Probabilisitic Seasonal Forecasts with Conditional Variational AutoEncoders

Alejandro Peraza González
Supervisors: Amanda Cardoso Duarte[*] and Lluís Palma[*]

2024
May

[*]Barcelona Supercomputing Center

Acknowledgements

I offer my sincerest gratitude to my partner, family, and friends for their constant support. To the directors of my thesis for the guidance and knowledge provided. Finally, I want to thank the professors and companions from UPC for the expertise I gained throughout the Master.

Contents

List of Figures

List of Tables

Glossary

Boundary condition Conditions that need to be satisfied in the solution of a differential equation. Common boundary conditions for the atmosphere are that the velocity component normal to the earth's surface vanishes and that the individual derivative of pressure vanishes at the upper surface [1] .

Ensemble forecast Collection of simulations that conform to a probabilistic prediction. Each simulation of the ensemble forecast is known as a Ensemble member. Several members are needed to capture the signal of the chaotic nature of the climate system. The climate model is run many times from very slightly different initial conditions. Often physics of the model is also slightly perturbed, and some ensembles use more than one model within the ensemble (multi-model EPS) or the same model but with different combinations of physical parameterization schemes (multi-physics EPS) [2] .

Ensemble member Forecast simulation. Several members conform an ensemble forecast .

General Circulation Models (GCM) Series of processes to predict future atmospheric conditions by solving dynamics and physics equations that explain the movements and changes of the atmosphere. Also known as climate or dynamical models .

Geopotential height "The height of a given point in the atmosphere in units proportional to the potential energy of unit mass (geopotential) at this height relative to sea level." [3] .

Modes of variability "Natural variability of the climate system, in particular on seasonal and longer time scales, predominantly occurs with preferred spatial patterns and time scales, through the dynamical characteristics of the atmospheric circulation and interactions with the land and ocean surfaces" [4] .

Operational prediction "They are short-term forecasts at a detailed level (e.g., SKU) used to drive production schedules, transfer of goods in the distribution network, procurement of materials required to meet schedules, etc. The planning horizon used for operational forecasting varies with the lead time required to execute supply planning." [5] .

Acronyms

AE AutoEncoder.

AUPRC Area Under the Precision-Recall Curve.

cβVAE Final model.

CDF Cumulative Distribution Function.

CLIM Climatology.

CMIP Coupled Model Intercomparison Project.

CMIP6 Coupled Model Intercomparison Project version 6.

CRPS Continuous Ranked Probability Score.

cVAE Conditional Variational AutoEncoder.

cVAE0 Deep learning baseline.

ECMWF European Center for Medium-range Weather Forecasts.

ELBO Evidence Lower Bound.

ENSO El Niño-Southern Oscillation.

ERA5 fifth generation of ECMWF reanalysis for the global climate and weather.

GCM General Circulation Model.

GPU Graphical Processing Unit.

JJA June, July, August.

KL Kullback-Leibler.

LOESS Locally estimated scatterplot smoothing.

ML Machine Learning.

MSE Mean Squared Error.

netCDF Network Common Data Form.

PCA Principal Component Analysis.

PERS	Persistence.
ReLU	Rectified Linear Unit.
SEAS5	ECMWF's fifth generation seasonal forecast system.
SGD	Stochastic Gradient Descent.
SST	Sea Surface Temperature.
t-SNE	t-distributed stochastic neighbor embedding.
TAS	Temperature 2-meter Above Surface.
VAE	Variational AutoEncoder.
VI	Variational inference.
WCRP	World Climate Research Programme.
ZG500	500 hectopascal geopotential height.

Notation

Random variable X

Realizations of a random variable x

Marginal probability distribution of a random variable $p(X = x)$ with shorthand notation as $p(x)$

Conditional probability distribution $p(X = x|Z = z)$

Joint probability distribution $p(X, Z)$

Expectation $\mathbb{E}(\cdot)$

Kullback-Leibler Divergence $D_{KL}(\cdot||\cdot)$

Normal distribution parameters μ as the mean and σ as the standard deviation

Abstract

While traditional numerical models are the primary source of seasonal predictions, they are computationally expensive to run and induce drifts and biases in their outputs. Machine learning offers an alternative by learning complex patterns from data, potentially overcoming these limitations. This project investigates the use of generative neural networks to create computationally efficient seasonal forecast models. These models can incorporate randomness (stochasticity) and generate ensemble forecasts, mimicking the capabilities of established methods at a lower cost. We implement a Conditional Variational AutoEncoder (cVAE) approach, previously successful in six-month predictions. Here, we focus on predicting summer's average temperature (three months) based on May's climate data. The network learns from data generated by the sixth Coupled Model Intercomparison Project (CMIP) and is evaluated against real-world observations from the fifth generation of ECMWF reanalysis for the global climate and weather (ERA5) reanalysis dataset. To isolate the method's performance, we employ detrending, e.g. removing long-term trends from the data using a reference period independent of the testing data. Finally, we compare our model's predictions to established baselines commonly used in seasonal forecasting for a comprehensive evaluation. Our approach managed to deliver seasonal forecasts that matched the performance of SEAS5, the leading model, across various regions. Although the general proficiency of our model falls short of SEAS5, it surpasses standard benchmarks like climatology and persistence. Consequently, our predictions provide a rapid and computation-light estimate of the seasonal climate, retaining the skillfulness from numerical methods to a certain degree.

1 Introduction

Weather forecasting becomes increasingly challenging beyond a two-week timeframe, primarily due to the stochasticity nature of the atmosphere [6]. However, certain components of the climate system, such as the land surface and the ocean, can force the atmosphere dynamics extending the predictability limit, to a certain extent, beyond two weeks. Predicting the earth's system state beyond two weeks is known as *climate prediction*, and is designed to bridge the gap between weather forecasting and climate projections.

Climate predictions are generally classified into three categories: subseasonal (2-12 weeks), seasonal (1-15 months), and decadal (1-30 years). Beyond the decadal forecasts, we have *climate projections*, that range from 20 to 100 years. These different timescale predictions are important to make decisions at different levels in the decision-making process. In this work, we focus specifically on the *seasonal timescale*. This type of prediction has demonstrated high importance in the decision-making of different socio-economical sectors such as hydrology, agriculture, or economy [7–10].

Current approaches to seasonal forecasting rely on General Circulation Models (GCM) [11,12]. These models are designed to simulate the physical processes that govern the climate. The numerical equations of the process involved in the climate system are known but can only be solved with discrete numerical methods. Since representing the Earth in a continuous space is computationally infeasible, scientists adopted a grid system. This allows us to apply equations, but it also means that processes smaller than the grid cells are not accurately represented. The phenomena that cannot be represented due to the grid resolution are approximated by parameterizations that differ between GCMs. Every GCM uses the same equations, but the parameterizations allow them to model the induced biases and drift differently, aiming to reflect the actual climate behavior more accurately.

On the other hand, recent Machine Learning (ML) advancements in different fields, such as computer vision, natural language processing, and weather forecasting [13–19] show how this approach can identify patterns and relationships that traditional modeling may miss. This unique ability offers a high potential for improving seasonal prediction. For example, ML algorithms can identify hidden correlations between ocean temperatures, the atmospheric state, or climate events like El Niño-Southern Oscillation (ENSO), leading to more accurate predictions [20–22]. Still, the application of ML is yet to be further explored in this context, as the literature available is relatively short.

In this thesis, our objective is to explore the use of Variational AutoEncoders to produce global seasonal predictions of mean temperature. Variational AutoEncoders (VAEs) are used in this context because they are powerful tools for capturing complex patterns in data. VAEs also provide a probabilistic framework, introducing uncertainty in the predictions, which is crucial for reliable seasonal forecasting. We aim to assess their efficacy in improving prediction accuracy, and their reliability in predicting real-world large-scale climate data, while also proposing a more cost-effective method for obtaining these forecasts. Through a series of experiments and analysis, we seek to highlight the potential advantages and challenges of this approach.

The following sections of this thesis are organized as follows: Section 2 presents a more in-depth explanation of seasonal predictions, general circulation models, along with a review of the related work focused on machine learning. Section 3 describes the methodology employed, covering the method, the data, and the metrics used in the development of this work. Section 4 presents the results obtained as the outcomes of the experiments conveyed as well as a comparison with other existing approaches. Additionally, includes a discussion about the project's accomplishments, and the limitations that we encounter. Finally, section 5 presents the conclusion of this thesis and potential avenues for future work.

2 Background and related work

2.1 Seasonal prediction

Seasonal forecasting plays a crucial role in decision-making across various scenarios, including the estimation of energy consumption, preparation for extreme events, and crop yield predictions. However, deterministic forecasts alone cannot capture the stochasticity and complexity inherent in the climate system. Thus, providing reliable probabilistic information and addressing the associated uncertainties is critical. The effectiveness of seasonal forecasts is evaluated based on their skill [2] – e.g. how accurately the forecasts reflect the behavior of the climate system. Before any forecasts reach users, a rigorous assessment of the model's skill is performed to ensure they meet user needs and maintain trust.

To generate a forecast for a specific period, it is very important to first understand the factors that are influencing it. In figure 1, we can observe the contribution of three different sources (atmosphere, land, and ocean) to the general weather and climate predictability for the forecast time. We can observe that the predictability of the atmosphere decreases significantly after two weeks, which is the main reason for the limited accuracy of weather forecasts beyond this timeframe. On the other hand, the predictability coming from the ocean decreases more gradually, due to the slow and persistent changes occurring over weeks or even months in the ocean, therefore playing an important role in seasonal forecasts [23].

One example of the ocean's influence on seasonal predictability is the El Niño-Southern Oscillation (ENSO). This event is characterized by strong deviations in the eastern and central equatorial Pacific sea surface temperature that repeat, on average, every four years. The pattern that repeats over the years oscillates between two opposing phases: El Niño, which refers to the warming of the central equatorial Pacific, and La Niña, the opposing cold phase. Although ENSO occurs in the tropical Pacific, it impacts not only its surroundings but also the global climate. These remote influences are also known as teleconnections, and can occur with other modes of variability. It is important to provide skillful seasonal predictions to capture the teleconnections affecting the target variable, region, and time scale.

2.2 General circulation models

General Circulation Models (GCM), also known as climate or dynamical models, are the main tool to perform climate predictions nowadays. GCMs simulate different climate and physical processes (eg. rain, carbon cycle, rivers,

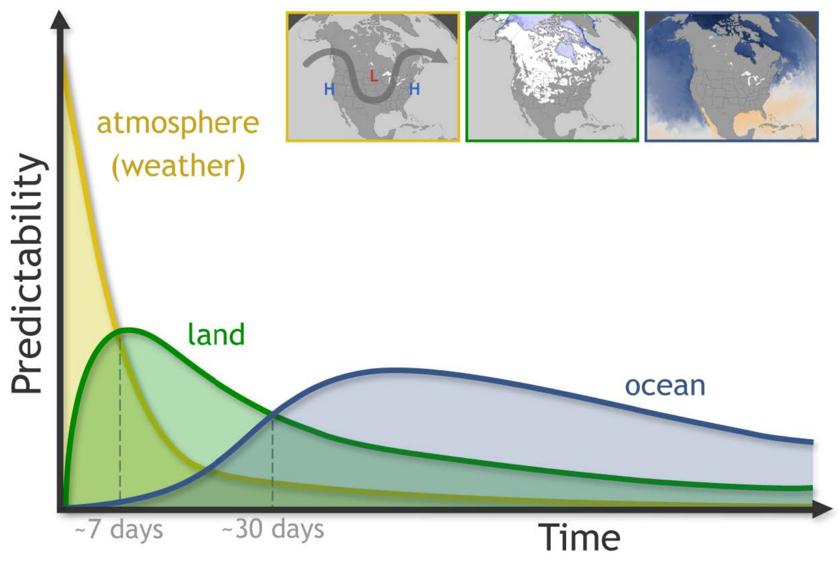


Figure 1: Impact of atmosphere, land, and ocean on predictability as a function of lead time [24]. The predictability inherent in the atmosphere decreases rapidly due to its chaotic nature, while processes related to the land are valuable for producing short to medium-range forecasts. For seasonal predictions, the main source of predictability is the ocean since changes in the oceans are slow and persistent over weeks or months.

oceans, etc) via numerical methods. The climate system is discretized into a spatiotemporal grid of the entire globe. For each time step (e.g. every 6 hours) and grid section, the equations are resolved numerically, consequently requiring high computational resources. To manage the high computational demand, both the spatial and temporal resolution of the climate models are restricted, along with the number of ensemble members generated for a ensemble forecast. Climate models are used for different applications that range from research to operational prediction. A key difference among these models is whether they are initialized or forced by boundary conditions.

When climate models are initialized, they incorporate observations of the Earth's current state as their starting point, being later executed for the desired duration. This procedure is important when the initial state of the system is key to predicting its target state i.e. sub-seasonal (next month), seasonal (next 6 months), and decal (next 10 years) predictions. As an example, the fifth seasonal forecasting system SEAS5 [25] from the European Center for Medium-range Weather Forecasts (ECMWF) is an initialized dynamical model that runs at the start of every month to predict the next 3 seasons. This model is considered the state-of-the-art for most seasonal prediction applications.

Contrary to initialized models, forced models don't assimilate the current conditions of the Earth system. They start from a control run in which the model is executed until reaching a stable state, known as a spin-up (or also called warm-up) period. After the control run, the simulation is run imposing boundary conditions (mostly radiative forcings). These models are useful when impacts due to changes in boundary conditions are way more important than initial conditions i.e. long-term climate projections. The Coupled Model Intercomparison Project (CMIP) from World Climate Research Programme (WCRP) provides in each iteration a set of forced climate models from different institutions that perform climate projections. In each CMIP iteration (the latest being CMIP6), different models are run several times under common green-houses scenarios to capture plausible climate change trends until the end of the century. Even though climate models are not used to provide seasonal predictions, they are a big source of data about the climate system that can be used to train Machine Learning models.

2.3 Autoencoders, Variational AutoEncoders and Conditional Variational AutoEncoders

AutoEncoders (AEs) [26–28] are a type of artificial neural network designed to identify efficient encodings of unlabeled data, typically used for reducing the data's dimensionality or to extract features from the data. Formally, these neural networks seek to learn two functions [29, 30]: an encoder function $E_\phi : \mathbb{R}^n \to \mathbb{R}^p$ and a decoder function $D_\theta : \mathbb{R}^p \to \mathbb{R}^n$, where n is the dimensionality of the input space and p is the dimensionality of the encoded space, generally being $p < n$.

The encoder function E_ϕ maps an input $x \in \mathbb{R}^n$ to a reduced representation $y \in \mathbb{R}^p$. The decoder function D_θ aims to reconstruct the original input from this reduced representation, producing $\hat{x} \in \mathbb{R}^n$, such that they satisfy:

$$arg \min_{\phi,\theta} \mathbb{E}[\Delta(x, D_\theta(E_\phi(x)))]$$

where $\mathbb{E}$ is the expectation over the distribution of the input data, and Δ is the reconstruction loss function, which measures the distance between the decoder output and the input, often the l_2-norm or Mean Squared Error (MSE).

Considering its characteristics, the Encoder segment of the architecture shows similarities to various dimensionality reduction methods. If the encoder and decoder segments execute linear operations while non-linear operations are excluded, the autoencoder would achieve a latent representation similar to a Principal Component Analysis (PCA). Consequently, an autoencoder can be described as a generalization of PCA, with the capability of not only discovering a low-dimensional hyperplane where the data reside but also identifying a non-linear manifold [30].

However, it is important to note that the latent space representation of these networks does not follow any probability distribution. Consequently, the output of autoencoders primarily consists of reconstructions rather than attempts to generate new data. This characteristic is evident in the design of the loss function, which focuses on reconstruction accuracy, without incorporating elements that would encourage the generation of new data.

To address this issue, *Variational AutoEncoder (VAE)* [31] have been introduced. These neural networks are designed to learn the probability distribution of the input data, enabling them to generate unseen samples. To accomplish this, VAEs are structured so that their latent space follows a specific and predefined probability distribution.

Given an input data $X = \{x_1, x_2, ..., x_n\}$, the objective of generative models like Variational AutoEncoders is to estimate the marginal distribution $p(X = x)$[1]. This approach assumes that X represents a random variable influenced by an underlying latent variable Z, which dictates its distribution. With that in mind, Z will follow a known probability distribution such as a Gaussian. Subsequently, a parametric function p_θ is defined, with θ adjusting this distribution to more closely resemble $p(x)$. In practice, a neural network is utilized to learn and optimize the set of parameters θ.

Directly estimating the posterior distribution $p_\theta(z|x)$ is computationally infeasible. By following Bayes' theorem [32] it would require integrating over Z to obtain the marginal of X. Thus, instead of directly calculating the posterior distribution, VAEs rely on Variational Inference (VI) to approximate it [33]. Specifically, amortized stochastic VI is used. Here, the term "amortized" means that the encoder is trained to learn the distribution of the latent variable Z, allowing the decoder to focus on a generation task. "Stochastic" refers to using mini-batches instead of the entire training data at once.

To apply VI, we consider the prior distribution $p_\theta(z)$, its likelihood $p_\theta(x|z)$, and the unknown posterior distribution $p_\theta(z|x)$. In addition, we consider ψ as the variational parameters that adjust the distribution of the selected family of distributions (Gaussian). The probabilistic encoder will learn the variational parameters ψ to alleviate the learning process of the decoder. The loss function considered is known as the Evidence Lower Bound (ELBO) which acts as a lower bound for the "evidence" $p_\theta(x)$. By minimizing the -ELBO, we obtain an estimated posterior predictive distribution that converges to a locally optimal maximum likelihood estimation.

[1]We will use p(x) as a shorthand notation for p(X=x)

The ELBO, defined in equation (1), captures two key aspects of a Variational AutoEncoder. The first term, called the reconstruction term, is the expected log-likelihood, which measures how well the model reconstructs the input data. The second term, the Kullback-Leibler (KL) divergence [34] quantifies the difference between the model's estimate of the latent variable's distribution (posterior) and a predefined distribution (prior). The KL divergence acts as a regularizer, encouraging the model's latent space to resemble the prior while still capturing the necessary information for reconstruction. Thus, by having both terms the VAE aims at generating samples that are similar to the input data, while also learning a meaningful latent variable that defines the creation of new data.

$$\mathcal{L}(\psi|\theta, x) = \mathbb{E}_{q_\psi(z)}[\log p_\theta(x|z)] - D_{KL}(q_\psi(z)||p_\theta(z)) \tag{1}$$

While VAEs offer speed and unique properties for climate forecasting, a limitation is their need for paired input-target data. *Conditional Variational AutoEncoder (cVAE)* addresses this by allowing targets to be conditioned on additional inputs. CVAEs share the same fundamental purpose as VAEs, but the probability distributions approximated during the training process are shaped by this extra conditioning information available.

2.4 Related work

General Circulation Models (GCMs) face significant drawbacks, including uncertainties associated with the initial conditions, the high computational cost, and the significant drifts and biases in their predictions. A major source of uncertainty is the diverse and unevenly distributed set of observations that define the current climate state. This uncertainty propagates quickly as the simulations run. Additionally, the big computational cost of GCMs imposes a reduction in the spatial and temporal resolution of climate simulations. Thus, key physical processes (i.e. cloud dynamics) are not explicitly resolved by the numerical equations and need to be parametrized (approximated). These parameterizations introduce biases and errors, often leading to inaccurate simulations of teleconnections, which are crucial for seasonal prediction [35].

In this scenario, Machine Learning (ML) offers a promising approach to overcoming the limitations faced by traditional dynamical models. Its ability to learn complex, non-linear patterns and its low computational cost for predictions, make it a compelling tool for climate prediction (in inference time). ML's success in fields like computer vision, Natural Language Processing (NLP), and healthcare has triggered interest among climate scientists, with several applications to weather forecasting already proposed. These include, foundational models [36–38], transformer-based models [16–19, 39], graph neural networks [13], diffusion models [14] and hybrid physics-ML models [15].

One of the key challenges when it comes to applying ML to seasonal forecasting is the limited availability of training data. Most existing observational climate datasets began in the 1950s, offering only around 70 to 100 data points relevant to the predictions (focusing on interannual fluctuations) if only observational records are used. This number is significantly less than the millions of samples typically required by deep learning algorithms, posing a significant bottleneck for applying ML in these cases. Training on climate model simulations from CMIP has shown the potential to deal with this limitation.

Akin to advances in weather predictions, recent approaches using machine learning have emerged in the field of seasonal forecasting. Gibson et al. [40] explored the potential of training machine learning models on climate simulations (e.g. the output of the CMIP climate model) to improve seasonal precipitation forecasts. For that, they use 40 CMIP5 historical simulations of 1920-2005 leading to a total of ∼3400 years. As the predictors, PCA was applied on each month of the data to extract key features from input variables (e.g., Sea Surface Temperature (SST), precipitation). For the targets, the proposed approach opts for K-means clustering of precipitation anomalies over the western US. Two separate seasonal predictions were considered: November through January and January through March. With these inputs and targets, multiple ML models are trained and their generalization is tested with historical reanalyses (e.g. ERA5 and others) showing how their deterministic forecasts outperform climate models from the North American Multi-Model at predicting precipitation in the western US.

Andersson et al. [41] proposed an approach to seasonal sea ice forecasting called IceNet. The model was trained on historical simulations and climate projections from two CMIP6 models from 1850-2100 (with 2200 years in total) and tested using reanalysis data from ERA5 from 1979-2011 to forecast the next 6 months of monthly-averaged sea ice concentration maps (of different classes) at 25 km resolution. Using 25-member ensemble forecasts, this model advanced the range of accurate sea ice forecasts, outperforming a state-of-the-art dynamical model (ECMWF SEAS5) in seasonal forecasts of summer sea ice, particularly for extreme sea ice events.

Ling et al. [42] focus on improving the skill at predicting the interannual variability Indian Ocean Dipole (IOD) as the existing predictions were limited to three months ahead. For that, they propose a multi-task learning model named MLT-NET. The model was first trained on historical simulations of a wide range of CMIP5 and CMIP6 climate models (∼2000 years) and tested with reanalysis data from 1983 to 2019. The resulting 10-member ensemble predictions indicate that the model can predict the IOD up to 7 months of lead time while outperforming state-of-the-art methods.

Pan et al. [43] proposes an approach to the seasonal forecasting task using a sample size that is an order of magnitude larger. This is achieved by considering historical simulations as well as climate projections from CMIP5 and CMIP6 climate models (30 models in total) having a total of 52.201 years of simulations. The model was trained using both historical simulations and climate projections while reserving the last 30 years of the historical period to test the model. Entity embeddings (learned numerical representations) are used as a way of representing the different biases in the climate models which allows the trained model to potentially produce seasonal forecasts that mimic their behavior. Contrary to previously covered works, where reanalysis data (from ERA5) was used to fine-tune the entire network, it was used here to create a fine-tuned embedding such that the embeddings closer to it are from models better suited for the corresponding prediction task. One of the key problems of considering multiple climate models is that the different biases are not corrected by giving them average weight in the training but the use of the embeddings to account for their differences can help in compensating those biases and benefit from the existing data.

Seeking to address limitations of existing methods, especially the restricted ensemble sizes, Pan et al. [43] used a Conditional Variational AutoEncoder (cVAE) as a probabilistic method to obtain ensemble forecasts for 6-months averaged with two months of lead time. The use of CVAEs allows for a quick inference time and thus, the capability of producing large ensemble forecasts. The proposed cVAE architecture utilizes residual blocks of convolutions to encode information from the upper ocean thermal state (at 8 different height levels) to predict monthly averages of Temperature 2-meter Above Surface (TAS) and precipitation. The data was standardized point-wise independent of the simulations. The same was applied to the target after taking its monthly average. The results demonstrate that the proposed method excels at model-world forecasts of a specific climate model (e.g. CanESM), where model-world refers to predictions made using the learned embedding that corresponds to this climate model. Additionally, they do real-world forecasts by considering reanalysis data from the Ocean Reanalysis System 5 (ORAS5) were used as predictors to predict precipitation (from the Global Precipitation Climatology Project) and temperature (from ERA5). Using the predictions conditioned on the embedding of a specific model for real-world forecasts, they achieve comparable performance to a dynamic seasonal forecasting model. Finally, the model is benchmarked on the North American Multi-Model Ensemble (NMME). To do so, they create two 300-member ensemble forecasts, one where each climate model has the same influence and another using the fine-tuned embedding on ERA5 data. The results show that both approaches achieve similar performance results compared to the NMME ensemble forecast with a significantly reduced computational time.

This project stands out for its ability to generate global probabilistic predictions, similar to state-of-the-art ensembles from dynamical models utilized by operational centers worldwide. A key feature of this model is its use of "entity embeddings", which captures the unique dynamics present in different CMIP models and reveals their similarities and differences in representing climate behavior. Furthermore, the proposed architecture allows for fine-tuning using reanalysis data like ERA5, offering the flexibility to adjust specific components or the entire model. This fine-tuning process can be either complete or partial by only fine-tuning a specific module, such as the entity embeddings. Additionally, it allows for tailoring the model to specific tasks by using select subsets of climate models and facilitates experimentation with various predictors, targets, and climate models. By leveraging insights from climate model projections, the project overcomes limitations associated with traditional forced models, ultimately aiming to produce insightful seasonal forecasts.

3 Methodology

This section presents the methodology implemented in this work, inspired by [43]. The goal of this thesis is to test the applicability of the proposed model in predicting a 3-month average seasonal forecast with different lead times. Therefore, we implement a conditional Variational AutoEncoder that allows conditioning the desired target on different variables. During the implementation of the methodology, we customized it specifically for 3-month predictions. This involved applying a combination of architectural and data modifications.

Among these modifications, we implement the proposed cVAE methodology tuning it to be a cβVAE to overcome posterior collapse issues caused by the increment in complexity of the task. In terms of the variables considered, we swap the upper ocean thermal state with Sea Surface Temperature (SST) and include Temperature 2-meter Above Surface (TAS) and 500 hectopascal geopotential height (ZG500) as predictands to predict TAS of 3-month averaged predictions. As for the climate models considered, we use a relatively small number of CMIP6 models for accessibility reasons. From them, we only use historical simulations (with a large number of ensembles) for the training of the model as we consider that climate projections can be far from real-world forecasts. We split the available ensemble members of each climate model to create the training and validation sets. Similarly to their approach, we use reanalysis data from ERA5 to create a fine-tuned embedding, but, we only use a reduced period of it. With the remaining years from the reanalysis, we test the generalization of the model as in other mentioned works.

In the following sections, we present the implementation of our method (section 3.1), the data used (section 3.2), and the evaluation metrics considered in this work (section 3.3).

3.1 Method

Here, we utilize $\mathbf{X}$ to represent the predictor variables, $\mathbf{Y}$ for the predictand variables, and $\mathbf{M}$ for the indices of the General Circulation Model (GCM). The implemented Conditional Variational AutoEncoder (cVAE) aims to estimate the probability of a target variable $y \in Y$ occurring given specific predictor variables $x \in X$ and GCM indices $m \in M$. This is written mathematically as the conditional probability distribution $p(y|x, m)$.

To achieve this, the cVAE introduces a latent variable, Z, which is assumed to follow a normal distribution defined by a specific mean (μ) and standard deviation (σ). This latent variable is independent of the original input data $(\mathbf{X}, \mathbf{M})$ [44]. During the prediction stage, when the target variable is unknown, the model assumes a standard normal distribution for Z. Additionally, the cVAE utilizes a parametric function, denoted as $p_\theta(\mathbf{y}|\mathbf{x},\mathbf{m})$, parameterized by θ, to approximate the true target distribution as closely as possible.

To approximate $p_\theta(y|x,m)$ we use the Variational inference (VI) framework as explained in section 2.3. This involves a probabilistic encoder (q_ϕ), modeled by a neural network, that approximates the posterior distribution of a latent variable Z. Additionally, a probabilistic decoder (p_ψ), also a neural network, approximates the conditional likelihood $p_\theta(y|z,x,m)$. Both the encoder and decoder are assumed to be Gaussian distributions parameterized by ϕ and ψ respectively.

Then, we can formulate the Evidence Lower Bound (ELBO) or $\mathcal{L}$ as was shown in equation (1) obtaining the equation:

$$logp_\theta(y_i|x_i,m_i) \geq \mathbb{E}_z(\log p_\psi) - D_{KL}(q_\phi||p(z|x_i,m_i)) = \mathcal{L} \tag{2}$$

where the ELBO $\mathcal{L}$ acts as a lower bound for the log-likelihood of the predictand given the predictors and model index $logp_\theta(y_i|x_i,m_i)$. The ELBO itself is formed by the two terms previously mentioned, where the reconstruction and the regularization terms. The intuition is the same as in section 2.3.

To train the network using Stochastic Gradient Descent (SGD), backpropagation needs to be employed to efficiently calculate the gradients needed for parameter updates. However, directly calculating the gradient concerning a latent variable like Z, which is inherently stochastic, is impossible. To address this challenge, we leverage the properties of Gaussian distributions. We introduce a new variable ϵ, following a standard normal distribution $N(0,\mathbb{I})$ so that $z = \mu + \sigma * \epsilon$. This reparametrization of the Gaussian distribution is known as **reparametrization trick** [31]. This transformation ensures that Z retains the desired Gaussian distribution$z$ $\mathbb{N}(\mu,\sigma)$ while allowing for gradient calculations as ϵ is no longer stochastic. It is important to note that SGD is an optimization technique for minimizations, and in this context, we aim to minimize the negative evidence lower bound instead of maximizing it.

While the reparametrization trick enables the gradient computation, an analytical form of the loss function is still required. The loss function consists of two terms: a reconstruction term, measured using Mean Squared Error (MSE), and a regularization term penalizing the divergence between the approximate posterior and the prior distribution of the latent variable.

Our goal is to find a closed-form solution (analytical solution) for the Kullback-Leibler (KL) divergence, $D_{KL}(q_\phi||p(z|x_i,m_i))$, between two Gaussian distributions. Here, q_ϕ is assumed to be a Gaussian distribution with parameters μ_ϕ and Σ_ϕ, while $p(z|x_i, m_i)$ represents a standard normal distribution. Fortunately, since Gaussian distributions belong to the exponential family, a closed-form expression for the KL divergence between two Gaussians exists, as shown below:

$$\begin{aligned} D_{KL}(q_\phi||p(z|x_i, m_i)) &= D_{KL}(\mathcal{N}(\mu_\phi, \Sigma_\phi)||\mathcal{N}(0, \mathbb{I})) \\ &= \frac{1}{2}\left(tr(\Sigma_\phi) + \mu_\phi^T \mu_\phi - k - ln|\Sigma_\phi|\right) \end{aligned} \tag{3}$$

Once the model is trained, the inference can be done on the posterior predictive distribution obtained by sampling z and using the log-likelihood function learned by the decoder. During inference, the model does not have access to the target and is not able to sample z from the encoder. Instead, z is sampled from $p(z|x, m)$ which is assumed to be a standard Gaussian.

This method offers significant flexibility in training models suited for different sets of variables, lead times, timescales, and seasons between others. But the benefits extend beyond that. VAEs are inherently designed to learn meaningful representations of the data. This learned representation, similar to how principal components capture most of the variance, should translate to more informative data for forecasting. Furthermore, unlike many VAE applications, the generative network directly produces the desired forecasts, eliminating the need for additional downstream tasks that might introduce bias. Finally, the fast inference allows for generating large ensembles of forecasts, which can be computationally challenging with other deep learning methods like diffusion models.

3.1.1 Architecture

Figure 2 illustrates the architecture used in our model, which contains two separate encoders. One encoder focuses on processing the predictor variables (X), while the other handles the predictand (Y). These encoded representations are then combined with embeddings to create a unified representation within the latent space. Finally, the decoder uses samples from this latent space to generate seasonal forecasts.

The encoders and the decoder are composed of residual and transposed residual blocks respectively [45]. Residual blocks are used here to reduce the input size during encoding (down-sampling) and compress the important information in the data. In contrast, transposed residual blocks are essential for expanding the data (up-sampling) in the decoder. This expansion aims to decompress the information necessary to create accurate predictions.

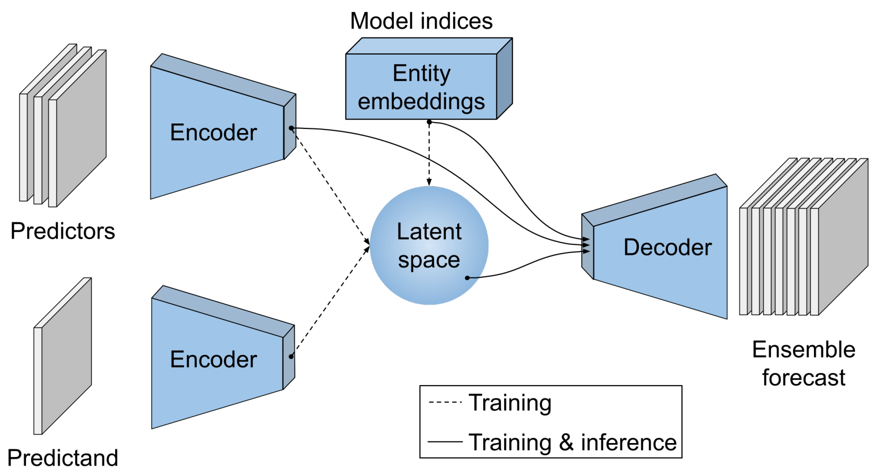
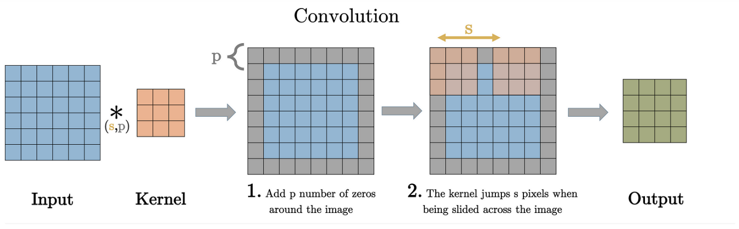


Figure 2: **Model overview:** Our model contains two encoding processes. First, separate encoders compress the information from the predictor variables (X) and the predictand (Y). Second, embeddings are created based on additional information (model indices). These embeddings, along with the encoded data from both X and Y, are then combined to form a unified representation in the latent space. Finally, the decoder utilizes random samples from the latent space, together with the encoded predictors and predictand, to generate multiple seasonal forecasts, forming an ensemble forecast.

These blocks are composed of several core operations, such as convolutions and transposed convolutions, max-pooling, batch-normalization, Rectified Linear Unit (ReLU), and dropout, which are explained below.

Convolutions are matrix operations in which a kernel (filter) slides across the input applying element-wise multiplication and summing its results into the output, as it is shown in figure 3. Convolutions depend mainly on two parameters, the padding (p), which refers to the the number of rows and columns of zeros added around the input, and the strides (s), which indicate the number of columns the filter moves after every operation. Convolutions are generally purposed to down-sample the data.

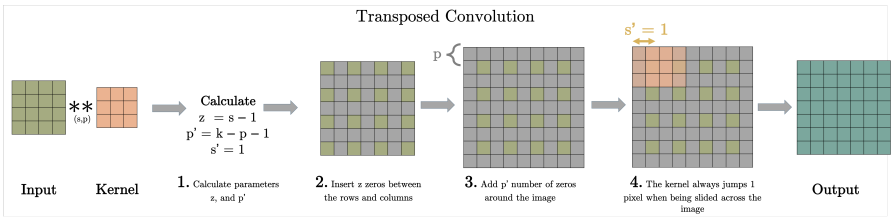
Transposed convolutions, on the other hand, work differently than standard convolutions. They are designed for upsampling. However, it is important to note that they differ from deconvolutions, which aim for a perfect reversal of the convolution operation. Figure 4 shows a step-by-step illustration of the transposed convolution operation. If we consider strides of 2 and padding of 1 pixel on all sides, we can compute z, p', and s as shown in figure 4. A larger matrix is created by adding z zeros between the input elements and p' rows around the matrix. Later, a standard convolution with strides of $s' = 1$



Convolution
p
s
∗
(s,p)
Input
Kernel
1. Add p number of zeros around the image
2. The kernel jumps s pixels when being slided across the image
Output

Figure 3: Illustration of the **convolution** operation [46]

is applied to this expanded matrix producing the output of the transposed convolution operation as a result. This process, as shown in the figure, effectively doubles the output size compared to the original input. In our architecture, we used transposed convolutions of strides = 2 and padding of either 1 or 0 (depending on preserving the original shape) to consistently achieve upsampling by a factor of 2.



Transposed Convolution
∗∗
(s,p)
Calculate
z = s − 1
p' = k − p − 1
s' = 1
p
s' = 1
Input
Kernel
1. Calculate parameters z, and p'
2. Insert z zeros between the rows and columns
3. Add p' number of zeros around the image
4. The kernel always jumps 1 pixel when being slided across the image
Output

Figure 4: Illustration of a **transposed convolution** operation [46]. Transposed convolutions are used as up-sampling operations. The explanation considers s=2, k=3, p=1 and as such, z=1 and p'=1

Similar to convolutions, *max-pooling* utilizes a sliding kernel that iterates across the input data. However, unlike convolutions that perform element-wise multiplication and summation, max-pooling identifies the maximum value within the kernel region and uses that as the output value.

Batch Normalization improves training efficiency. For this, each batch of data goes through normalization which involves subtracting the mean and dividing by the standard deviation of the batch. Thus, the output will be a linear regression with a learnable slope and intercept.

For the activation function, we used *ReLU*. This activation function adds non-linearity to the network. It simply outputs the input value if it's positive, and zero otherwise ($f(x) = max(0, x)$).

Dropout is a technique used to help prevent overfitting. It randomly deletes a certain proportion of neurons during training. By doing this, it “turns off” some neurons with a specific probability, forcing the network to learn more robust features.

With these core operations in mind, we can delve into the structure of **residual blocks** used in the encoders. Residual blocks (illustrated in figure 5) leverage "shortcut connections" to achieve the functionality of $F(x) + x$, where x represents the input data and $F(x)$ refers to the output of the stack of convolutions (showing at the top of figure 5). These connections (represented at the bottom of figure 5), as described in [47–49], bypass one or more layers in the network. In our implementation, the shortcut connections are applied as a convolution with a $1x1$ kernel (equivalent to a linear layer), adding the linearly transformed input directly to the output of the stacked convolutional layers. This not only preserves valuable information throughout the block but also potentially mitigates vanishing gradients.

The main operations of these blocks are convolutions, used here to learn the spatial features of the data. Every convolution uses a stride and padding as one (and everything else is set as the default implementation of PyTorch) which preserves the input shape. The residual blocks rely on a sequence of stacked convolutions. Each convolution is followed by batch normalization and a ReLU activation function. However, the final convolution in the block deviates slightly. Here, the outputs from the "main" and "residual" paths (refer to figure 5) are aggregated together before applying the final ReLU activation. After the last activation function, max-pooling is applied to downsample the data by half.

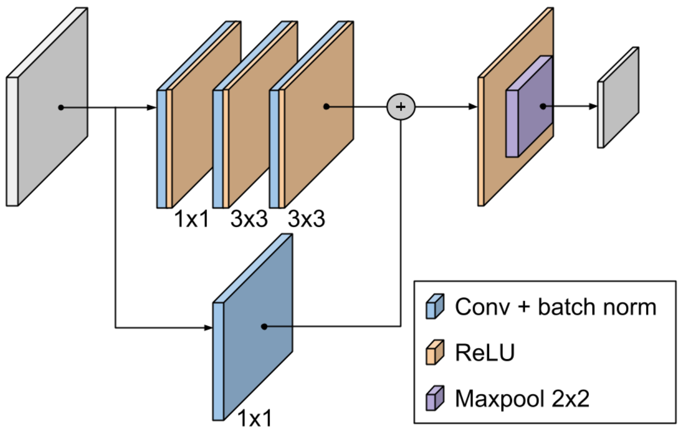


Figure 5: **Structure of a residual block**. The input is processed through two different paths: the "main path" that contains three convolutional layers and the "residual path" with a single convolutional layer. Each convolution is followed by batch normalization to improve training stability and a ReLU activation function to introduce non-linearity. The final ReLU is applied after the summation of the main and residual outputs. Finally, a max-pooling is applied to reduce the input shape by half.

In the decoder, we employed a series of **transposed residual blocks**. The structure of this block is illustrated in figure 6. While similar to the structure of the residual blocks, these blocks utilize transposed convolutions and omit the max-pooling operation at the end. Each path contains a transposed convolution with strides of 2, which upsamples the input shape by a factor of two, effectively doubling the shape after a transposed convolution is applied.

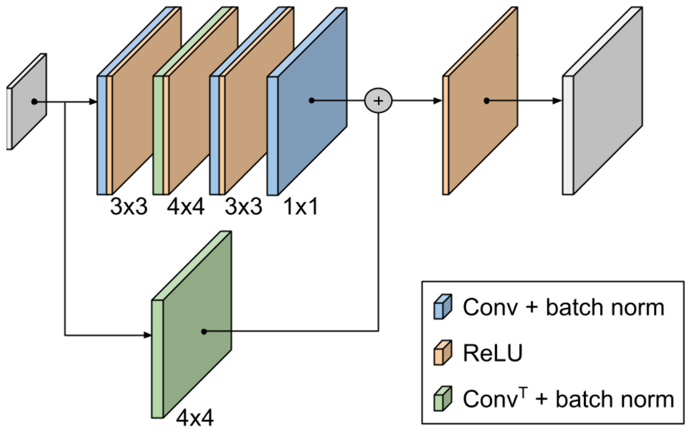


Figure 6: **Structure of a transposed residual block.** Similarly to the residual blocks, we used a skip connection to pass the input information to the following layers as a residual. Transposed convolutions are applied to double the shape of the input allowing decompression of the information every time a transposed residual block is applied. After every convolution and transposed convolution the batch is normalized and the ReLU activation function is applied where the last activation is applied after adding the result of the two paths.

In addition to the encoders and the decoder, we also apply a **entity embedding** [50] of the climate models. We refer to entity embeddings as a numerical codification assigned to each climate model. To do so, the index of the corresponding model is first one-hot encoded. The one-hot encoding consists of creating an array of zeros as long as the number of categories and assigning a 1 for the current category. Then, a linear layer is applied creating a numerical representation for each one-hot encoded index. On top of the one-hot encoding and the linear layer, a sigmoid activation function is used such that the outputs are in the range of [-1,1]. We use the embeddings to learn a numerical representation that is different from the climate models considered.

To gain a clearer understanding of the latent space, we analyze the connections between the various components of the encoders, the decoder, and the entity embedding block. As previously discussed, the latent space requires the extraction of the mean and standard deviation that characterize the latent

Gaussian distribution. Figure 7 illustrates the process where the outputs of the encoders are flattened and concatenated with the embeddings. This concatenated vector is later fed into two distinct linear layers to derive the mean and standard deviation, which define the latent distribution. To maintain a differentiable computational graph, we apply a reparametrization trick [31], where ϵ is sampled from a $N(0, 1)$ distribution, and z is computed as $z = \mu + \sigma * \epsilon$. Finally, the output is reshaped to a format compatible with the decoder.

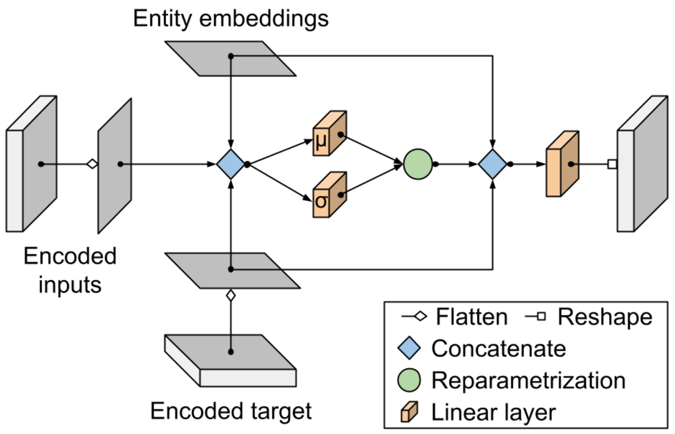


Figure 7: Structure of the components involved to learn the mean and standard deviation used for the latent space Z. First, the encoded inputs and target are flattened at concatenated together along with the entity embeddings. With that, two linear layers are applied, one for the mean (μ) and the other for the standard deviation (σ) of z since we assumed it to be Gaussian. Then, we apply the parametrization trick $z = \mu + \sigma * \epsilon$ sampling ϵ from a standard Gaussian and concatenate the output with the encoded inputs and target again. Finally, we apply a linear layer and reshape the data to match the input shape of the decoder.

A detailed overview of the final architecture is presented in figure 8. The final architecture contains two encoders consisting of residual blocks and a decoder composed of transposed residual blocks. Each residual or transposed residual block applied, doubles or halves the shape of the data, respectively, where the output channels of the convolutions and transposed convolutions of each block are defined in figure 8. Note that the processes of flattening, reshaping, and the linear layers involved in the latent space are not detailed here, but are explained in figure 7. After passing through the transposed convolution blocks of the decoder, the final step is to apply a convolution that restores the target's original shape.

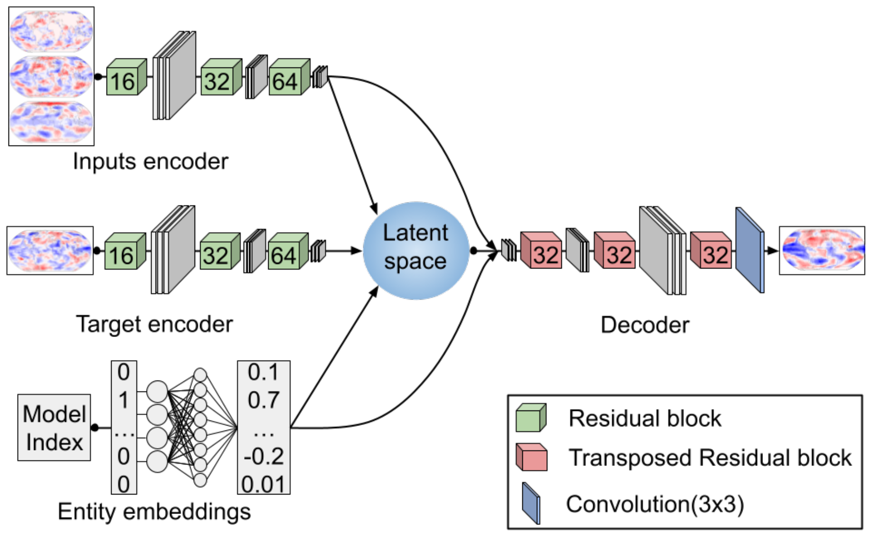


Figure 8: **Complete overview of the final architecture**. On the left, we can find the two encoders and the entity embedding block. The encoders use residual blocks where the numbers in the blocks represent the number of channels in each of its convolutions. The model indices are one-hot encoded and then passed through a fully connected layer to obtain the entity embeddings. The encoded inputs and target along with the entity embeddings are used to create the latent space (see figure 7 for details). Samples from the latent space along with the encoded input and the entity embeddings are fed into the decoder. Contrary to the encoders, the decoder uses transposed residual blocks where the number of the transposed residual blocks refers to the channels considered. Lastly, a final convolution is applied to restore the shape of the targets. Selecting a random year, we show the inputs (TAS, SST, and ZG500 from May), the target (TAS from summer), and the mean ensemble forecast of a 256 ensemble forecast.

3.1.2 Extension to Beta-VAE

While training the method for 3-month predictions we encountered instability issues in the loss function [51, 52] leading to posterior collapse. Notably, these problems were not mentioned in [43] potentially because we approached a more complex task. To stabilize the training and mitigate the risk of posterior collapse, we explored the use of Beta-VAEs [53], in which the KL divergence term of the Evidence Lower Bound (ELBO) is adjusted using a hyperparameter. This was considered since the mean squared error and the Kullback-Leibler divergence were on different scales during the training causing the MSE (used for the reconstruction term) to not be considered less in the loss reducing the focus of reconstruction. An extended discussion of this imbalance in the loss function can be seen in [54, 55].

Fu et al. [56] have demonstrated the effectiveness of KL divergence annealing in improving the training process. This technique involves cyclically varying the weight of the KL divergence loss term between a starting and ending value throughout training. The weight increases from 0 to 1 and repeats this cycle as many times as the pre-defined number of cycles. A defined monotonic function (e.g., linear, sigmoid, or tanh) dictates the weight's growth pattern, where a proportion adjusts the weight's growth speed in each cycle. Annealing the KL divergence term allows the model to give more importance to the reconstruction term at the start of each cycle. Annealing the weight of the KL divergence encourages the importance of the reconstruction term in the loss function. Thanks to that, the training becomes more stable and less propense to posterior collapse.

3.2 Data

The model was trained with climate model data from the Coupled Model Intercomparison Project version 6 (CMIP6). Table 1 shows a summary of the models used and the total number of members available for each of them. Due to the imbalanced amount of simulations available from each climate model (with some of them containing less than 10 simulations available), a stratified training and validation split was employed. This strategy ensures that each climate model is represented in both the training and validation sets. Specifically, 70% of the simulations from each model were allocated to the training set, while the remaining 30% were used for validation. This approach mitigates potential biases arising from the uneven distribution of simulations across climate models.

Dynamical model	Members
ACCESS-CM2	3
CNRM-CM6-1	29
CanESM5	25
EC-Earth3	8
MIROC-ES2L	30
MIROC6	50
NorCPM1	30

Table 1: Dynamical models used for this project. All of them come from CMIP6

The climate model data is stored in Network Common Data Form (netCDF) files, a format suitable for multidimensional arrays and metadata. The data has a spatial resolution of 5° x 5° and covers the period from 1850 to 2013. In this project, we used historical simulations instead of climate projections. These simulations offer a more reliable foundation for the model compared to future projections due to their closer alignment with observed data, reducing the potential for drift. The data is standardized using the selected reference period from 1950 to 1985 (further explained in section 4.2.1). To perform standardization, we obtain the trend of this period using Locally estimated scatterplot smoothing (LOESS) [57] and its standard deviation. Grid-point wise predictors and predictand were subtracted from their trend and divided by the standard deviation. The predictand was standardized after obtaining the month-average of the selected months in the prediction. For both predictors and predictand, each simulation was standardized independently.

More specifically, the following predictors were used: Sea Surface Temperature (SST) since the ocean is the highest predictability source for seasonal predictions, the Temperature 2-meter Above Surface (TAS) for land representation, and the 500 hectopascal geopotential height (ZG500) for atmospheric representation. These input variables were selected as a reduced set of variables that could provide the highest predictability for seasonal predictions. The selection was guided by the collaboration of climate scientists. This combination of three variables provides compreensive information on the state of the land, ocean, and atmosphere, all corresponding to May averages. For the predictand, we used the three-month average temperature (TAS) for June, July, August (JJA)

3.3 Evaluation Metrics

Based on related research and especially the suggestions from [58], we selected metrics to fulfill three main purposes: measure the skillfulness of the predictions, assess the quality of the ensemble forecasts, and test how well we can predict extreme events.

To measure the skillfulness of the predictions, we use the Pearson correlation coefficient [59] as defined in equation (4). The correlation is computed between the mean forecasts (x) and observations (y) as a measure of association between them. Additionally, the statistical bias is used to measure how the mean value of our forecasts differs from the true value of our estimation (observations) and it is computed as the difference between the mean forecast and mean observation. We consider the bias between the mean of observations and the mean of the mean ensemble forecasts (note that it's the bias between means and not the mean of biases). For interpretability, we compute the negative of the bias so that a positive bias represents that the forecasts are warmer than the observations and the opposite for negative values. Both the correlation and bias are used as deterministic comparisons and their focus is not on the probabilistic properties of the forecast.

$$r_{x,y} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}} \tag{4}$$

Continuous Ranked Probability Score (CRPS) was used to assess the quality of the ensemble forecasts produced. We can understand the CRPS as "a quadratic measure of the difference between the forecast Cumulative Distribution Function (CDF) and the empirical CDF of the observation" [60]. Equation (5) shows how the CRPS is computed where F represents the true CDF and the indicator function checking for $x \geq y$ would serve as the empirical CDF. We compare the empirical CDFs from the observations (as the true CDF) and the mean forecasts. CRPS will have values in the [0,1] where 1 implies that the forecasts are accurate for the observations considered whereas a value of 0 is the opposite, therefore, a higher CRPS is desirable. When we refer to the Continuous Ranked Probability Skill Score (CRPSS), the CRPS is computed using climatology as a reference. It can be computed as shown in equation (6). As such, positive values can be interpreted as the forecasts being more accurate than climatology, and the opposite for negative values.

$$CRPS(F, y) = \int (F(x) - 1_{\{x \geq y\}})^2 dx \tag{5}$$

$$CRPSS = 1 - \frac{CRPS_{forecast}}{CRPS_{climatology}} \tag{6}$$

To test the prediction of extreme events, we use the Area Under the Precision-Recall Curve (AUPRC). This is computed as in equation (7) considering different classification settings that define how to calculate the precision (see equation (8)) and recall (see equation (9)). For that purpose, four different scenarios will be considered where the classifications will focus on predicting correctly the forecasts at ±1 and ±2 standard deviations. In each scenario, the positive will be considered events that are in the target range (for example, +1 standard deviations away from the mean, 0). The higher the *auprc*, the better the predictions are in terms of precision and recall for the classification scenario.

$$AUPRC = \sum_{n}(R_N - R_{n-1})P_n \tag{7}$$

$$P = \frac{T_p}{T_p + F_p} \quad (8) \qquad R = \frac{T_p}{T_p + F_n} \quad (9)$$

4 Results

This section presents a methodology implemented in this thesis inspired by the model presented in [43] but specifically tailored for 3-month seasonal predictions, by considering a β-Variational AutoEncoder. We begin by outlining the experimental setup, including detailed implementation choices. This is followed by a description of the ablation studies conducted to refine and develop the final model. Finally, we present the final results evaluating the performance of the 3-month predictions generated by our model compared to the established benchmarks.

4.1 Experiment setup

Here, we delve into the key components of an experiment, conceptualizing experiments as the act of training either a single model or a collection of models. First, we will cover the task definition used to focalize the results in a certain prediction. Then, we will briefly describe how the metrics will be visualized for a forecast and to perform forecast comparisons. Next, the benchmarks used to compare the final results. Finally, a comprehensive explanation of the method's implementation and the training procedures.

Task definition. We define a specific task based on a 3-month prediction, where we focus on forecasting Temperature 2-meter Above Surface (TAS) using monthly-averaged inputs from May to predict the average of June, July, August (JJA) of the same year. We changed the prediction objective compared to [43], to make the prediction task closer to what is done in operational weather and climate services providers [35]. Figure 9 illustrates an example forecast for the summer of 1986 generated using the first embedding which corresponds to the ACCESS-CM2 model.

Metric visualizations Considering a certain forecast, we visualize the metrics as a map, however, this complicates the process of comparing between two forecasts. For that, we use scorecards. We refer to scorecards as heatmaps showing the relative difference between two forecasts, where one is considered the reference. Each row of the heatmap represents an evaluation metric, and each column represents a geographical region (see figure 10). The color maps of the heatmap are limited to [-1, 1] to enhance visual clarity and facilitate the comparisons within the scorecards. As an illustration, figure 20 shows the first time where we use a scorecard to compare between forecasts.

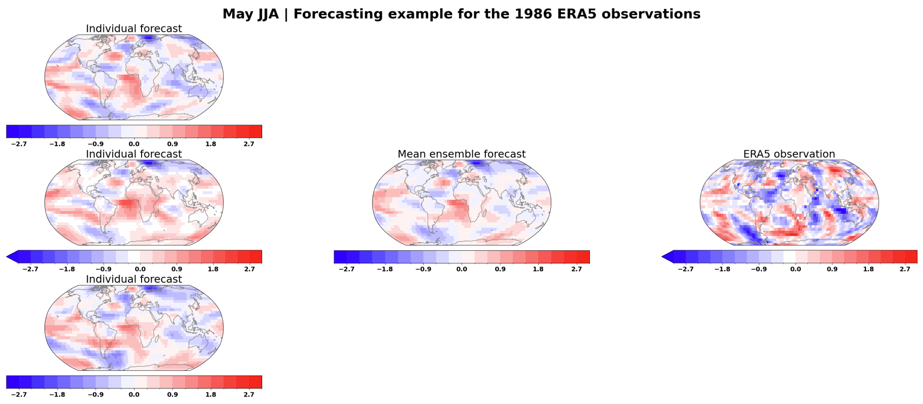


Figure 9: Example forecasts predicting the summer of 1986 using ERA5 as inputs. The first column contains individual predictions, the second column contains the mean ensemble forecast and the last column presents the ERA5 observation

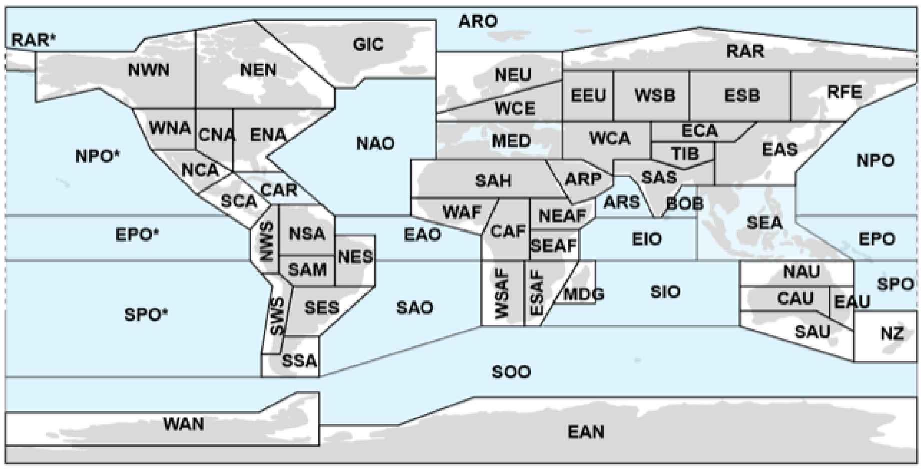


Figure 10: Illustration of the IPCC climate reference regions for subcontinental analysis of climate model data [61]

Benchmarks. The performance of our model is evaluated using ERA5 for the period 1986-2020 against several baselines. The simplest baselines include Climatology (CLIM) and Persistence (PERS), representing the average historical value and the previous season's value, respectively. Additionally, we compare against a Deep learning baseline (cVAE0) that uses an initial configuration and ECMWF's fifth generation seasonal forecast system (SEAS5), the current state-of-the-art dynamical model for seasonal predictions.

The *climatology* (CLIM) baseline represents the average scenario from the climate perspective. It is calculated by averaging the target variable (e.g., TAS) over a chosen reference period (here we choose 1950-1985), essentially representing the expected value without considering any specific year. This approach generates an ensemble forecast composed of all historical possibilities within the reference period. Importantly, forecasts using climatology are independent of lead time as they simply reflect the long-term average. As a consequence, the mean ensemble forecast of the climatology will be approximately null, reflecting the averaging of detrended (i.e., having the linear trend removed) historical values.

The *persistence model* (PERS), represents a simple yet informative baseline. It assumes that the upcoming climate conditions will remain unchanged from the initial state of the forecast.For instance, if a JJA (June-July-August) summer prediction is issued in May (lead time of two months), the predicted JJA average TAS (near-surface air temperature) would simply be the observed TAS from May. This essentially replicates the observed value with a one-to-one mapping, functioning as an identity function.

The *deep learning baseline* (cVAE0) uses the basic configuration file defined in listing 1. Note that the Kullback-Leibler divergence scheduler is not considered and the *kl_weight* is set to 1, thus, the deep learning baseline would be a conditional VAE instead of a conditional Beta-VAE. At each lead time, the same configuration is used to predict summer.

Source Code 1: Fixed hyperparameters after hyperparameter tuning

```
model:
  num_in_features: 6
  embedding_size: 100
  z_size: 1000
  first_in_channel: 32

trainer:
  lr: 1.e-4
  kl_weight: 1.e-1
  kl_ratio: 0.5
```

The *seasonal forecasting climate model* (SEAS5) is a state-of-the-art dynamical seasonal forecasting model. Differently from our model, which focuses on 3-month predictions, SEAS5 natively predicts the following 6 months using data from the day it is initialized, i.e. as we are predicting JJA, it would use information from July 1st.

Implementation details. The code was implemented in Python using PyTorch to define the network and Xarray to load and manipulate Network Common Data Form (netCDF) files. The reference paper does not provide code, thus, it was designed from scratch taking inspiration and suggestions from other works like [13]. The implementation is flexible enough to support multi-month predictions and the use of any set of variables at any desired region, not only globally. The latter allows running forecasts on a certain region of interest, like Europe, using global inputs or even from another region as long as every input comes from the same grid.

The model is trained on a single Graphical Processing Unit (GPU) NVIDIA V100 from the CTE-POWER cluster of MareNostrum4 taking 40 ± 5 minutes. In every experiment where we train the model, we test different learning rates in the range[1.e-5, 1.e-3] as suggested in [62]. To manage multiple trainings, we use Autosubmit [63], a Python-based workflow manager that can launch and monitor tasks.

4.2 Ablation studies

Numerous factors can influence the performance of the results, and it is not feasible to address them all. In this thesis, we will conduct the following ablation studies to help us determine the final model and achieve the results presented in section 4.3. First, we explore the standardization process that can be used to overcome the biases in the input data. Then, we study the impact of the ensemble size to decide which size to use on further tasks. Afterward, we tune the hyperparameters to optimize the results. Finally, we use ERA5 (1950-1985) to fine-tune the entity embedding layer and obtain a fine-tuned embedding that could help understand which climate models were more appropriate for a prediction task.

4.2.1 Extracting the trend from the data

The purpose of seasonal predictions is frequently to predict how warmer or colder a season is compared to a reference. In that regard, the interest is more on the anomalies around the trend than the trend itself. With that in mind, we consider detrending the data and standardizing it against the arbitrarily decided reference period 1950-1985, which we do not use for model testing. Figure 11 shows the members of 4 climate models (data from 1850-2013) as well as the ERA5 observations (from 1950-2013). The different simulations of the climate models are represented with thin continuous lines. The figure shows in discontinuous lines, the detrended data using Locally estimated scatterplot smoothing (LOESS), a local regression method used to smooth time series data. It can be seen that different climate model data contain different biases regarding the observational data. To deal with those biases, we extract the trends obtained respectively using loess for ERA5. However, to perform this process with CMIP6 data it is computationally infeasible. Instead, we consider the trend as the ensemble mean by year. The standard deviation used to standardize the data is obtained from the reference period previously mentioned.

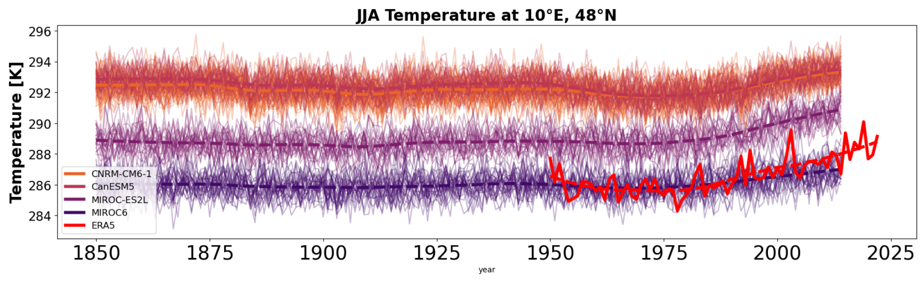


Figure 11: Plot showing CMIP6 and ERA5 data before the standardization process. We show only four climate models described in the legend for visualization purposes. The thinnest lines represent the different simulations associated with the model of their color. In thicker continuous lines, the ensemble mean by year is shown and the discontinuous ones are the time series obtained from applying loess. The bias between the climate model data and ERA5 data varies depending on the model before the trend is removed

Figure 12 shows the effect of the standardization of both types of data. The reference period is highlighted in grey, from which the trend is computed in its respective way and it is applied to the rest of the data. Detrending the data, which successfully removes the long-term trend, allows us to focus on predicting anomalies. When the trend is included in the data, it can mask the actual predictive skill of the model. For instance, the correlation coefficient might be artificially inflated due to the inherent increasing or decreasing trend over time. Removing the trend could denoise the performance and show how well the methodology captures anomalies around the trend. For operational use of the forecasts, the trend could be added back in the same way it was removed.

It is important to note that in the work of [43], it was not mentioned any details about the detrending. The data used in their work was standardized (targets after monthly-averaged) for each simulation, but a reference period was not specified. As an example, we perform an experiment where the data is not detrended and instead, the standardization is done as mentioned in the original paper. CMIP6 and ERA5 data are subtracted from their mean and divided by the standard deviation of the entire period. The mean was calculated independently for each model, with every simulation of a model having the same mean each year. Figure 13 shows the performance of the model when only this standardization is considered. We can observe that the performance excels at predicting warm events (+1 +2 sigma AUPRC) at the expense of having a low correlation with the observations, a worse ensemble quality, and a huge bias.

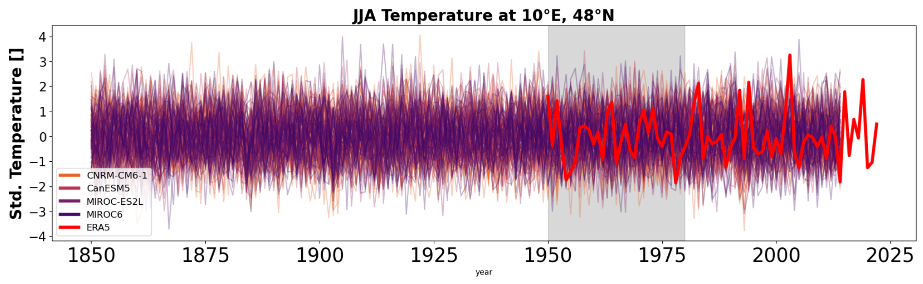


Figure 12: The data is standardized in two steps: first, successfully detrending it by subtracting the trend (loess for ERA5 and ensemble mean for CMIP6) and second, dividing by the standard deviation obtained from the reference period highlighted in grey. The resulting anomalies of the climate models and ERA5 are now aligned and the initial bias has been solved. Additionally, having the data centered around zero facilitates the convergence of the training and avoids vanishing or exploding gradients.

After testing the standardization approach described in [43], the results are strongly biased which does not occur in their work. We can assume the implementation was wrong or some steps were not detailed. Thus, we will standardize the data as shown in figure 12, which successfully detrends the data and the resulting anomalies are normalized correctly.

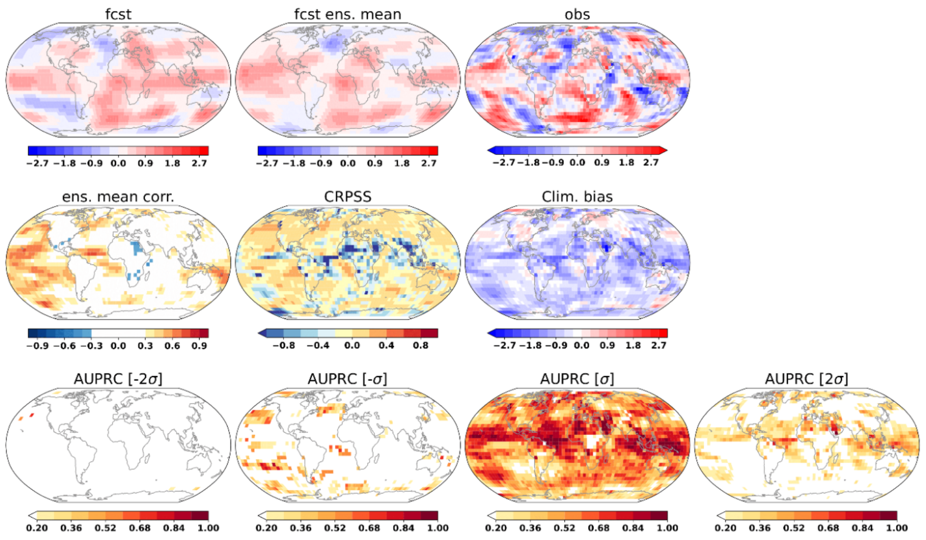


Figure 13: Experiment standardizing the climate model and ERA5 data only by the model mean. The trend is not sufficiently removed. The model is highly biased to predicting warmer temperatures thus having high AUPRC on +1 and +2 standard deviations at the expense of predicting badly cold anomalies

4.2.2 Impact of the ensemble size

A key advantage of our methodology compared to established approaches like SEAS5 lies in its ability to readily scale the ensemble forecast size with minimal computational overhead. This characteristic allows for potentially improved performance by leveraging larger ensembles without significant computational costs. Figure 14 shows the importance of the size of the ensemble for 3-month seasonal predictions. For that, we obtain a 300-member forecast from May to summer and perform bootstrap resamples at different sample sizes (similarly done in [43]). For each resample, the mean ensemble forecast is used to compute the spatial average correlation globally (y-axis). This is done 100 times at each sample size (x-axis). The result displays that an increase in the ensemble size helps achieve consistently better performance in terms of correlation, but the difference is exponentially lower as the sample size increases. Increasing the ensemble size further from 256 would not imply a significant increase in performance, because of that, the ensemble forecasts will be performed on this ensemble size.

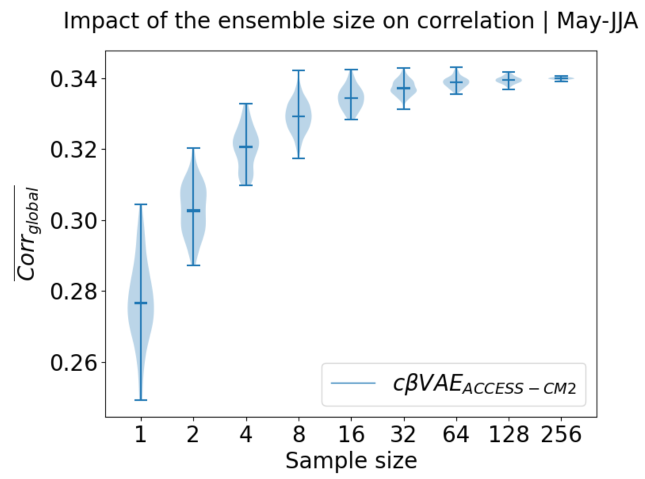


Figure 14: Effect of the increase on ensemble size on the global correlation. The results are obtained on cβVAE with the first embedding available (climate model ACCESS-CM2) using the mean ensemble forecast. The horizontal axis represents the number of samples and the vertical one the global correlation. From an original ensemble forecast of 300 members, the distribution shown at each step is formed by 100 bootstrap resamples of the corresponding samples

4.2.3 Hyperparameter tuning

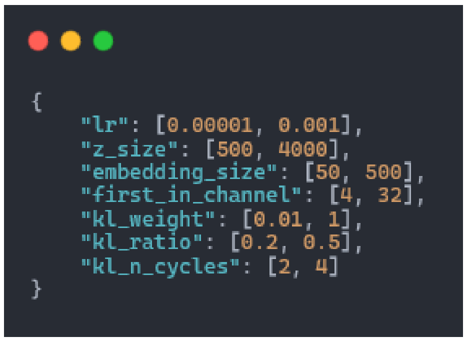
Using the experiment setup explained in section 4.1, we tested different values for a set of hyperparameters. Table 2 shows the subset of the parameters used for tunning. The hyperparameter tuning followed the suggestions from [62]. We analyze the results from these experiments to determine optimal hyperparameters for the subsequent 3-month prediction tasks (excluding the learning rate, which was left unfixed). The performance for each trial is evaluated based on the model with the lowest training-validation loss. In cases where a KL divergence scheduler is implemented, the best model is chosen by considering only epochs where the annealed weight for beta reaches its maximum value (1). This approach ensures we select a model that achieves peak performance during the period with the strongest KL divergence influence.

We define hyperparameter ranges for our experiments using a JSON (JavaScript Object Notation) file, as illustrated in figure 15. The ranges where neither the start nor the end of the range are floating values will be considered integer hyperparameters. Following the guidebook mentioned [62], we use a version of a quasi-random search algorithm, called Sobol sequence. The configuration files are created by iterating over the Sobol sequences considering the different hyperparameter ranges.

Part to optimize	Hyperparameters
Embedding layer	*embedding_size* - Output size of the embeddings
Bottleneck	*z_size* - Size of the latent mean and standard deviation
	first_input_channel - Channels of the first decoder residual block
Trainer	*epochs* - Number of training steps
Optimizer	*lr* - Learning rate (step size) of the stochastic gradient descent
KL Divergence	*kl_weight* - Fixed weight for the KL term of the loss
	For the scheduler: *kl_ratio kl_n_cycles kl_monotonic_function*
Dropout	*fixed_dropout* - Fixed dropout value
	dropout_pressence - Dropout on encoder/decoder or both

Table 2: Hyperparameters considered for the different parts of the methodology. The only optimizer considered is Adam for the moment, for which only the learning rate was considered to be tuned, having: $beta_1$=0.9, $beta_2$=0.999, *epsilon*=1e-07 as their default values. The option to schedule the dropout was implemented but discarded for the sake of simplicity.

The hyperparameter tuning starts on a basic configuration (no Kullback-Leibler (KL) scheduler nor dropout) which was the one used for the model cVAE0. As an example, the first experiment was run using the hyperparameter ranges shown in figure 15 to define 128 configurations following a Sobol sequence. figure 16 shows the experiment results in terms of each trial's validation loss history. The training that achieved the lowest validation loss is highlighted in green and the lowest five are indicated in the top-right legend. Along with this initial experiment, we ran others to fix the hyperparameters' values and determine the usefulness of the KL Scheduler and the dropout. In terms of the KL Scheduler, the decision was to include it since it helped stabilize the training even though the results did not noticeably improve. Regarding the dropout, it was included solely in the decoder based on the results obtained. This decision is supported by studies that show how a weak decoder can help prevent posterior collapse [64]. Finally, the YAML code in listing 2 shows the final set of hyperparameters.



```
{
    "lr": [0.00001, 0.001],
    "z_size": [500, 4000],
    "embedding_size": [50, 500],
    "first_in_channel": [4, 32],
    "kl_weight": [0.01, 1],
    "kl_ratio": [0.2, 0.5],
    "kl_n_cycles": [2, 4]
}
```

Figure 15: JavaScript Object Notation (JSON) file with the ranges of the hyperparameters for a certain experiment. With these ranges, a Sobol sequence is started to define multiple configurations that are ran and compared to tune the hyperparameters

Source Code 2: Fixed hyperparameters after hyperparameter tuning

```
model:
  num_in_features: 6
  embedding_size: 200
  z_size: 2000
  first_in_channel: 25

trainer:
  lr: 1.e-4
  kl_weight: 1
  kl_ratio: 0.5
  kl_n_cycles: 4
  kl_monotonic_fn: "linear"
  kl_start: 0
  kl_stop: 1
  dropout_pressence: ["Decoder"]
  fixed_dropout: 0.5
```

The resulting model after the hyperparameter tuning experiments is denoted as cβVAE. We compare its performance against the cVAE0 created on the aforementioned starting configuration in appendix A. For that comparison, we use a scorecard as defined in section 4.1 where we focus on correlation, CRPS, and AUPRC at ± 1 standard deviations (see section 3.3 for metric interpretations). We use the model cVAE0as the reference. Thus, positive (red) values show an improvement by the cβVAE in the specific region and metric. Except for the CRPS where negative differences (blue) imply better quality of the ensem-

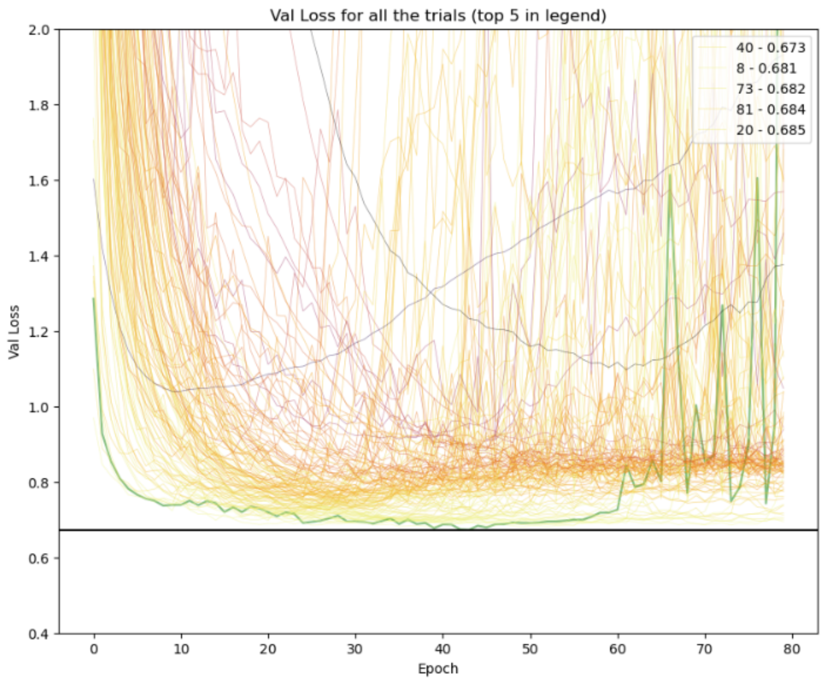


Figure 16: Validation losses of the different trials of the first experiment by epoch. Highlighted in green is the training that achieved the lowest validation of all the trials in the experiment

ble forecast. Figure 17 shows the difference between cβVAE and cVAE0 both predicting summer from May and using the first embedding available that corresponds to the model ACCESS-CM2. We can conclude that the results improved significantly in most regions after the experiments.

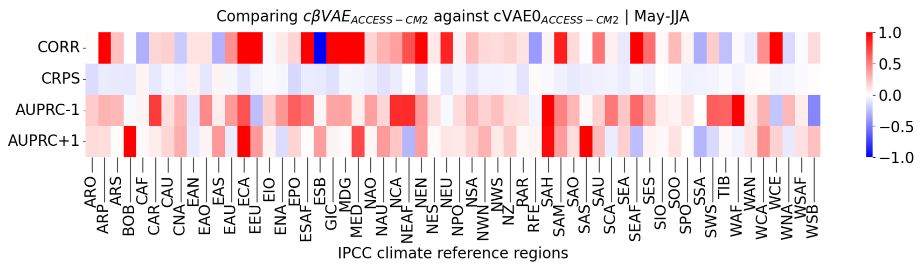


Figure 17: Scorecard comparing cβVAE to cVAE0 at predicting summer from May on verification period using the first embedding that corresponds to the climate model ACCESS-CM2. cVAE0 uses the configuration before hyperparameter tuning and cβVAE is the result of the hyperparameter tuning strategy. We find overall improvements in the correlation of most regions, a lower CRPS, and better classification of both colder and warmer events. We can conclude that tuning the hyperparameters improved the overall performance of the model

4.2.4 Fine-tuned embeddings using observational data

Based on the experiment done in [43] on obtaining an "optimal embedding" by fine-tuning a trained model on ERA5 data, we perform a similar experiment and compare the results with the other embeddings. For fine-tuning, every layer except the entity embedding one is frozen. Then, we run 40 training epochs with everything as the initial training but using the training split of ERA5 (which uses the reference period 1950-185) and substituting the first embedding available. After the fine-tuning of observational data of the first embedding, we make use of t-distributed stochastic neighbor embedding (t-SNE), a statistical tool used for dimensionality reduction purposes, to display the embedding space in 2 dimensions. Figure 18 shows the result of running t-SNE from scikit-learn [65] on a perplexity of 15. The smaller the distance between points in the t-SNE representation, the closer their entity embeddings are. It can be noticed that the original embeddings tend to be as much separated as possible, implying that the learned representations differ a lot between themselves. Regarding the ERA5 fine-tuned embedding, it is almost equidistant to any of the other points that represent the original embeddings from the climate models used.

Considering how none of the original embeddings are close to the fine-tuned one we can assume that no model performs considerably similarly to ERA5 on 3-month temperature prediction. If some embeddings were much closer to the "optimal" one, they can be considered by themselves for the corresponding prediction task. To test the performance implications of using the fine-tuned embedding, figure 19 compares the two of them showing how there are regions where the metrics improve and others where they worsen. Overall, the use of the fine-tuned embedding does not make a significant enough improvement.

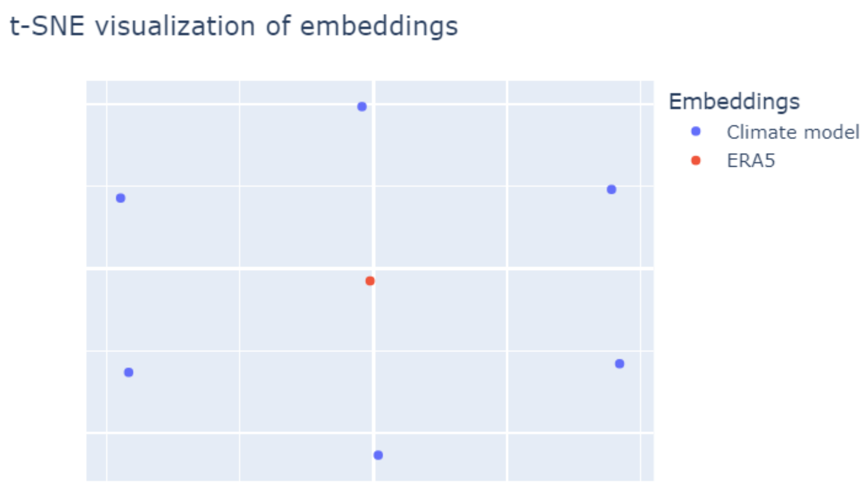


Figure 18: 2D visualization of the original embeddings and the fine-tuned embedding using ERA5 data. The dimensionality reduction performed using t-SNE was considering 15 nearest neighbors (perplexity) and the default parameters from the method sklearn.manifold.TSNE. Distances in the two dimensions can be interpreted as distances between the embeddings. The fact that the points are evenly distributed in the latent space hinders that the representation may not be meaningful since there should be a certain degree of similarity between the climate models

Since the fine-tuned is not as useful as desired, we compare it with the mean embedding (by first acquiring every embedding and then averaging them) which should be similar based on figure 18. We can see in figure 20 that using the mean embedding is not significantly better than the optimal embedding either. This is to be expected since the mean embedding is not distant from the fine-tuned one if it was included in the t-SNE representation previously. Lastly, to check if any embedding is better than the mean one, figure 21 shows a comparison where the mean embedding is used as the control group. Overall, the mean embedding seems to lead to slightly better performance but any embedding could be used and the results would not be heavily affected. We can conclude that either the fine-tuned, the mean, or any of the learned embeddings influence considerably the resulting forecasts. Thus, the learned embedding representations are not meaningful in this setting. The mean embedding will be used for further tasks since the performance is similar to using the optimal embedding and much easier to compute.

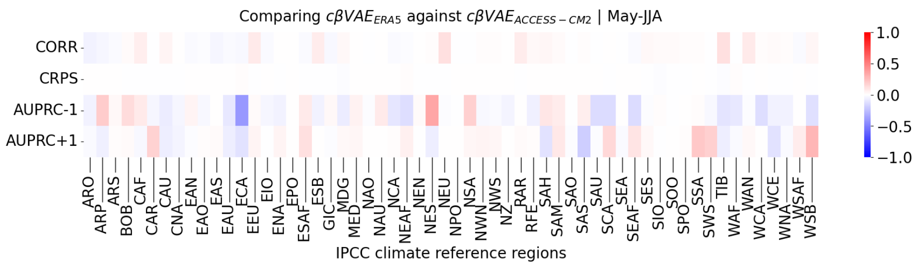


Figure 19: Scorecard considering the comparison between cβVAE using the fine-tuned embedding on ERA5 data against using the first one (climate model ACCESS-CM2) predicting summer from May on the reference period set for verification. Overall, There are no significant differences that can be noticed in the comparison

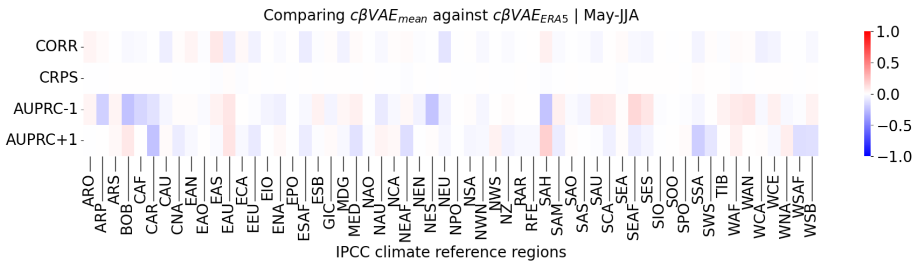


Figure 20: Scorecard considering the comparison between cβVAE using the mean embedding against the fine-tuned embedding at predicting summer from May on the reference period set for verification. The mean embedding is obtained as the mean of the embeddings. There is no significant differences to choose of the two options. Considering that the fine-tuned embedding is approximately equidistant to the original embeddings, it makes sense that it is similar to the mean embedding

4.3 Final results

The ablation studies conveyed in section 4.2, helped us make decisions to define the final model used to obtain the final results. The Final model (cβVAE) has its hyperparameters tuned as in listing 2, the ensemble forecasts will have 256 simulations, we will use the mean embedding for inference, and the data will be standardized and detrended by simulation and year. To measure the relative performance of this model, the results will be compared against the different benchmarks described in section 4.1: Deep learning baseline (cVAE0), SEAS5, Climatology (CLIM), and Persistence (PERS).

Figure 22 shows the scorecard (see section 4.1) comparing cβVAE using

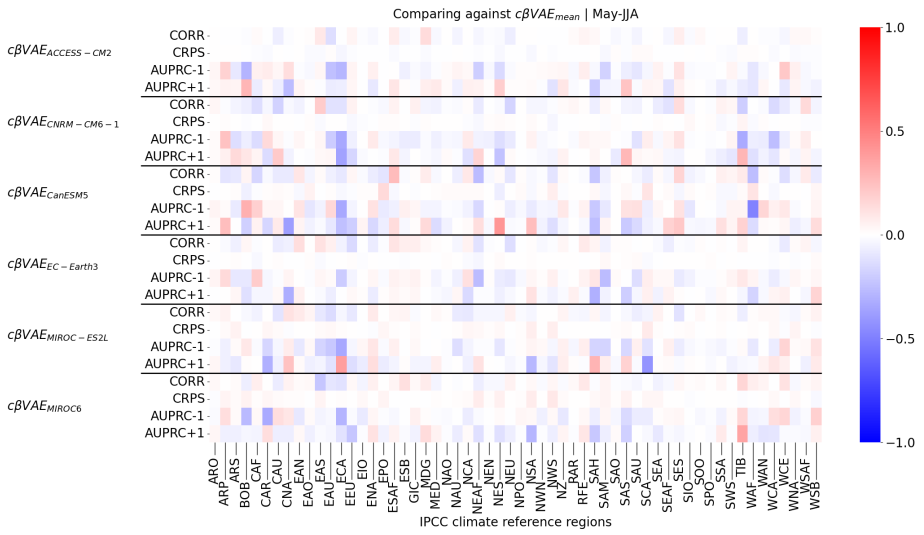


Figure 21: Joined scoreboard that shows the difference of the results considering each embedding against the results obtained using the mean embedding for the task of predicting summer from May on the reference period set for verification. The rows are separated into groups showing the considered metrics where each group is associated with the comparison between the results using the corresponding embedding and the results using the mean embedding. It can be considered that no embedding leads to significantly better results, concluding that the impact of the embeddings on the predictions is not relevant

the mean embedding and the modifications mentioned above against the three baselines considered. The correlation between climatology and cβVAE was excluded since the mean ensemble forecast of the climatology is full of values close to zero. It can be seen that overall ECMWF's fifth generation seasonal forecast system (SEAS5) surpasses the performance of the proposed approach (a lower CRPS is better), the persistence model achieves similar correlations but the other metrics are generally worse and, the results of climatology are comparably poor. Still, it is noticeable that even if SEAS5 surpasses the proposed approach, there are multiple regions like SPO, SWS, or SOO where the metrics are fairly similar. Most of the regions where the performance is matched correspond to ocean regions, they are easier to predict than land areas since the Sea Surface Temperature (SST) is closely related to the Temperature 2-meter Above Surface (TAS) (the target) of the same region.

We are interested in evaluating how the lead time affects the skillfulness of the predictions using the proposed final model cβVAE compared to the different benchmarks defined in section 4.1. For that purpose, the predictions were obtained using as input May to January to predict the summer of the same

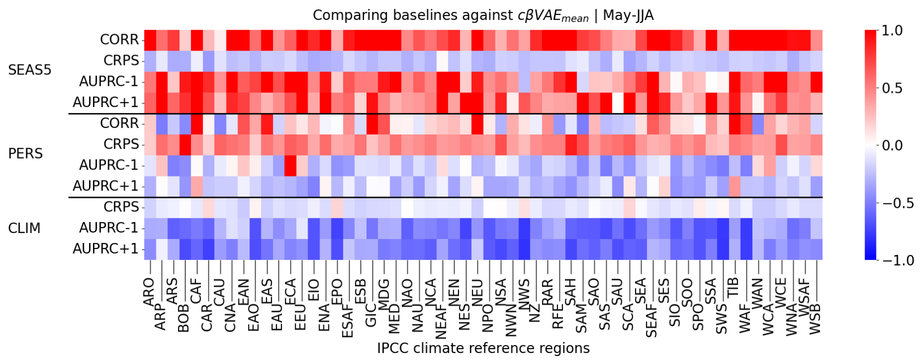


Figure 22: Scorecard with comparisons against cβVAE using the mean embedding considering the correlation and Area Under the Precision-Recall Curve (AUPRC) at ± 1 standard deviation. It can be seen that the predictions from SEAS5 are better in almost every region whereas the persistence performs generally worse or equally. On the other hand, the climatology is worse than cβVAE at almost every region.

year. Figure 23 displays the averaged ensemble mean correlation of the North South-America (NSA) region from the IPCC climate reference regions. This region is mostly formed by land which is harder to predict since the changes occur faster than those in the ocean. We can clearly see that the ensemble mean correlation decays as the lead time increases except for the climatology which is the same. As expected, the highest correlation of the ensemble mean for the observations was achieved by SEAS5 at every lead time. On the other hand, CLIM stands at a null correlation since the mean ensemble forecast is almost null as a result of averaging the anomalies of multiple years. In May and April, we find the following order: the final model (cβVAE), the deep learning baseline (cVAE0), PERS. This shows again how our proposed model serves as a middle point between persistence and SEAS5. At the following lead times, the superiority of the proposed models is not maintained showing similar results.

4.4 Discussion and Limitations

Several constraints have emerged throughout this project. Many of these constraints arise from the complexity of working with climate data; others stem from the limitations imposed by the Variational AutoEncoder (VAE) architecture.

Fundamental data limitations include the number of climate models used and the performance of these models representing the natural climate dynamics, the spatial resolution of the grid, and the quantity and quality of the variables con-

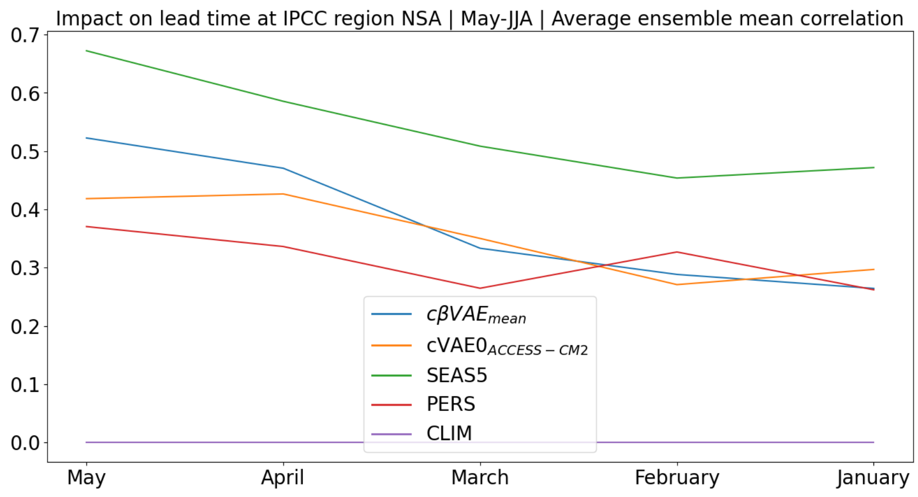


Figure 23: Impact on lead time at the North South-American region comparing the average ensemble mean correlation. As expected, the best correlation was achieved in the SEAS5 forecasts. Next, we find that for May and April, cβVAE stands over the other consider baselines having cVAE0 next in correlation. At March our both approaches join and after that they perform similarly to the persistence method. This is just the conclusion of focusing on a certain land region, but as we can see in figure 22, at most regions cβVAE will achieve a lower correlation than SEAS5 and a higher one than the persistence model.

sidered. Likewise, several pre-processing steps are needed to accommodate the data for the training pipeline; these include re-gridding to a joint spatial and temporal resolution or de-trending and standardizing the data. This thesis has shown that minor changes in these pre-processing steps might lead to drastic changes in the final model's performance and validation. Additionally, the heterogeneous availability of the number of simulations for each climate model can affect the data split, unbalancing their representation and potentially contributing to the ineffectiveness of the entity embeddings. This issue was left out of the scope of this thesis and remains to be further explored.

The decision to use VAEs (Variational Autoencoders) over other generative models has certain limitations. One major issue is that the generated samples can often appear blurry. This is primarily because VAEs optimize a trade-off between reconstruction accuracy and the smoothness of the latent space, which can lead to less sharp images. Additionally, VAEs are prone to a phenomenon known as posterior collapse, where the learned latent representations become too simplistic and do not effectively capture the variability of the data. This results in the encoder producing nearly identical outputs for different inputs, undermining the model's generative capabilities.

When it comes to seasonal predictions, these limitations can have significant impacts. Blurred generation of fields can lead to less precise predictions, as the model may need to capture the finer details and variations in seasonal patterns. Furthermore, posterior collapse implies the model is not correctly predicting the range of potential climatic scenarios, leading to generalized predictions that fail to account for extreme weather events or unusual seasonal shifts. This lack of specificity can result in less valuable predictions for planning and decision-making in sectors reliant on accurate seasonal forecasts, such as agriculture and disaster management.

Overall, while many factors have hindered the performance of our forecasts, these limitations also provide opportunities for future work. Thus, we foresee the following future steps:

- **Expanding the Dataset:** Incorporate more and better climate models, enhance the spatial resolution, and consider additional variables to improve the representation of the climate system.
- **Improve the model architecture:** Explore more recent architectures like transformers, graph neural networks, or other generative approaches like score-based models that better capture the nonlinear and stochastic nature of the climate system.
- **Refining Entity Embeddings:** Investigate alternative methods to develop more meaningful entity embeddings and how the information of this embedding is merged into the network for better model performance.

By addressing these areas, we can enhance the performance and applicability of VAEs in seasonal climate predictions.

5 Conclusion and future work

This thesis investigates the application of conditional variational autoencoders (cVAEs) for seasonal temperature prediction. Inspired by the methodology presented in [43], we propose an approach that incorporates a conditional β-VAE with dropout regularization in residual blocks. The proposed model aims to predict the average June-July-August (JJA) temperature based on the climate state of the preceding month.

While the achieved performance did not surpass established methods like ECMWF's fifth generation seasonal forecast system (SEAS5), it demonstrates potential as a viable alternative. The cVAE framework outperforms climatology and persistence at early lead times while offering valuable flexibility. This flexibility allows for its application as a shared pipeline for various weather and climate machine-learning tasks.

In addition to the implementation of the proposed methodology, rigorous data handling procedures were employed. Both climate model and observational data undergo standardization with detrending to focus on anomalies. Evaluation involves comparisons with frequently used baselines like climatology, persistence, and SEAS5, the current leading seasonal forecasting model. We also utilize metrics like correlation and CRPS to capture key aspects of forecast performance.

This project contributes to the growing field of machine learning applications in weather and climate prediction. By leveraging observations and climate model data, machine learning offers promising avenues for improved seasonal forecasts. Our work serves as a starting point for further research, providing valuable insights into the current state-of-the-art methods and establishing a flexible pipeline for future advancements in seasonal forecasting with machine learning.

In future work, we foresee different venues to improve the performance of the proposed model. As an example, expanding the dataset by incorporating a greater number of simulations, extending the time period covered, or including additional variables, could contribute to better performance. On the same line, the flexibility of the implementation allows for modifications such as using different loss functions, alternative distributions for the latent space, or extensions to the architecture. Additionally, post-processing calibrations of the results could be considered as suggested in [41] to adjust the distribution of the ensemble forecasts.

Although other generative approaches were considered, Variational AutoEncoders were chosen for specific reasons; however, future research could investigate a similar setup using diffusion models. The challenge with diffusion models lies in the potential trade-off between improved accuracy and slower inference, possibly resulting in smaller ensemble forecasts. Nonetheless, given that the current ensemble size does not compensate for quality deficiencies, diffusion models might enhance the overall quality of the ensemble forecast. Furthermore, considering the limited impact of entity embeddings for the three-month prediction task, we propose using the reserved observations for fine-tuning the entire network after pre-training with climate model data.

A Annotated configuration file

Source Code 3: Example yaml config

```yaml
# Random seed used to initialize every random process, not forgetting
  # about the DataLoaders to secure a reproducible execution.
seed: 1357

# Output directory for the experiment where each trial will have its own directory
output_dir: '/esarchive/scratch/lpalma/tmp/'

# Sets the threshold level required to log a message.
  # Being at INFO, every logging will be shown.
DEBUG_level: 'INFO'

# Configuration for the data from the CMIP6 climate models
dataset_cmip6:

  # NetCDF file containing the target data from the climate models
  target_path: '/esarchive/scratch/lpalma/esmvaltool/V23_INPUTS/work/inputs/cmip/inputs.nc'
  # NetCDF file containing the input data from the climate models
  input_path: '/esarchive/scratch/lpalma/esmvaltool/V23_INPUTS/work/inputs/cmip/inputs.nc'
  # Input variables to be selected
  input_vars: &input_vars ['tos','zg500','tas']
  # Target variable to be selected. Only one
  target_var: &target_vars 'tas'
  # Months of the year used for input. Working with one or the mean of them
  input_months: &input_months [4]
  # Months of the year used for target. Working with one or the mean of them
  target_months: &target_months [6, 7, 8]
  # Range of years used to select inputs and targets. Last year not included
  years: [1850, 2014]
  # Number of threads used for the data loading
  num_workers: &num_workers 4
  # Flag to consider or not the embeddings
  embeddings: True
  # Array that defines the region of interest for all the input variables.
  input_domain: &input_domain [-80, 80, -180, 180] # Global without poles
  # Array that defines the target region
  target_domain: &target_domain [-80, 80, -180, 180] # Global without poles
  # Flag to compute or not the mean of the input months
  inputs_month_mean: &inputs_month_mean False
  # Flag to compute or not the mean of the target months
  target_month_mean: &target_month_mean True
  # Arguments for the scaler used to standardize the inputs.
  inputs_scaler:
    # How to compute the climatology to be substacted as the mean. Can be:
      # clim - the mean is computed by the groupby type only
      # ens_mean - the mean is computed by the groupby type and the ensemble dimension
      # ens_mean_loess - loess is applied after obtaining the mean as in ens_mean
        # The trend is more accurately obtained but takes more time on large datasets
    clim_type: 'ens_mean'
    # Reference period from which the mean and standard deviation will be used
      # To standardize the inputs
    std_ref_years: [1950, 1985]
    # Primary dimension on which the mean will be computed independently
    groupby_type: 'model_id'

  # Idem as the inputs_scaler but for the targets
  target_scaler:
    clim_type: 'ens_mean'
    std_ref_years: [1950, 1985]
    groupby_type: 'model_id'

# Configuration for the data from the ERA5 climate models
  # Same as the dataset_cmip6 configuration but for the ERA5 data
dataset_era5:

  target_path: '/esarchive/scratch/lpalma/esmvaltool/V23_INPUTS/work/inputs/era5/inputs.nc'
  input_path: '/esarchive/scratch/lpalma/esmvaltool/V23_INPUTS/work/inputs/era5/inputs.nc'
  input_vars: *input_vars
  target_var: *target_vars
  input_months: *input_months
  target_months: *target_months
  years: [1950, 2021] # A range will created. Last year not included
  num_workers: *num_workers
  embeddings: False
  input_domain: *input_domain
  target_domain: *target_domain

  inputs_month_mean: *inputs_month_mean
  target_month_mean: *target_month_mean

  inputs_scaler:
    clim_type: 'ens_mean_loess'
    std_ref_years: [1950, 1985]

  target_scaler:
    clim_type: 'ens_mean_loess'
    std_ref_years: [1950, 1985]

# Model specific configuration
model:

  # Class of the model to be instantiated.
    # The other arguments are dependent on the type of model
  type: "CGF"
  # Input size of the embedding layer. Should match the number of climate models
  num_in_features: 6
  # Output size of the embedding layer
  embedding_size: 30
  # Size of the mean and standard deviation obtained in the bottleneck from the
    # outputs of the encoder
  z_size: 1000
  # Chanels of the first transposed convolution block of the decoder.
    # It is allowed to be changed to adapt to the z_size parameter
  first_in_channel: 8

  # Shape of the inputs(X). The variables are stacked in the channel dimension
  input_shape_X: [3, 72, 32]
  # Chanels of the residual blocks of the input's encoder
  residual_blocks_X: [16, 32, 64]
  # Shape of the targets(Y). The channel dimension is mandatory even if empty
  input_shape_Y: [1, 72, 32]
  # Chanels of the residual blocks of the target's encoder
  residual_blocks_Y: [16, 32, 64]
  # Channels of the transposed residual blocks on the decoder
  residual_blocks_decoder: [32, 32, 32]

# Configuration for the trainining process. Contains the arguments for the DataLoader,
  # the optimizer, the loss function and the dropout.
  # Note: The dropout is included in the configuration of the training and not
  # the model since we were considering whether or not to anneal the dropout
  # as with the KL divergence in the loss
trainer:

  # Number of training epochs
  epochs: 50
  # Size of the mini-batches considered for the Stochastic Gradient Descent
  batch_size: 128

  # Learning rate or step size of the optimizer. For now, only using Adam
  lr: 1.e-4
  # Weight decay factor for the optimizer (Adam) used for regularization
  weight_decay: 1.e-1
  # Name of the loss function used to train the model
  loss: "elbo_loss_mse"
  # Fixed weight of the KL term in the ELBO.
    # This parameter is dependent on the loss selected
  kl_weight: 0.1

  # KL scheduler that will create another weight (beta) to tune the importance
    # of the term in the ELBO loss during the training epochs.
    # At each cycle, the weight starts at the starting point and increases
    # considering the ratio and the mononotonically increasing function until reaching
    # the end point. When a new cycle begins, the weight is reset to the start.
  # Ratio considered as to how much the scheduled value increases at each step
  kl_ratio: 1
  # Number of cycles of the scheduler
  kl_n_cycles: 2
  # Monotonically increasing function to schedule the KL beta (another weight)
  kl_monotonic_fn: "linear"
  # Starting point of the scheduler
  kl_start: 1
  # End point of the scheduler
  kl_stop: 1

  # Dropout used in the encoder and/or decoder.
    # no - no dropout anywhere
    # fixed - fixed dropout where indicated in dropout_pressence
    # schedule - scheduled dropout in dropout_pressence
  dropout: "fixed"
  # Where the dropout will be included.
    # For both encoder and decoder, the position is fixed
  dropout_pressence: ["Encoder", "Decoder"]

  # Dropout value used when the dropout type is set to fixed
  fixed_dropout: 0.5

  # Dropout scheduler considered when the type is set to schedule
  # The arguments are the same as the KL ones
  dropout_ratio: 1
  dropout_n_cycles: 2
  dropout_monotonic_fn: "linear"

# Configuaration of the verification after the training
verification:

  # List of metrics to be computed
  metrics :  ["ensmean_corr", "auprc_neg1sig", "auprc_neg2sig", "auprc_1sig", "auprc_2sig",
  # Flag to compute the metrics as skill scores compared to the
    # climatology of the reference years
  comp_clim_fcst: True
  # Years considered to verify the method.
  reference_years: [1986, 2020]
```

References

[1] "Boundary conditions - glossary of meteorology." [Online]. Available: https://glossary.ametsoc.org/wiki/Boundary_conditions

[2] S. J. Mason, "Guidance on verification of operational seasonal climate forecasts," *World Meteorological Organization, Commission for Climatology XIV Technical Report*, 2013.

[3] American Metheorological Society, "Geopotential height - Glossary of Meteorology." [Online]. Available: https://glossarytest.ametsoc.net/wiki/Geopotential_height

[4] C. Field, V. Barros, T. Stocker, D. Qin, D. Dokken, K. Ebi, M. Mastrandrea, K. Mach, G.-K. Plattner, S. Allen, M. Tignor, and P. Midgley, "Glossary of terms," in *Managing the Risks of Extreme Events and Disasters to Advance Climate Change Adaptation*, C. Field, V. Barros, T. Stocker, D. Qin, D. Dokken, K. Ebi, M. Mastrandrea, K. Mach, G.-K. Plattner, S. Allen, M. Tignor, and P. Midgley, Eds. Cambridge, UK, and New York, NY, USA: Cambridge University Press, 2012, pp. 555–564.

[5] "Operational Forecasts | IBF." [Online]. Available: https://ibf.org/knowledge/glossary/operational-forecasts-204

[6] E. N. Lorenz, "Section of Planetary Sciences: The Predictability of Hydrodynamic Flow*,†," *Transactions of the New York Academy of Sciences*, vol. 25, no. 4 Series II, pp. 409–432, 1963, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.2164-0947.1963.tb01464.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.2164-0947.1963.tb01464.x

[7] T. C. Portele, C. Lorenz, B. Dibrani, P. Laux, J. Bliefernicht, and H. Kunstmann, "Seasonal forecasts offer economic benefit for hydrological decision making in semi-arid regions," *Sci Rep*, vol. 11, no. 1, p. 10581, May 2021, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41598-021-89564-y

[8] P. Roudier, A. Alhassane, C. Baron, S. Louvet, and B. Sultan, "Assessing the benefits of weather and seasonal forecasts to millet growers in Niger," *Agricultural and Forest Meteorology*, vol. 223, pp. 168–180, Jun. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168192316302416

[9] F. J. Meza, J. W. Hansen, and D. Osgood, "Economic Value of Seasonal Climate Forecasts for Agriculture: Review of Ex-Ante Assessments and Recommendations for Future Research," *Journal of Applied Meteorology and Climatology*, vol. 47, no. 5, pp. 1269–1286, 2008, publisher: American Meteorological Society. [Online]. Available: https://www.jstor.org/stable/26172212

[10] M. Bruno Soares, M. Daly, and S. Dessai, "Assessing the value of seasonal climate forecasts for decision-making," *WIREs Climate Change*, vol. 9, no. 4, p. e523, 2018, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wcc.523. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/wcc.523

[11] L. E, "The nature and theory of the general circulation of the atmosphere," *World Meteorological Organization*, vol. 161, 1967. [Online]. Available: https://cir.nii.ac.jp/crid/1574231873842498176

[12] N. A. Phillips, "The general circulation of the atmosphere: A numerical experiment," *Quarterly Journal of the Royal Meteorological Society*, vol. 82, no. 352, pp. 123–164, 1956, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qj.49708235202. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/qj.49708235202

[13] R. Lam, A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, A. Pritzel, S. Ravuri, T. Ewalds, F. Alet, Z. Eaton-Rosen, W. Hu, A. Merose, S. Hoyer, G. Holland, J. Stott, O. Vinyals, S. Mohamed, and P. Battaglia, "GraphCast: Learning skillful medium-range global weather forecasting," Dec. 2022, arXiv:2212.12794 [physics]. [Online]. Available: http://arxiv.org/abs/2212.12794

[14] I. Price, A. Sanchez-Gonzalez, F. Alet, T. Ewalds, A. El-Kadi, J. Stott, S. Mohamed, P. Battaglia, R. Lam, and M. Willson, "GenCast: Diffusion-based ensemble forecasting for medium-range weather," Dec. 2023, arXiv:2312.15796 [physics]. [Online]. Available: http://arxiv.org/abs/2312.15796

[15] D. Kochkov, J. Yuval, I. Langmore, P. Norgaard, J. Smith, G. Mooers, J. Lottes, S. Rasp, P. Düben, M. Klöwer, S. Hatfield, P. Battaglia, A. Sanchez-Gonzalez, M. Willson, M. P. Brenner, and S. Hoyer, "Neural General Circulation Models," Nov. 2023, arXiv:2311.07222 [physics]. [Online]. Available: http://arxiv.org/abs/2311.07222

[16] T. Kurth, S. Subramanian, P. Harrington, J. Pathak, M. Mardani, D. Hall, A. Miele, K. Kashinath, and A. Anandkumar, "FourCastNet: Accelerating Global High-Resolution Weather Forecasting Using Adaptive Fourier Neural Operators," in *Proceedings of the Platform for Advanced Scientific Computing Conference*, ser. PASC '23. New York, NY, USA: Association for Computing Machinery, Jun. 2023, pp. 1–11. [Online]. Available: https://dl.acm.org/doi/10.1145/3592979.3593412

[17] K. Bi, L. Xie, H. Zhang, X. Chen, X. Gu, and Q. Tian, "Pangu-Weather: A 3D High-Resolution Model for Fast and Accurate Global Weather Forecast," Nov. 2022, arXiv:2211.02556 [physics]. [Online]. Available: http://arxiv.org/abs/2211.02556

[18] K. Chen, T. Han, J. Gong, L. Bai, F. Ling, J.-J. Luo, X. Chen, L. Ma, T. Zhang, R. Su, Y. Ci, B. Li, X. Yang, and W. Ouyang, "FengWu: Pushing the Skillful Global Medium-range Weather Forecast beyond 10 Days Lead," Apr. 2023, arXiv:2304.02948 [physics]. [Online]. Available: http://arxiv.org/abs/2304.02948

[19] L. Chen, X. Zhong, F. Zhang, Y. Cheng, Y. Xu, Y. Qi, and H. Li, "FuXi: A cascade machine learning forecasting system for 15-day global weather forecast," Oct. 2023, arXiv:2306.12873 [physics]. [Online]. Available: http://arxiv.org/abs/2306.12873

[20] M. Pal, R. Maity, J. V. Ratnam, M. Nonaka, and S. K. Behera, "Long-lead Prediction of ENSO Modoki Index using Machine Learning algorithms," *Sci Rep*, vol. 10, no. 1, p. 365, Jan. 2020, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41598-019-57183-3

[21] M. A. De Castro Santos, D. A. Vega-Oliveros, L. Zhao, and L. Berton, "Classifying El Niño-Southern Oscillation Combining Network Science and Machine Learning," *IEEE Access*, vol. 8, pp. 55 711–55 723, 2020, conference Name: IEEE Access. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9042254

[22] H. A. Dijkstra, P. Petersik, E. Hernández-García, and C. López, "The Application of Machine Learning Techniques to Improve El Niño Prediction Skill," *Frontiers in Physics*, vol. 7, 2019. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fphy.2019.00153

[23] J. Dutton, "Sea surface temperatures: Useful role in long-range forecasting," Dec. 2021. [Online]. Available: https://www.worldclimateservice.com/2021/07/18/sea-surface-temperatures/

[24] N. US Department of Commerce, "STI Climate Bulletin," publisher: NOAA's National Weather Service. [Online]. Available: https://www.weather.gov/sti/STI_ClimateBulletin

[25] S. J. Johnson, T. N. Stockdale, L. Ferranti, M. A. Balmaseda, F. Molteni, L. Magnusson, S. Tietsche, D. Decremer, A. Weisheimer, G. Balsamo, S. P. E. Keeley, K. Mogensen, H. Zuo, and B. M. Monge-Sanz, "SEAS5: the new ECMWF seasonal forecast system," *Geoscientific Model Development*, vol. 12, no. 3, pp. 1087–1117, Mar. 2019, publisher: Copernicus GmbH. [Online]. Available: https://gmd.copernicus.org/articles/12/1087/2019/

[26] D. H. Ballard, "Modular learning in neural networks," in *Proceedings of the sixth National Conference on artificial intelligence-volume 1*, 1987, pp. 279–284.

[27] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological cybernetics*, vol. 59, no. 4-5, pp. 291–294, 1988.

[28] G. E. Hinton and R. Zemel, "Autoencoders, minimum description length and helmholtz free energy," *Advances in neural information processing systems*, vol. 6, 1993.

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[30] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," Apr. 2021, arXiv:2003.05991 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2003.05991

[31] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," Dec. 2022, arXiv:1312.6114 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1312.6114

[32] T. Bayes and n. Price, "LII. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, F. R. S. communicated by Mr. Price, in a letter to John Canton, A. M. F. R. S," *Philosophical Transactions of the Royal Society of London*, vol. 53, pp. 370–418, Jan. 1997, publisher: Royal Society. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rstl.1763.0053

[33] K. P. Murphy, *Probabilistic machine learning: advanced topics*, ser. Adaptive computation and machine learning. Cambridge, Massachusetts London, England: The MIT Press, 2023.

[34] J. M. Joyce, "Kullback-Leibler Divergence," in *International Encyclopedia of Statistical Science*, M. Lovric, Ed. Berlin, Heidelberg: Springer, 2011, pp. 720–722. [Online]. Available: https://doi.org/10.1007/978-3-642-04898-2_327

[35] F. J. Doblas-Reyes, J. García-Serrano, F. Lienert, A. P. Biescas, and L. R. L. Rodrigues, "Seasonal climate predictability and forecasting: status and prospects," *WIREs Climate Change*, vol. 4, no. 4, pp. 245–268, 2013, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wcc.217. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/wcc.217

[36] J. Jakubik, S. Roy, C. E. Phillips, P. Fraccaro, D. Godwin, B. Zadrozny, D. Szwarcman, C. Gomes, G. Nyirjesy, B. Edwards, D. Kimura, N. Simumba, L. Chu, S. K. Mukkavilli, D. Lambhate, K. Das, R. Bangalore, D. Oliveira, M. Muszynski, K. Ankur, M. Ramasubramanian, I. Gurung, S. Khallaghi, Hanxi, Li, M. Cecil, M. Ahmadi, F. Kordi, H. Alemohammad, M. Maskey, R. Ganti, K. Weldemariam, and R. Ramachandran, "Foundation Models for Generalist Geospatial Artificial Intelligence," Nov. 2023, arXiv:2310.18660 [cs]. [Online]. Available: http://arxiv.org/abs/2310.18660

[37] S. K. Mukkavilli, D. S. Civitarese, J. Schmude, J. Jakubik, A. Jones, N. Nguyen, C. Phillips, S. Roy, S. Singh, C. Watson, R. Ganti, H. Hamann, U. Nair, R. Ramachandran, and K. Weldemariam, “AI Foundation Models for Weather and Climate: Applications, Design, and Implementation,” Sep. 2023, arXiv:2309.10808 [physics]. [Online]. Available: http://arxiv.org/abs/2309.10808

[38] T. Nguyen, J. Brandstetter, A. Kapoor, J. K. Gupta, and A. Grover, “ClimaX: A foundation model for weather and climate,” Jul. 2023, arXiv:2301.10343 [cs]. [Online]. Available: http://arxiv.org/abs/2301.10343

[39] X. Man, C. Zhang, J. Feng, C. Li, and J. Shao, “W-MAE: Pre-trained weather model with masked autoencoder for multi-variable weather forecasting,” Dec. 2023, arXiv:2304.08754 [physics]. [Online]. Available: http://arxiv.org/abs/2304.08754

[40] P. B. Gibson, W. E. Chapman, A. Altinok, L. Delle Monache, M. J. DeFlorio, and D. E. Waliser, “Training machine learning models on climate model output yields skillful interpretable seasonal precipitation forecasts,” *Commun Earth Environ*, vol. 2, no. 1, pp. 1–13, Aug. 2021, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s43247-021-00225-4

[41] T. R. Andersson, J. S. Hosking, M. Pérez-Ortiz, B. Paige, A. Elliott, C. Russell, S. Law, D. C. Jones, J. Wilkinson, T. Phillips, J. Byrne, S. Tietsche, B. B. Sarojini, E. Blanchard-Wrigglesworth, Y. Aksenov, R. Downie, and E. Shuckburgh, “Seasonal Arctic sea ice forecasting with probabilistic deep learning,” *Nat Commun*, vol. 12, no. 1, p. 5124, Aug. 2021, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41467-021-25257-4

[42] F. Ling, J.-J. Luo, Y. Li, T. Tang, L. Bai, W. Ouyang, and T. Yamagata, “Multi-task machine learning improves multi-seasonal prediction of the Indian Ocean Dipole,” *Nat Commun*, vol. 13, no. 1, p. 7681, Dec. 2022, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41467-022-35412-0

[43] B. Pan, G. J. Anderson, A. Goncalves, D. D. Lucas, C. J. W. Bonfils, and J. Lee, “Improving Seasonal Forecast Using Probabilistic Deep Learning,” *Journal of Advances in Modeling Earth Systems*, vol. 14, no. 3, p. e2021MS002766, 2022, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2021MS002766. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1029/2021MS002766

[44] K. Sohn, H. Lee, and X. Yan, “Learning Structured Output Representation using Deep Conditional Generative Models,” in *Advances*

in Neural Information Processing Systems, vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://papers.nips.cc/paper_files/paper/2015/hash/8d55a249e6baa5c06772297520da2051-Abstract.html

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[46] A. Anwar, "What is Transposed Convolutional Layer?" Apr. 2021. [Online]. Available: https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11

[47] B. D. Ripley, *Pattern Recognition and Neural Networks.* Cambridge: Cambridge University Press, 1996. [Online]. Available: https://www.cambridge.org/core/books/pattern-recognition-and-neural-networks/4E038249C9BAA06C8F4EE6F044D09C5C

[48] C. M. Bishop, G. Hinton, C. M. Bishop, and G. Hinton, *Neural Networks for Pattern Recognition.* Oxford, New York: Oxford University Press, Nov. 1995.

[49] B. Venables and B. Ripley, "Modern Applied Statistics With S," in *Springer*, Jan. 2002, journal Abbreviation: Springer.

[50] C. Guo and F. Berkhahn, "Entity Embeddings of Categorical Variables," Apr. 2016, arXiv:1604.06737 [cs]. [Online]. Available: http://arxiv.org/abs/1604.06737

[51] Y. Wang, D. M. Blei, and J. P. Cunningham, "Posterior Collapse and Latent Variable Non-identifiability," Jan. 2023, arXiv:2301.00537 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2301.00537

[52] J. Lucas, G. Tucker, R. Grosse, and M. Norouzi, "Understanding Posterior Collapse in Generative Latent Variable Models," Apr. 2019. [Online]. Available: https://openreview.net/forum?id=r1xaVLUYuE

[53] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *International conference on learning representations*, 2016.

[54] A. A. Alemi, B. Poole, I. Fischer, J. V. Dillon, R. A. Saurous, and K. Murphy, "Fixing a Broken ELBO," Feb. 2018, arXiv:1711.00464 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1711.00464

[55] A. Asperti and M. Trentin, "Balancing Reconstruction Error and Kullback-Leibler Divergence in Variational Autoencoders," *IEEE Access*, vol. 8, pp. 199 440–199 448, 2020, conference Name: IEEE Access. [Online]. Available: https://ieeexplore.ieee.org/document/9244048

[56] H. Fu, C. Li, X. Liu, J. Gao, A. Celikyilmaz, and L. Carin, "Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing," Jun. 2019, arXiv:1903.10145 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1903.10145

[57] W. S. Cleveland, "Robust Locally Weighted Regression and Smoothing Scatterplots," *Journal of the American Statistical Association*, vol. 74, no. 368, pp. 829–836, Dec. 1979, publisher: Taylor & Francis _eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1979.10481038. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1979.10481038

[58] C. A. S. Coelho, B. Brown, L. Wilson, M. Mittermaier, and B. Casati, "Chapter 16 - Forecast Verification for S2S Timescales," in *Sub-Seasonal to Seasonal Prediction*, A. W. Robertson and F. Vitart, Eds. Elsevier, Jan. 2019, pp. 337–361. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128117149000164

[59] D. Freedman, R. Pisani, and R. Purves, "Statistics (international student edition)," *Pisani, R. Purves, 4th edn. WW Norton & Company, New York*, 2007.

[60] M. Zamo and P. Naveau, "Estimation of the Continuous Ranked Probability Score with Limited Information and Applications to Ensemble Weather Forecasts," *Math Geosci*, vol. 50, no. 2, pp. 209–234, Feb. 2018. [Online]. Available: https://doi.org/10.1007/s11004-017-9709-7

[61] M. Iturbide, J. M. Gutiérrez, L. M. Alves, J. Bedia, R. Cerezo-Mota, E. Cimadevilla, A. S. Cofiño, A. Di Luca, S. H. Faria, I. V. Gorodetskaya, M. Hauser, S. Herrera, K. Hennessy, H. T. Hewitt, R. G. Jones, S. Krakovska, R. Manzanas, D. Martínez-Castro, G. T. Narisma, I. S. Nurhati, I. Pinto, S. I. Seneviratne, B. van den Hurk, and C. S. Vera, "An update of ipcc climate reference regions for subcontinental analysis of climate model data: definition and aggregated datasets," *Earth System Science Data*, vol. 12, no. 4, pp. 2959–2970, 2020. [Online]. Available: https://essd.copernicus.org/articles/12/2959/2020/

[62] V. Godbole, G. E. Dahl, J. Gilmer, C. J. Shallue, and Z. Nado, "Deep learning tuning playbook," 2023, version 1. [Online]. Available: https://github.com/google-research/tuning_playbook

[63] D. Manubens-Gil, J. Vegas-Regidor, C. Prodhomme, O. Mula-Valls, and F. J. Doblas-Reyes, "Seamless management of ensemble climate prediction experiments on HPC platforms," in *2016 International Conference on High Performance Computing & Simulation (HPCS)*, Jul. 2016, pp. 895–900. [Online]. Available: https://ieeexplore.ieee.org/document/7568429

[64] T. Pelsmaeker and W. Aziz, "Effective Estimation of Deep Generative Language Models," May 2020, arXiv:1904.08194 [cs]. [Online]. Available: http://arxiv.org/abs/1904.08194

[65] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," Jun. 2018, arXiv:1201.0490 [cs]. [Online]. Available: http://arxiv.org/abs/1201.0490