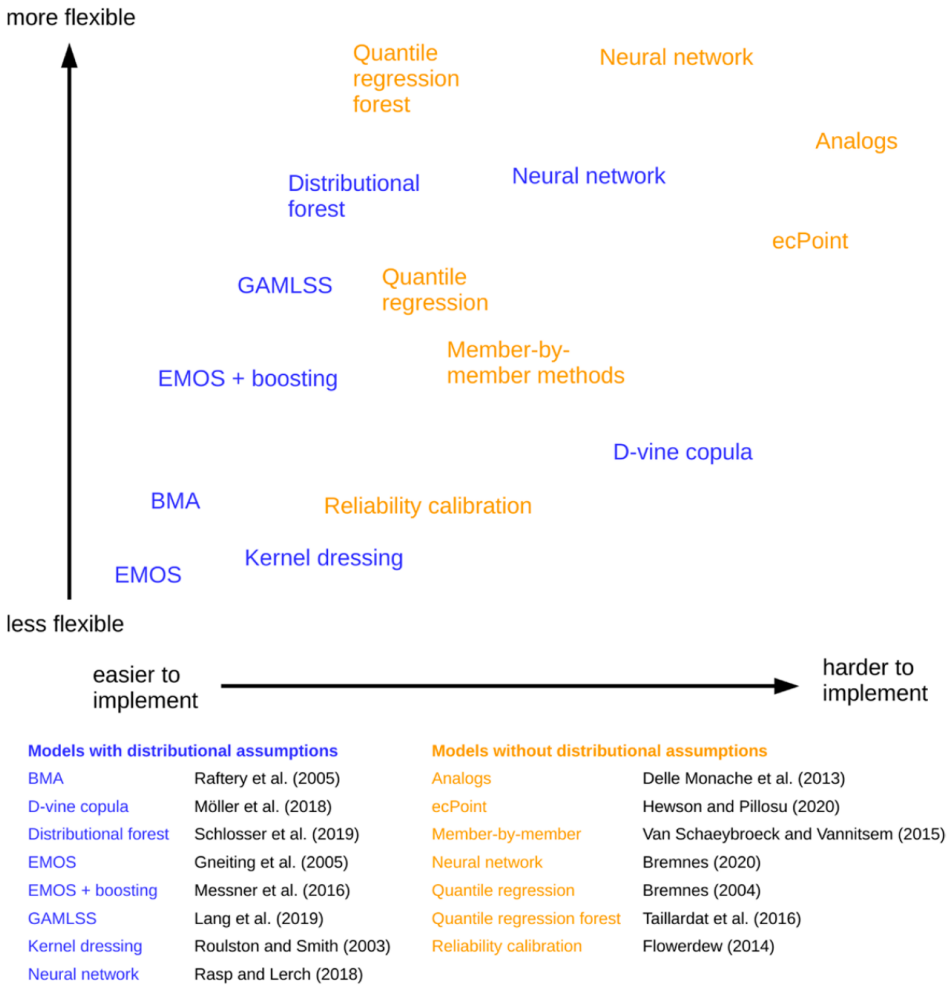


FIGURE 22 – Distribution-based and free approaches from [37]

B WeatherBench2 files and variables

The precised files we used for our study are listed below :

— ERA5 weekly : `gs://weatherbench2/datasets/era5_weekly/1959-2023_01_10-1h-240x121_equiangular_with_poles_conservative.zarr`

— IFS extended range pressure levels (train) : `gs://weatherbench2/datasets/ifs_extended_range/weekly/ifs-ext-reforecast-pressure-levels-weekly_avg.zarr`

— IFS extended range single level (train) : `gs://weatherbench2/datasets/ifs_extended_range/weekly/ifs-ext-reforecast-full-single-level-weekly_avg.zarr`

— IFS extended range pressure levels (test) : `gs://weatherbench2/datasets/`

`ifs_extended_range/weekly/ifs-ext-pressure-levels-weekly_avg.zarr`
— IFS extended range single level (test) : `gs://weatherbench2/datasets/ifs_extended_range/weekly/ifs-ext-full-single-level-weekly_avg.zarr`

Table 2 summarizes the static and temporal indicators used as input features, with the ** symbol indicating those added only to the DRUNet single+ setup.

Static indicators
Land–sea mask
Latitude*
Longitude*
Standard deviation of subgrid orography**
Vegetation cover low**
Vegetation cover high**

Temporal indicators
cos(day of year)
sin(day of year)
Forecast lead time

TABLE 2 – Static (left) and Temporal (right) indicators used as input features. * indicates additional ones for DRUnet architectures, ** indicates the additional ones for DRUNet single+

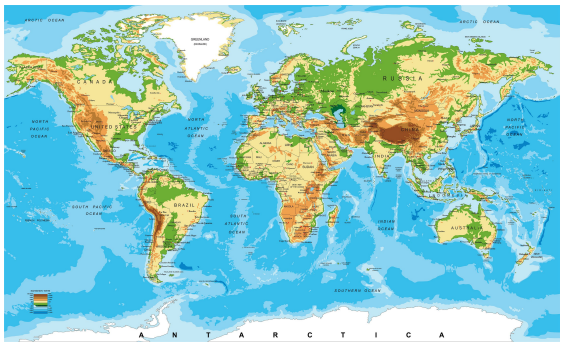
Table 3 summarizes the NWP predictors used in our models, indicating whether they come from single or pressure levels. Those are defined over 10 pressure levels (10, 50, 100, 200, 300, 500, 700, 850, 925, 1000 hPa) so each variable produces 10 independant features, except for specific humidity where the 10, 50, and 100 hPa levels only had NaN values. The standard deviation (* symbol) of our target variables are only added for DRUNet architectures. We also only compute wind speed from u and v for the 1000 hPa as it is the closest to 10m wind speed.

NWP predictors	Level(s)
2-m temperature	single
Min 2-m temperature	single
Max 2-m temperature	single
2-m dewpoint temperature	single
Convective available potential energy	single
Sea ice cover	single
Sea surface temperature	single
Skin temperature	single
Snow albedo	single
Snow density	single
Snow depth	single
Soil moisture top 100cm	single
Soil moisture top 20cm	single
Soil temperature top 100cm	single
Soil temperature top 20cm	single
Total cloud cover	single
Total column water	single
Total precipitation	single
Specific humidity	pressure (7)
Temperature	pressure (10)
U component of wind	pressure (10)
V component of wind	pressure (10)
Geopotential	pressure (10)
10m wind speed	pressure (1)
Standard deviation of 2-m temperature*	single
Standard deviation of 10m wind speed*	pressure (1)

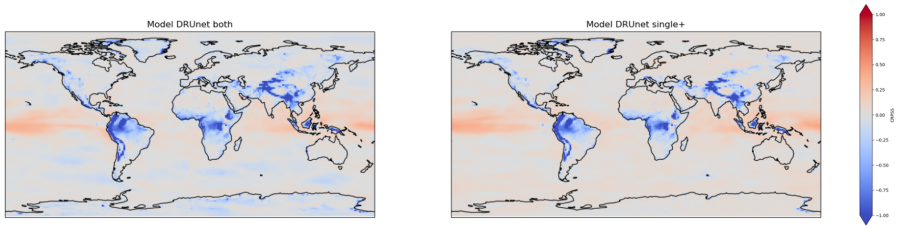
TABLE 3 – NWP predictors used for as input for our models and their file origin (single level or pressure levels). * indicates the additional predictors in DRUNet architectures.

C Surface type and wind speed

We see in Figure 23 that the negative CRPSS areas of DRUNet both (b) correspond to elevation changes in topography (a), around mountains (Andes, Himalaya) or lakes (Congo-Rwanda/Burundi), whereas the model performs better on more homogeneous areas like Northern Africa or Australia. Adding orography and vegetation cover in DRUNnet single+ (b, left) did improve the results on low lands, but the difficulties in high altitudes remain. This confirms our guess discussed in section 5.3.3 on the UNet disadvantage compared to grid-point wise strategies like EMOS.



(a) Topography of the Earth, with the colorbar ranging from green for lowland areas to brown for high mountains.



(b) CRPSS of wind speed predictions from DRUNet both (left) and DRUNet single+ (right)

FIGURE 23 – (a) Topography of the Earth and (b) DRUNet both (left), DRUNet single+ (right) CRPSS for wind speed predictions.

Références

[1] Zied Ben Bouallègue, Fenwick Cooper, Matthew Chantry, Peter Düben, Peter Bechtold, and Irina Sandu. Statistical Modeling of 2-m Temperature and 10-m Wind Speed Forecast Errors. *Monthly Weather Review*, 151, 01 2023.

[2] Xie L.-Zhang H. et al. Bi, K. Accurate medium-range global weather forecasting with 3D neural networks. *Nature 619, 533–538*, 2023.

[3] Zied Ben Bouallègue, Jonathan A. Weyn, Mariana C. A. Clare, Jesper Dramsch, Peter Dueben, and Matthew Chantry. Improving medium-range ensemble weather forecasts with hierarchical ensemble transformers. *Artificial Intelligence for the Earth Systems*, 3(1) :e230027, 2024.

[4] John Bjørnar Bremnes. Probabilistic forecasts of precipitation in terms of quantiles using NWP model output. *Monthly Weather Review*, 132(1) :338 – 347, 2004.

[5] John Bjørnar Bremnes. Ensemble postprocessing using quantile function regression based on neural networks and bernstein polynomials. *Monthly Weather Review*, 148(1) :403 – 414, 2020.

[6] Taco S. Cohen, Mario Geiger, Jonas Koehler, and Max Welling. Spherical cnns, 2018.

[7] Copernicus. Climate reanalysis. *https://climate.copernicus.eu/climate-reanalysis*.

[8] Copernicus and ECMWF. ERA5 documentation. *https://confluence.ecmwf.int/display/CKB/ERA5%3A+data+documentation*.

[9] Philippe Courtier and Olivier Talagrand. Variational assimilation of meteorological observations with the direct and adjoint shallow-water equations. *Tellus A : Dynamic Meteorology and Oceanography*, Jan 1990.

[10] Thépaut J.-N. Courtier, P. and A. Hollingsworth. A strategy for operational implementation of 4d-var, using an incremental approach. *Quarterly Journal of the Royal Meteorological Society*, 120 :1367–1387, 1994.

[11] Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5) :313–318, 2012.

[12] F. Vitart et al. Outcomes of the WMO Prize Challenge to improve Subseasonal to Seasonal predictions using Artificial Intelligence. *Bulletin of the American Meteorological Society*, 103(12), 2022.

[13] William J. Merryfield et al. Current and emerging developments in subseasonal to decadal prediction. *Bulletin of the American Meteorological Society*, 101(6) :E869 – E896, 2020.

[14] European Centre for Medium-Range Weather Forecasts (ECMWF). NWP models. *https://www.ecmwf.int/en/forecasts*.

[15] Yun Fan, Vladimir Krasnopolsky, Huug Dool, Chung-Yu Wu, and Jon Gottschalck. Using Artificial Neural Networks to Improve CFS Week 3-4 Precipitation and 2-meter air Temperature Forecasts. *Weather and Forecasting*, 38, 01 2021.

[16] Itamar Faran. CRPS — A Scoring Function for Bayesian Machine Learning Models. *Towards Data Science, https://towardsdatascience.com*, 2023.

[17] Tilmann Gneiting, Adrian E. Raftery, Anton H. Westveld, and Tom Goldman. Calibrated Probabilistic Forecasting using Ensemble Model Output Statistics and Minimum CRPS Estimation. *Monthly Weather Review*, 133(5) :1098 – 1118, 2005.

[18] Nina Horat and Sebastian Lerch. Deep learning for post-processing global probabilistic forecasts on sub-seasonal time scales, 2023.

[19] Maya Janvier, Redouane Lguensat, Julie Deshayes, Aurélien Quiquet, Didier Roche, and V. Balaji. Surrogate modeling based history matching for an earth system model of intermediate complexity. In *NeurIPS 2023 Workshop on Tackling Climate Change with Machine Learning*, 2023.

[20] Nathaniel Johnson, Dan Collins, Steven Feldstein, Michelle L'Heureux, and Emily Riddle. Skillful wintertime north american temperature forecasts out to 4 weeks based on the state of enso and the mjo*. *Weather and Forecasting*, 29 :23–38, 02 2014.

[21] Ryan Keisler. Forecasting global weather with graph neural networks, 2022.

[22] Dmitrii Kochkov, Janni Yuval, Ian Langmore, Peter Norgaard, Jamie Smith, Griffin Mooers, Milan Klöwer, James Lottes, Stephan Rasp, Peter Düben, Sam Hatfield, Peter Battaglia, Alvaro Sanchez-Gonzalez, Matthew Willson, Michael P. Brenner, and Stephan Hoyer. Neural General Circulation Models for Weather and Climate. *Nature*, July 2024.

[23] Roger Koenker and Gilbert Bassett. Regression quantiles. *Econometrica*, 46 :pp. 33–50, 1978.

[24] 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. Learning skillful medium-range global weather forecasting. *Science*, 382(6677) :1416–1421, 2023.

[25] David Landry, Anastase Charantonis, and Claire Monteleoni. Leveraging deterministic weather forecasts for in-situ probabilistic temperature predictions via deep learning, 2024.

[26] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer : Hierarchical vision transformer using shifted windows, 2021.

[27] Annarita Mariotti, Cory Baggett, Elizabeth Barnes, Emily Becker, Amy Butler, Dan Collins, Paul Dirmeyer, L. Ferranti, Nathaniel Johnson, Jeanine Jones, Ben Kirtman, Andrea Lang, Andrea Molod, Matthew Newman, Andrew Robertson, Siegfried Schubert, Duane Waliser, and John Albers. Windows of opportunity for skillful forecasts subseasonal to seasonal and beyond. *Bulletin of the American Meteorological Society*, 101, 01 2020.

[28] Soukayna Mouatadid, Paulo Orenstein, Genevieve Flaspohler, Judah Cohen, Miruna Oprescu, Ernest Fraenkel, and Lester Mackey. Adaptive bias correction for improved subseasonal forecasting. *Nature Communications*, 14(1) :3482, 2023.

[29] OpenAI. Chatgpt. `https://www.openai.com/chatgpt`, 2024. Accessed : 2024-10-11.

[30] Kathy Pegion, Ben Kirtman, Emily Becker, Dan Collins, Emerson LaJoie, Robert Burgman, Ray Bell, Timothy Delsole, Dughong Min, Yuejian Zhu, Wei Li, Eric Sinsky, Hong Guan, Jon Gottschalck, E. Metzger, Neil Barton, Deepthi

Achuthavarier, Jelena Marshak, Randal Koster, and Hyemi Kim. The Subseasonal Experiment (SubX) : A Multimodel Subseasonal Prediction Experiment. *Bulletin of the American Meteorological Society*, 100, 07 2019.

[31] Romain Pic, Clement Dombry, Philippe Naveau, and Maxime Taillardat. Distributional Regression U-Nets for the Postprocessing of Precipitation Ensemble Forecasts, 07 2024.

[32] 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.

[33] 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.

[34] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization, 2019.

[35] Maxime Taillardat, Olivier Mestre, Michaël Zamo, and Philippe Naveau. Calibrated ensemble forecasts using quantile regression forests and ensemble model output statistics. *Monthly Weather Review*, 144(6) :2375 – 2393, 2016.

[36] Tippett MK Pegion K. Trenary L, DelSole T. Monthly ENSO Forecast Skill and Lagged Ensemble Size. *J Adv Model Earth Syst.*, 2018.

[37] S. Vannitsem, John Bjørnar Bremnes, Jonathan Demaeyer, Gavin Evans, Jonathan Flowerdew, Stephan Hemri, Sebastian Lerch, Nigel Roberts, Susanne Theis, Aitor Atencia, Zied Ben Bouallègue, Jonas Bhend, Markus Dabernig, Lesley De Cruz, Leila Hieta, Olivier Mestre, Lionel Moret, Iris Odak Plenkovic, Maurice Schmeits, and Jussi Ylhäisi. Statistical Postprocessing for Weather Forecasts – Review, Challenges and Avenues in a Big Data World. *Bulletin of the American Meteorological Society*, 102 :1–44, 11 2020.

[38] Jonathan A. Weyn, Dale R. Durran, Rich Caruana, and Nathaniel Cresswell-Clay. Sub-Seasonal Forecasting with a Large Ensemble of Deep-Learning Weather Prediction Models. *Journal of Advances in Modeling Earth Systems*, 13(7), July 2021.

[39] C. et al. White. Advances in the application and utility of subseasonal-to-seasonal predictions. *Bulletin of the American Meteorological Society*, 103 :E1448–E1472, 2022.

[40] Wikipedia. Finite element method. *https://en.wikipedia.org/wiki/Finite_element_method*.

[41] Baicen Xiao. Strategies for Balancing Multiple Loss Functions in Deep Learning. *Medium*, 2024.