



# Research internship

## Improving climate extremes prediction : downscaling approach

**Field of study :** Applied Mathematics and Machine learning

**Scholar year :** 2022-2023

### Confidentiality notice

Non-confidential report and publishable on Internet

**Author:** Marine Berthier

**Promotion:** 2024

**ENSTA Paris Tutor:**  
Zacharie Alès

**Host Organism Tutor:**  
Tom Beucler

**Host organism:** University of Lausanne  
**Address :** Bâtiment Géopolis, UNIL Dorigny 1015 Lausanne - Switzerland

# Acknowledgments

I would like to express my sincere gratitude to my supervisor Tom Beucler, director of the  $\partial^3$ AWN laboratory <sup>1</sup> and my co-supervisor Erwan Koch, director of ECCE, for guiding me through that intriguing topic, giving me support throughout my internship and and precious feedbacks on my report. I would also like to thank Valérie Chavez and Grégoire Mariethoz, members of ECCE, for the time they spent answering my questions and for allowing me to conduct this research project. Additionally, I am grateful for the financial support provided by FGSE and HEC, and to Sabrina Damiani for her administrative work.

My deep thanks also go to the laboratory members Milton Gomez, Saranya Ganesh and Frederick Tam. Their warm welcome and willingness to share their knowledges in Machine Learning have been precious to me.

I want to express my gratitude to the fellow interns and colleagues at the University of Lausanne for their help and for the great working atmosphere.

Lastly, I wish to thank Zacharie Ales for accepting to be my ENSTA tutor.

---

<sup>1</sup>Data-driven atmospheric and water dynamics

## Abstract

In an era when our global climate is undeniably warming and its consequences start to show off, the ability to anticipate and assess the risks associated with extreme climatic events is crucial for preserving human societies and ecosystems.

This research endeavors to address this imperative by employing predictive modeling techniques to enhance the prediction of extreme events, especially heavy rainfalls and storms. While statistical models, as described by Coles [6], have shed light on the behavior of precipitation maxima, current climate models still exhibit limitations in their resolution. In a context of predicting precipitation, local phenomena are at stake : summer precipitation maxima, generally come from convective thunderstorms that are small in scale. For that reason we need to combine maxima predictions with downscaling. Because standard interpolation algorithms fail to realistically capture the spatial distribution of precipitation, more complex statistical models and machine learning models will be explored. In this report I will explain how a methodology for that purpose has been created and show the first results on comparing the models using the Continuous Ranked Probability Score (CRPS).

**KEY WORDS** Downscaling, Extreme Value Theory, Generalized Extreme Value distribution, Parametric Distribution Prediction, Machine Learning, Vector Generalized Linear Model, Vector Generalized Additive Model

## Internship context

I was looking for an internship in climate science, with the desire to apply the statistical and probabilistic knowledge I acquired during my last semester focused on data science at ENSTA. I wrote to Tom Beucier who is working on Machine Learning in the atmospheric and weather science. He introduced me to the members of ECCE : Expertise Center for Climate Extremes, which was just about to be created. I became the first ECCE internship allowing me to work with Erwan Koch, its director. I was both advised in statistics by Erwan Koch and in machine learning and atmospheric science by Tom Beucier. This was also a great opportunity to work in an interdisciplinary center : with people from the business studies and from the environmental science fields. As part of the  $\partial^3$ AWN lab, I attended the weekly meeting where we dedicated an hour to discussing our latest research findings, addressing challenges we encountered, sharing papers, and making conversations about various topics related to machine learning. Those moments gave me very interesting insight into the field.

# Contents

|          |                                                                                                      |           |
|----------|------------------------------------------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                                                                  | <b>6</b>  |
| <b>2</b> | <b>Main concepts</b>                                                                                 | <b>7</b>  |
| 2.1      | Mathematical framework for extremes prediction . . . . .                                             | 7         |
| 2.1.1    | Extreme Value Theory . . . . .                                                                       | 7         |
| 2.1.2    | Finding the GEV parameters . . . . .                                                                 | 8         |
| 2.1.3    | What GEV enables . . . . .                                                                           | 8         |
| 2.2      | Downscaling . . . . .                                                                                | 8         |
| <b>3</b> | <b>Preliminaries</b>                                                                                 | <b>10</b> |
| 3.1      | Statistical methods . . . . .                                                                        | 10        |
| 3.1.1    | Generalized Linear Model . . . . .                                                                   | 10        |
| 3.1.2    | Generalized Additive Model . . . . .                                                                 | 10        |
| 3.1.3    | Vector Generalized Linear/Additive Model . . . . .                                                   | 11        |
| 3.2      | Neural Networks . . . . .                                                                            | 11        |
| 3.2.1    | Architectures of Neural Networks . . . . .                                                           | 12        |
| <b>4</b> | <b>Methodology</b>                                                                                   | <b>18</b> |
| 4.1      | Data processing . . . . .                                                                            | 18        |
| 4.2      | One metric to discriminate and to train the Neural Network : the CRPS                                | 20        |
| 4.2.1    | Parametric Distributional Prediction . . . . .                                                       | 20        |
| 4.3      | Downscaling approaches and targets . . . . .                                                         | 22        |
| 4.3.1    | Previous results on downscaling using interpolation . . . . .                                        | 22        |
| 4.3.2    | Downscaling precipitation maxima . . . . .                                                           | 23        |
| 4.4      | Chosen downscaling framework and interest variables . . . . .                                        | 24        |
| 4.5      | Statistical models . . . . .                                                                         | 24        |
| 4.6      | Navigating Neural Networks: challenges, insights, and solutions to<br>establish a protocol . . . . . | 26        |
| 4.6.1    | Training, validation and test set . . . . .                                                          | 29        |
| 4.7      | Tested architectures arising from these reflections . . . . .                                        | 29        |
| 4.7.1    | Hyperparameter settings . . . . .                                                                    | 31        |
| 4.8      | Transformers thoughts . . . . .                                                                      | 31        |
| <b>5</b> | <b>Results</b>                                                                                       | <b>32</b> |
| 5.1      | Statistical models . . . . .                                                                         | 32        |
| 5.2      | Hyperparameters choice . . . . .                                                                     | 35        |
| 5.3      | CRPS results . . . . .                                                                               | 38        |
| <b>6</b> | <b>Further work</b>                                                                                  | <b>41</b> |
| <b>7</b> | <b>Conclusion</b>                                                                                    | <b>42</b> |
| <b>8</b> | <b>Appendix</b>                                                                                      | <b>45</b> |
| 8.1      | CRPS Formula proof . . . . .                                                                         | 45        |
| 8.1.1    | First lemma . . . . .                                                                                | 45        |
| 8.1.2    | From the original definition of the CRPS . . . . .                                                   | 46        |

|     |                                          |    |
|-----|------------------------------------------|----|
| 8.2 | Hyperparameters search : U-Net . . . . . | 47 |
|-----|------------------------------------------|----|

# 1 Introduction

Acknowledging the limitations imposed by coarse-resolution climate models, researchers have been investigating the process, called downscaling, consisting of generating higher-resolution projections of climate data at a local or regional scale for decades. [4]. As the field of machine learning continues to advance, a diverse array of neural network architectures has been explored : from Artificial Neural Networks to Convolutional Neural Networks [13] and more state-of-the-art architectures like U-Net [14] and Transformers [3] [9]. Usually when researchers evaluated the models ability to predict extreme events, they used algorithms that predicted overall temperature trends and then singled out the extreme values. However, this approach has its limitations, as it might not fully capture the complex interactions and localized factors that contribute to the occurrence of extreme events. To address this shortcoming, my research internship proposes a novel methodology that focuses on forecasting extremes by directly targeting maximum values in a downscaling context. The combination of Extreme Value theory and machine learning is already under investigation [5], but in contrast to merely predicting maxima, our focus shifts towards examining the parameters governing the associated probabilistic distribution. They contain more information such as mean, standard deviation and quantiles. Although some previous studies have tried predicting distribution patterns [1] [2], my research stands out for exploring the Generalized Extreme Value distribution parameters. You will learn in this report what methodology has been developped and the first results obtained.

## 2 Main concepts

This section is dedicated to present the most important concepts that was at the core of my research.

### 2.1 Mathematical framework for extremes prediction

#### 2.1.1 Extreme Value Theory

Extreme Value Theory focuses on the statistical behaviour of the maxima  $M_n = \max(X_1, \dots, X_n)$  where  $X_1, X_2, \dots, X_n$  is a sequence of independent random variables that have the distribution function  $F$ . Its behaviour can be theoretically known :

$$\begin{aligned}\Pr \{M_n \leq z\} &= \Pr \{X_1 \leq z, \dots, X_n \leq z\} \\ &= \Pr \{X_1 \leq z\} \times \dots \times \Pr \{X_n \leq z\} \\ &= \{F(z)\}^n.\end{aligned}$$

where  $z \in \mathbb{R}$

However in practice we almost never know the distribution function  $F$ . It can be estimated empirically but small error on  $F$  can lead to significant discrepancies on  $F^n$ . Here comes the following powerful theorem :

If there exist sequences of constants  $\{a_n > 0\}$  and  $\{b_n\}$  such that

$$\Pr \{(M_n - b_n) / a_n \leq z\} \rightarrow G(z) \text{ as } n \rightarrow \infty,$$

where  $G$  is a non-degenerate distribution function, then  $G$  belongs to the **Generalized Extreme Value** (GEV) family of distributions, i.e.,  $G$  can be written

$$G(z) = \exp \left\{ - \left[ 1 + \xi \left( \frac{z - \mu}{\sigma} \right) \right]^{-\frac{1}{\xi}} \right\}$$

defined on the set  $\{z : 1 + \xi(\frac{z - \mu}{\sigma}) > 0\}$  where the parameters satisfy  $-\infty < \mu < \infty, \sigma > 0$  and  $-\infty < \xi < \infty$ .  $\mu$  is the location parameter,  $\sigma$  is the scale parameter and  $\xi$  is the shape parameter.

The normalizing constants  $\{a_n > 0\}$  and  $\{b_n\}$  are not a preoccupation as we have :

$$\Pr \{(M_n - b_n) / a_n \leq z\} \approx G(z)$$

for large enough  $n$ . Equivalently,

$$\begin{aligned}\Pr \{M_n \leq z\} &\approx G \{(z - b_n) / a_n\} \\ &= G^*(z),\end{aligned}$$

where  $G^*$  is another member of the GEV family. As a result the distribution of  $M_n$  belongs to the GEV family.

**The series of block maxima** The block maxima  $M_n$  is defined as  $\max(X_1, \dots, X_n)$  as we usually look for maxima over a certain period of time : week, month, year, ...  $X_i$  could be for example the hourly total of precipitations and our time window will be the week or the month. We then generate a series of  $m$  block maxima,  $M_{n,l}, \dots, M_{n,m}$ , to which the GEV distribution can be fitted.

### 2.1.2 Finding the GEV parameters

The easiest way to find the GEV parameters is to use the maximum likelihood estimation (there are some concerns about the regularity conditions that are explained in the book of Coles [6])

Under the assumption that  $Z_1, \dots, Z_m$  are independent variables following the GEV distribution, the log-likelihood for the GEV parameters when  $\xi \neq 0$  is

$$\begin{aligned} \ell(\mu, \sigma, \xi) = & -m \log \sigma - (1 + 1/\xi) \sum_{i=1}^m \log \left[ 1 + \xi \left( \frac{z_i - \mu}{\sigma} \right) \right] \\ & - \sum_{i=1}^m \left[ 1 + \xi \left( \frac{z_i - \mu}{\sigma} \right) \right]^{-1/\xi} \end{aligned}$$

provided that

$$1 + \xi \left( \frac{z_i - \mu}{\sigma} \right) > 0, \text{ for } i = 1, \dots, m.$$

The case  $\xi = 0$  requires separate treatment using the Gumbel limit of the GEV distribution and leads to the log-likelihood

$$\ell(\mu, \sigma) = -m \log \sigma - \sum_{i=1}^m \left( \frac{z_i - \mu}{\sigma} \right) - \sum_{i=1}^m \exp \left\{ - \left( \frac{z_i - \mu}{\sigma} \right) \right\}.$$

### 2.1.3 What GEV enables

**Having a distribution of probability for maxima is especially useful as it enables us to look for quantiles and for information about extremes.** Estimates of extreme quantiles of the weekly/monthly maximum distribution are then obtained by inverting the distribution function. We get

$$z_p = \begin{cases} \mu - \frac{\sigma}{\xi} [1 - \{-\log(1-p)\}^{-\xi}], & \text{for } \xi \neq 0 \\ \mu - \sigma \log\{-\log(1-p)\}, & \text{for } \xi = 0 \end{cases}$$

where  $G(z_p) = 1-p$ ,  $1/p$  is typically referred to as a return period and  $z_p$  the return level associated with the return period since to a reasonable degree of accuracy, the level  $z_p$  is expected to be exceeded on average once every  $1/p$  years. More precisely,  $z_p$  is exceeded by the weekly/monthly maximum with probability  $p$ .

## 2.2 Downscaling

In warm and humid regions, when warm air rises into the upper atmosphere, it cools and condenses, forming clouds. When both the air is moist enough and there are



enough updrafts, these clouds can develop into storms and heavy rainfalls.

**Convective storms happen at local scale.** To offer qualitative risk assessment, one needs precipitation at a high resolution. However Global Climate Models still have coarse spatial resolution e.g. 100 km for some of them. And those able to have better resolution require lots of time and resources to compute at better scale. Even when they managed to do it, they often struggle to represent certain types of extreme events.

Two primary methods exist to extract data at local levels from the global climate projections :

- numerical downscaling, often referred to as "dynamical downscaling," which employs a nested Regional Climate Model
- empirical downscaling, which involves statistical and machine learning methods

Empirical downscaling is usually more computationally efficient and has also been shown to reduce climate projections biases. It is important though to keep in mind that while downscaling methods provide valuable insights, **they tend to produce extreme events that are less severe than observed**, possibly due to assumptions of linearity and other factors [7].

We seek to determine the optimal approach for going from a 12 km resolution map with meteorological and topological variables to a refined 2 km resolution map detailing GEV parameters. This analysis involves a comparative assessment of statistical methodologies and Machine Learning techniques.

## 3 Preliminaries

In this preliminary section, I offer an overview of the foundational statistical and machine learning concepts that I learned during my internship and that are essential for understanding the contents of this report. They will all be useful for parameters distribution prediction and downscaling.

### 3.1 Statistical methods

#### 3.1.1 Generalized Linear Model

General linear model is about writing the response variable  $y$  as a linear combination of the predictors  $x_1, x_2, \dots$

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \epsilon$$

where  $\beta_0, \beta_1, \beta_2, \dots, \beta_n$  are the coefficients associated with predictor variables  $x_1, x_2, \dots, x_n$  and  $\epsilon \sim \mathcal{N}(0, \sigma^2)$  is the error term.

It is a powerful versatile tool in statistical modeling but happens to be unsuitable to certain types of modelisation like if the range of  $y$  is restricted (if we want to predict a binary variable). Enter GLM, Generalized Linear Model, that addresses these limitations.

A GLM is made up of 3 things:

- The distribution family of the response variable, which is assumed to be a member of the exponential family of distributions (like Gaussian, Bernoulli, and Poisson)
- A linear predictor :

$$\eta_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \epsilon$$

- A link function that links the mean,  $E(y_i) = \mu_i$ , to the linear predictors. For example for binomial we use the  $\text{logit}(p) = \ln\left(\frac{p}{1-p}\right)$  function. So we have the following equation for the GLM :

$$g(\mu_i) = \eta_i$$

The GLM's ability to accommodate different distributions and link functions allows it to handle a wide range of data types and model complex relationships effectively.

#### 3.1.2 Generalized Additive Model

Generalized Additive Models, GAMs, are statistical models that allow one to model non-linear data. While a multiple linear model can be written as

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p + \epsilon,$$

a GAM has the general formula =

$$y = \beta_0 + f_1(x_1) + f_2(x_2) + \dots + f_p(x_p) + \epsilon$$

Each linear component  $\beta_j x_j$  is replaced by a smooth nonlinear function  $f_j(x_j)$ .

GAMs strike a balance between simple linear regression models, which are easy to interpret but limited in capturing complex relationships, and more complex machine learning models like neural networks, which are powerful in predicting complex relationships but are harder to interpret. GAMs can capture nonlinear relationships while allowing for interpretability and inference in the model results.

### 3.1.3 Vector Generalized Linear/Additive Model

Vector Generalized Linear Models (VGLM) are an extension of the traditional Generalized Linear Models that provide a more flexible framework for analyzing complex data structures and non-constant variance. The biggest disadvantage of GLMs is that they assume a fixed dispersion parameter, while VGLMs allow the dispersion to vary for each observation, accommodating overdispersion or underdispersion that might be present in the data. VGLMs enable also the modeling of multiple response variables simultaneously, incorporating correlations between them using various working correlation structures.

Let's take  $\mathbf{y}$ , a  $Q$ -dimensional vector. VGLMs are defined through the model for the conditional density

$$f(\mathbf{y} \mid \mathbf{x}; \mathbf{B}) = g(\mathbf{y}, \eta_1, \dots, \eta_M)$$

for some known function  $g(\cdot)$ , where  $\mathbf{B} = (\beta_1 \beta_2 \dots \beta_M)$  is a  $d \times M$  matrix of regression coefficients to be estimated. We may also write  $\mathbf{B}^\top = (\beta_{(1)} \beta_{(2)} \dots \beta_{(d)})$  so that  $\beta_j$  is the  $j$ th column of  $\mathbf{B}$  and  $\beta_{(k)}$  is the  $k$ th row. The  $j$ th linear predictor is then

$$\eta_j = \beta_j^\top \mathbf{x} = \sum_{k=1}^d \beta_{(j)k} x_k, \quad j = 1, \dots, M.$$

Then the difference between VGLM and VGAM is the same as for GLM and GAM. VGAMs replace the linear functions by smoothers such as splines. Hence, the central formula is

$$\eta_i = \sum_{k=1}^d f_k(x_{ik})$$

where  $f_k(x_k) = (f_{k(1)}(x_k), \dots, f_{k(M_k)}(x_k))^\top$  is a vector of  $M_k$  smooth functions of  $x_k$ .

## 3.2 Neural Networks

A neural network is a computational model inspired by the structure and functioning of human brain. It consists of interconnected artificial neurons, also called units, organized in layers.

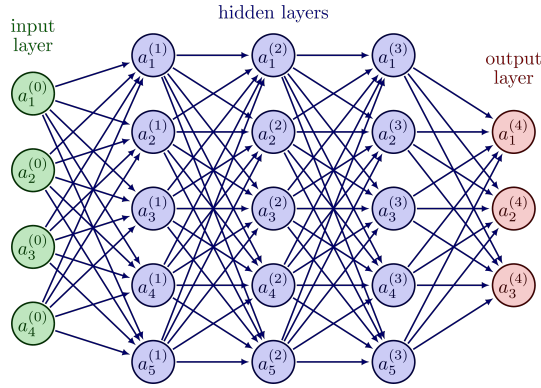


Figure 1: Basic architecture of a Neural Network

The most common type of neural network is the *feedforward neural network*, which consists of an input layer, one or more hidden layers, and an output layer. Each layer is composed of multiple neurons. Connections between neurons are represented by weights and each neuron takes a weighted sum of the inputs. Then an activation function (like a sigmoid, logistic or tanh function) is applied to the output of each neuron. Its goal is to control the output range (to constrain the values into a specific range), and also to introduce non-linearities as we want neural network to learn complex patterns. Afterwards the result of the activation function is passed to the next layer. This process is repeated layer by layer until the final output is generated.

During training, the weights and biases are adjusted based on a chosen optimization algorithm, such as gradient descent, to minimize the difference between the predicted outputs and the desired outputs. This process is typically performed using techniques such as backpropagation, which calculates and propagates the gradients backward through the network.

### 3.2.1 Architectures of Neural Networks

**Linear Neural Network** A linear neural network (LNN) is a neural network that only uses linear transformations in its layers. <sup>2</sup>

---

<sup>2</sup>For the visualization : <http://alexlenail.me/NN-SVG/index.html>

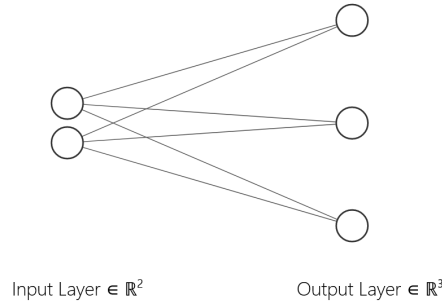


Figure 2: Architecture of the most simple Linear Neural Network, with 2 inputs (for example longitude and latitude) and 3 outputs (that could be the GEV parameters)

It can have several layers with different numbers of neurons.

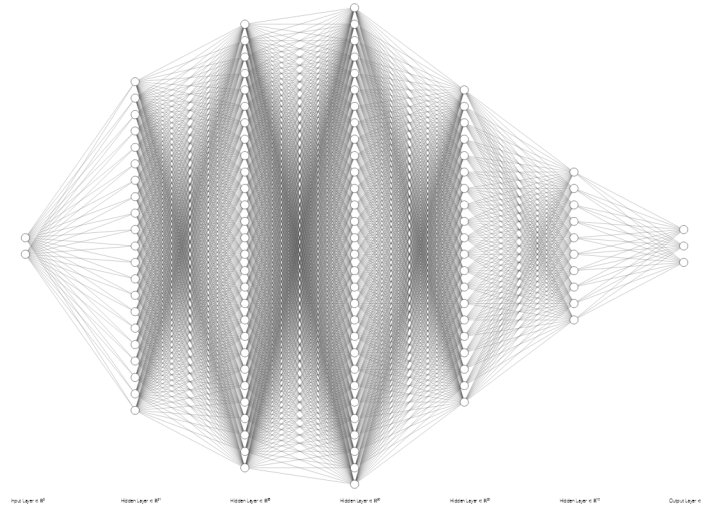


Figure 3: Linear Neural Network with several hidden layers

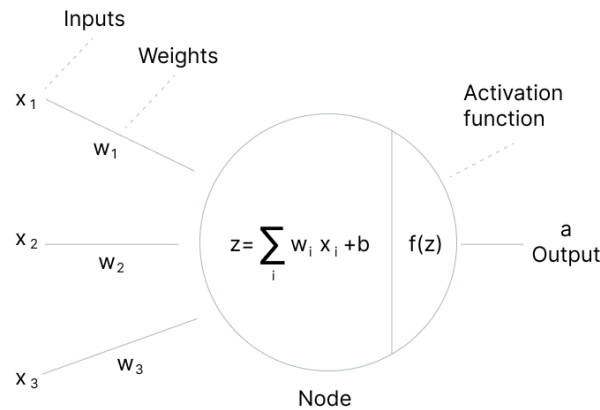
In a dense linear layer, each neuron is connected to every neuron in the previous layer. Mathematically, this operation can be represented as :  $\mathbf{y} = \mathbf{W} \mathbf{x} + \mathbf{b}$  where:

- $\mathbf{y}$  is the output of the linear layer
- $\mathbf{W}$  is a weight matrix that determines the strength of the connections between neurons
- $\mathbf{x}$  is the input to the linear layer
- $\mathbf{b}$  is a bias vector that is added element-wise

**In practice** Working with the GEV distribution requires to respect some constraints with the parameters : while  $\mu$  can be any real,  $\sigma$  must be positive and for precipitation we expect  $\xi$  to be between 0.1 and 0.7: so having  $\xi$  negative could lead to problem durant the training part. For that reason we will need activation functions.

**Multilayer Perceptron** Multilayer Perceptron is another type of artificial neural network that consists of multiple layers of interconnected artificial neurons. Compared to a LNN, activation functions are introduced.

For example to ensure that  $\sigma$  stays positive we add a Rectified Linear Unit, ReLU, activation function:  $ReLU(x) = \max(0, x)$ . Pytorch allows the customization of an activation layer, we can specify that only the scale parameter goes to a ReLU.



V7 Labs

Figure 4: Artificial Neural Network operations - drawing from V7 lab

**Convolutional Neural Network** The Convolutional Neural Network (CNN) is a type of neural network highly used in image recognition. The key component is the convolutional layer. It applies a set of learnable filters, also known as kernels, to the input image or feature map. These filters slide across the input, performing convolution.

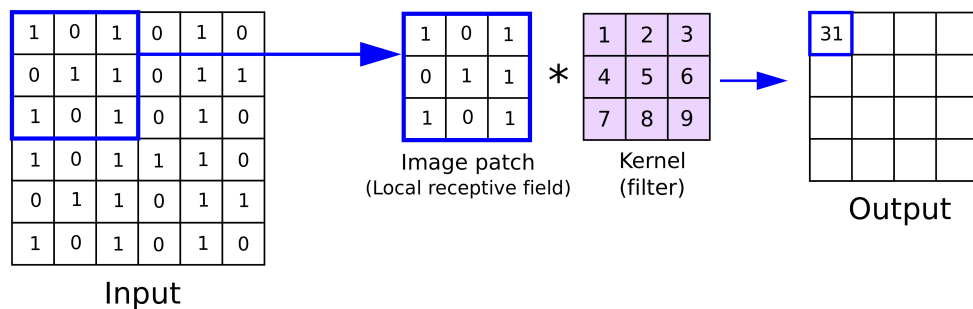


Figure 5: Kernel principle

Convolution enables us to extract local patterns and features by capturing spatial relationships between pixels.

Working with CNN requires working with channels which refers to the depth dimension of the input data. Each channel in a feature map captures specific patterns in

the input data. **In our case, channels will be precipitation, temperature, humidity ...** The number of channels is equal to the number of features.

Apart from those layers, it has pooling layers and fully connected layers. The pooling layer follows the convolutional layer and reduces the spatial dimension of the data. It aggregates neighboring values by taking the maximum (max pooling) or average (average pooling) within a defined window size as seen in the following pictures. By taking the maximum value within each pooling region, max pooling captures the most prominent feature present in that region.

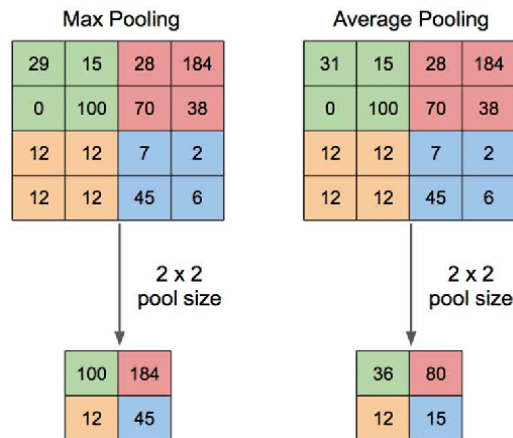


Figure 6: Max pooling principle

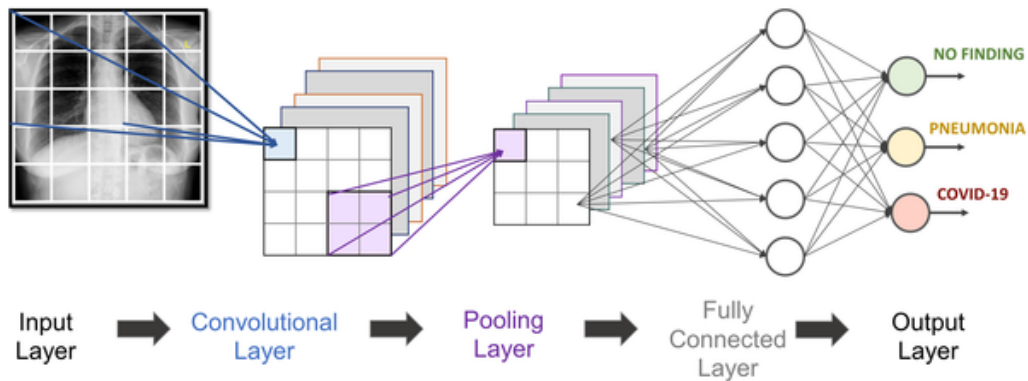


Figure 7: Classical CNN structure

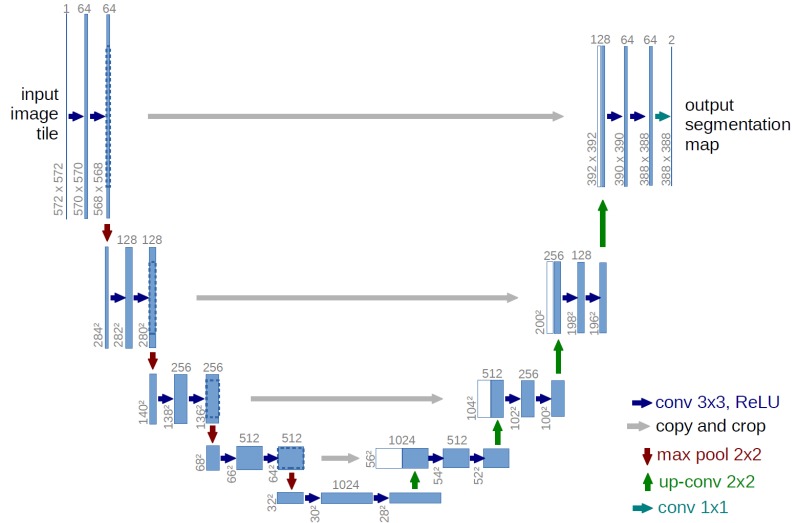


Figure 8: U-Net structure

**U-Net** The U-Net architecture is named after its shape, which resembles a U. It is a sort of CNN that consists of two main parts : a contracting path and an expanding path. The contracting path is responsible for extracting features from the input image, while the expanding path is responsible for reconstructing the image from the features.

**Transformer** A transformer is a state-of-the-art type of deep learning model architecture mostly used in Natural Language Processing. It was introduced in 2017 in the paper *Attention is all you need* [16]. It is composed of an encoder and a decoder.

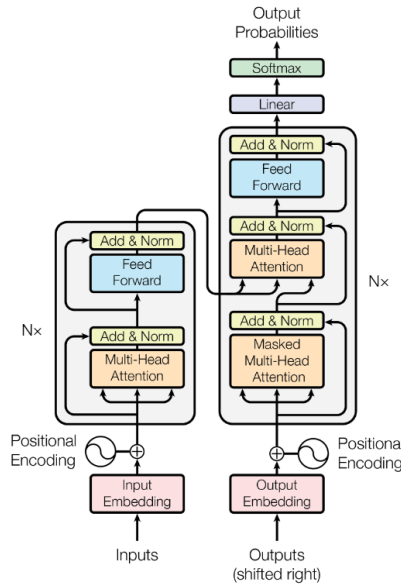


Figure 9: Transformer architecture from the paper

What is groundbreaking is the use of **self-attention mechanism** in both of them.



In the context of language processing self-attention allows the model to weigh the importance of different parts of the input sentence when generating the output. It can capture dependencies between words in a sentence, taking into account the contextual relationships between them.

It uses 3 matrix : the query, the key and the value. The query Q represents the piece of information that is seeking relevant information from the input sequence. The key K represents information that is used to retrieve relevant information from the input sequence. And the value represents the information associated with each key.

Then we compute the attention that is mathematically written as :

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V.$$

Where  $\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_{j=1}^N e^{x_j}}$

This step is done by the scaled dot-product attention and then the self-attention mechanism happens in the Multi-Head Attention block :

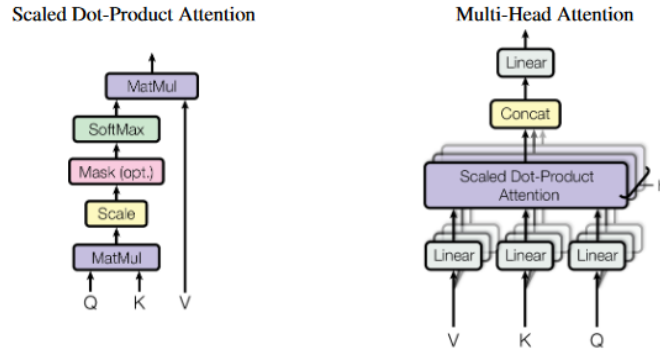


Figure 10: Transformer architecture from the paper

## 4 Methodology

### 4.1 Data processing

**Data type** Thanks to David Leutwyler who works at MeteoSwiss, we obtained 11 years of simulated data in the past, from 1999 to 2009 and 11 years in the future, from 2079 to 2089 that were used for his collaboration in the paper 'Clouds in Convection-Resolving Climate Simulations Over Europe' [11]. Those are from HadGEM-driven COSMO simulation, a type of atmospheric simulation that uses two different climate models to produce its results. The COSMO (Consortium for Small-scale Modeling) regional climate model is used to simulate the weather in a specific region, while the HadGEM (Hadley Centre Global Environmental Model) global climate model is used to simulate the weather for the entire globe. The model gives hourly data at a 2 km resolution. Along with the coordinates : time, rotated latitude and longitude (latitude and longitude values in the rotated coordinate system), we have 28 variables that display several climatic variables such as temperature at 2 meters high, relative humidity or water vapour that depend on time, lat and lon.

**Data format** The hourly data are saved in a .netCDF format which is a file format highly used for scientific data as it allows the storage of multidimensional data. It is handled by the xarray library of Python. As the data resolution is 2 km, it requires a large amount of memory, 1.9Tb for each year. The data are stored on a cluster called Curnagl. Working with them requires working on the server. After setting an interactive session, code is made with JupyterNotebook.

For downscaling purposes, we only keep 3 variables from the 28 displayed : T\_2M, the 2 meters surface temperature, TOT\_PR the hourly total of the precipitation and RELHUM\_2M the relative humidity at 2 meters high. The measures are done at 2 meters high to minimize the influence of the ground surface (grass, concrete, water,...) on the measurement.

**Variable extraction** Using Xarray's structures, we extract monthly precipitation maximum while also recording the temperature and humidity conditions during these peak precipitation periods. Looking at the data from the past, we have an 11-year dataset with a spatial resolution of 2 km on the Swiss map, producing 11 maxima for a particular month along with associated 11 temperature and humidity conditions.

Our aim is to forecast the distribution for a specific month. However, due to the limited sample size of only 11 instances, drawing probabilistic inferences becomes challenging. The dataset lacks the necessary sample numbers to accurately determine distribution parameters or to effectively compare with the Continuous Ranked Probability Score (CRPS). Therefore I combined two months so we can have a reasonable number of samples and so saying that July and August maxima belong to the same GEV distributions and trying to predict them using the temperature and humidity conditions observed during those months.

It is important to cut the year into seasons/2-months and not to look at all the maxima as in the GEV definition,  $X_i$  must be stationnary and precipitation depends

a lot of the time of the year.

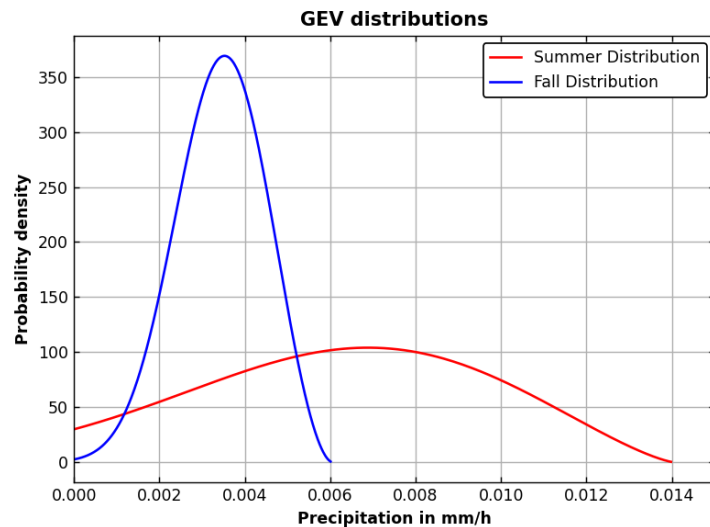


Figure 11: Change in the distribution regarding the season

To make our model generalizable to Switzerland's topography, we decided to add altitude data from the NASA Shuttle Radar Topography Mission (SRTM). I accessed these data through Open Elevation API, a free and open-source elevation API that is easy to set up in Python. I considered first downloading and processing the data myself, but Open Elevation API made the process much simpler.

**Area of interest** We focus on all Switzerland for the study but still work in a rectangle.

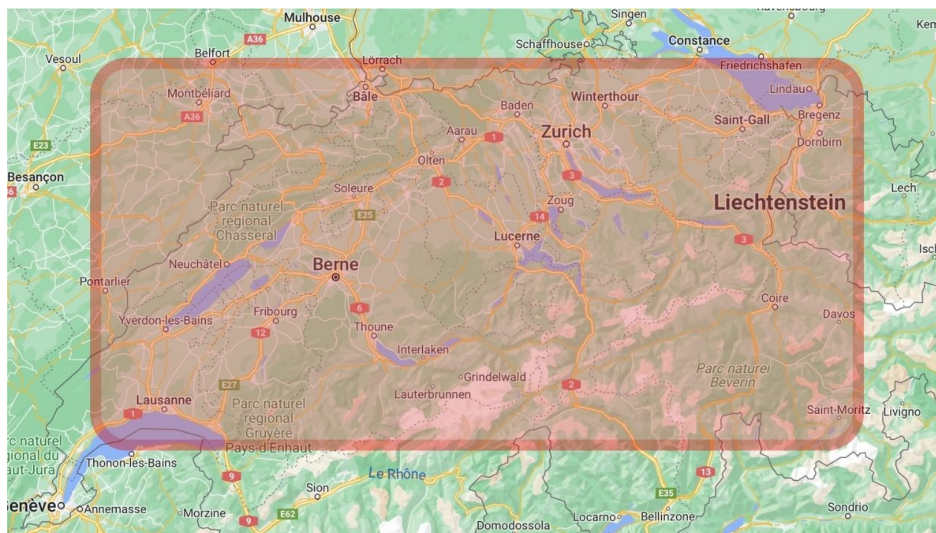


Figure 12: Spatial area of interest

## 4.2 One metric to discriminate and to train the Neural Network : the CRPS

### 4.2.1 Parametric Distributional Prediction

Parametric Distributional Prediction aims to predict the parameters of a probability distribution and that's what our neural networks are going to be trained for. During the training phase, we will have a probabilistic approach meaning we will customize a loss function that maximizes the likelihood or minimizes the distributional differences.

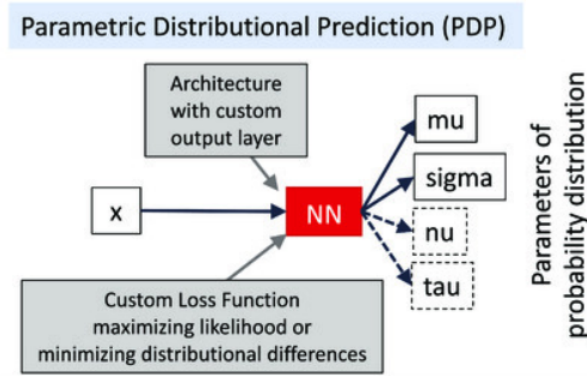


Figure 13: PDP principle

**Minimizing distributional differences : the CRPS** The Continuous Ranked Probability Score (CRPS) is much used for Bayesian machine learning models where the predictions are often not point-wise estimates but distributions. The expression for comparing a single ground truth value  $y$  to a Cumulative Distribution Function  $F$  is given by

$$\text{CRPS}(F, y) = \int_{-\infty}^{\infty} (F(x) - \mathbb{1}(x \geq y))^2 dx$$

where  $y$  is the ground truth value,  $F$  is the predicted CDF,  $\mathbb{1}(x \geq y)$  is the indicator function that equals 1 if  $x \geq y$  and 0 otherwise, and the integral is taken over the entire real line.

The CRPS happens to be a more state-of-the-art loss function and sometimes outperform the maximization of likelihood according to recent papers. [8]

When looking for the easiest way to implement the CRPS, I found an expression that directly linked the CRPS to the GEV parameters. [12]

$$\text{CRPS}(F_{\xi}, y) = \begin{cases} -y - 2 \text{Ei}(\log F_{\xi}(y)) + \gamma - \log 2, & \xi = 0 \\ y(2F_{\xi}(y) - 1) - 2G_{\xi}(y) - \frac{1 - (2 - 2^{\xi})\Gamma(1 - \xi)}{\xi}, & \xi \neq 0 \end{cases}$$

where

$$\text{Ei}(x) = \int_{-\infty}^x \frac{e^t}{t} dt$$

$$\text{CRPS}(F_{\xi,\mu,\sigma}, y) = \sigma \text{CRPS}\left(F_{\xi}, \frac{y - \mu}{\sigma}\right).$$

The CDFs and otherwise required functions are given by  $F_{\xi,\mu,\sigma}(x) = F_{\xi}\left(\frac{x-\mu}{\sigma}\right)$  for  $\xi = 0$  :  $F_{\xi}(x) = \exp(-\exp(-x))$

$$\begin{aligned} \text{for } \xi > 0 : \quad F_{\xi}(x) &= \begin{cases} 0, & x \leq -\frac{1}{\xi}, \\ \exp\left(-(1 + \xi x)^{-1/\xi}\right), & x > -\frac{1}{\xi}, \end{cases} \\ G_{\xi}(x) &= \begin{cases} 0, & x \leq -\frac{1}{\xi}, \\ -\frac{F_{\xi}(x)}{\xi} + \frac{\Gamma_u(1-\xi, -\log F_{\xi}(x))}{\xi}, & x > -\frac{1}{\xi}, \end{cases} \\ \text{for } \xi < 0 : \quad F_{\xi}(x) &= \begin{cases} \exp\left(-(1 + \xi x)^{-1/\xi}\right), & x < -\frac{1}{\xi}, \\ 1, & x \geq -\frac{1}{\xi}, \end{cases} \\ G_{\xi}(x) &= \begin{cases} -\frac{F_{\xi}(x)}{\xi} + \frac{\Gamma_u(1-\xi, -\log F_{\xi}(x))}{\xi}, & x < -\frac{1}{\xi}, \\ -\frac{1}{\xi} + \frac{\Gamma(1-\xi)}{\xi}, & x \geq -\frac{1}{\xi}. \end{cases} \end{aligned}$$

Unfortunately when I tried to implement it with PyTorch, I needed the gamma function, which is not available in PyTorch. This meant that there was no way to compute gradients, and even when I tried to auto-implement the function, it was still a large and unwieldy loss function.

For this reason, I demonstrate a CRPS formula that uses the expectation values and is inspired by the Cira TensorFlow implementation [10].

$$\text{CRPS}(F, y) = \mathbb{E}_F |Y - y| - \frac{1}{2} \mathbb{E}_F |Y - Y'|$$

However this one implies comparing two distribution samples. The chosen neural network gives as output the three parameters of the GEV, we then use them to create samples of the corresponding GEV distribution. Pytorch requires to be careful when creating the samples : as for the precedent CRPS form, the computation of the gradient requires the loss function to be the result of differentiable operations. For that reason we are not using pre-created libraries that give sample of GEVs like genextreme from scipy.stats but as we know the parameters of the distribution, we know the cumulative distribution function  $F$  and so we can sample values from  $X$  using a uniform random variable  $U$ .

$$X = F^{-1}(U)$$

The inverse cumulative distribution function for a GEV distribution with parameters  $\xi$ ,  $\mu$  and  $\sigma$  can be expressed as:

$$F^{-1}(p) = \begin{cases} \mu + \frac{\sigma}{\xi} [(-(-\log p)^{-\xi} - 1)], & \text{if } \xi \neq 0 \\ \mu - \sigma \log(-\log p), & \text{if } \xi = 0 \end{cases}$$

where  $0 < p < 1$  is the probability and  $F^{-1}(p)$  is the value corresponding to the quantile associated with  $p$ .

### 4.3 Downscaling approaches and targets

There are two main approaches in empirical downscaling : **Super resolution** and **Perfect Prognosis**

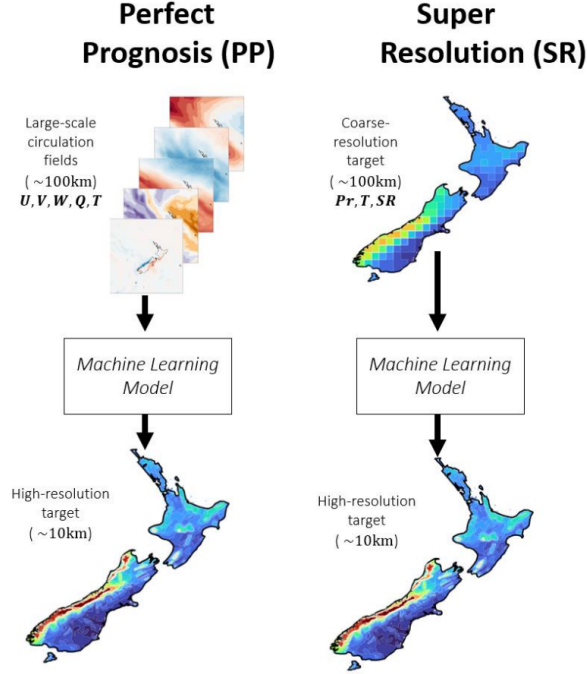


Figure 14: Empirical downscaling main approaches

Super-resolution is done by taking low-resolution climate variable and using a neural network to learn the relationship between the low-resolution variable and high-resolution variable. On the other hand Perfect Prognosis tries to build relationships between large-scale atmospheric variables and local weather. In order to predict heavy rainfall at a 2 km resolution : Super Resolution will use heavy rainfall at a 12 km while Perfect Prognosis will use temperature, relative humidity, pressure, ...

#### 4.3.1 Previous results on downscaling using interpolation

The last summer intern implemented three interpolation techniques to do downscaling on precipitation, temperature and humidity.

- Bilinear interpolation  $f_{\text{bilinear}}(x, y) = axy + b_1x + b_2y + c$  is the 2-dimensional version of linear interpolation
- Bicubic interpolation is taking a cubic function in 2 dimensions  $f_{\text{bicubic}}(x, y) = \sum_{i=0}^3 \sum_{j=0}^3 a_{ij}x^i y^j$
- Quintic interpolation  $f_{\text{quintic}}(x, y) = \sum_{i=0}^5 \sum_{j=0}^5 a_{ij}x^i y^j$

He implemented 6 metrics : RMSE, MAE, Structural Similarity Index (SSIM), Hellinger distance, Perkins Skill Score and temporal and spatial autocorrelation. The results showed that the quintic interpolation was the best of the three.

Though it was made to serve as baseline, it helps understand the first issues to tackle with Super-Resolution. I adapted his code to downscale the precipitation maxima of a particular month and year.

### 4.3.2 Downscaling precipitation maxima

Let's consider the 2 km-resolution map depicting hourly precipitation maxima for the month of July in 2008, which is visually represented in the graph labeled as "Original Data." Some maxima can exceed 40 mm/h.

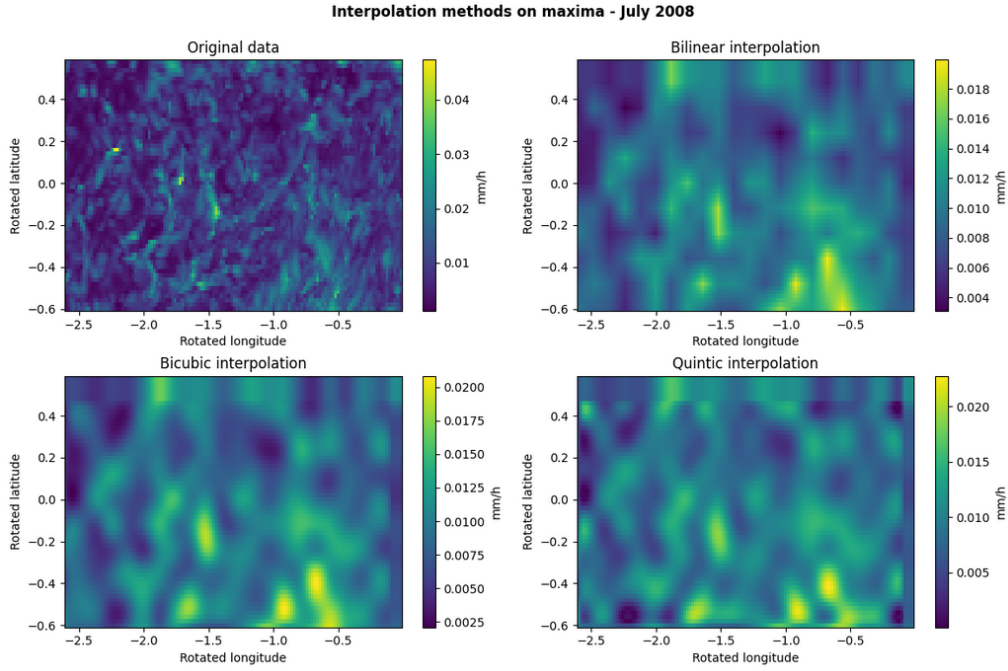


Figure 15: Downscaling of maxima

Subsequently, the grid is rescaled to a 12 km resolution and downscaling is performed. An observation from the change in the abscissa is the omission of the most remarkable maxima (ranging from 30 to 40mm/h) throughout the process.

The downscaling procedure is then applied to all monthly maxima recorded during July and August across the 11 years. The reduction in local extreme events is clearly apparent when examining the boxplot presentation of some grid points.

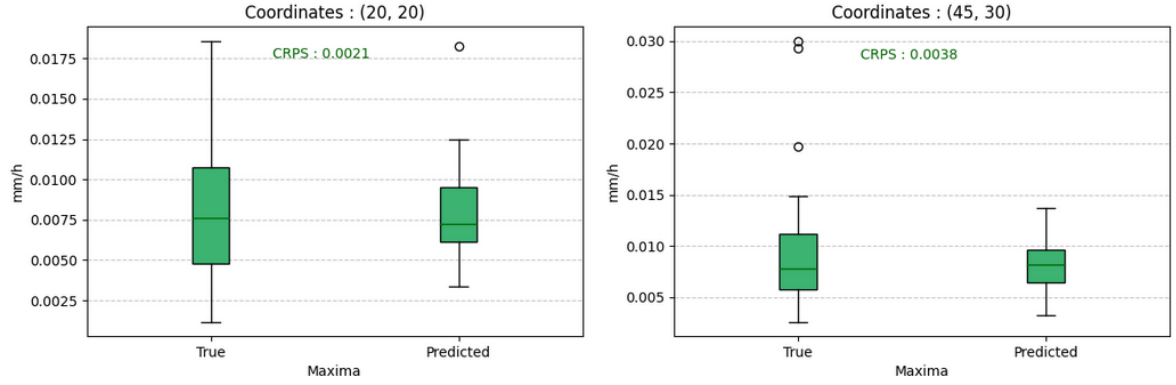


Figure 16: Boxplots of the maxima precipitation over July and August

#### 4.4 Chosen downscaling framework and interest variables

To tackle the loss of extreme phenomena, we will focus on Perfect Prognosis. As the goal of this project is to defined a clear methodology, I selected only temperature and relative humidity as interested meteorological variables to predict the 3 GEV parameters. The topological ones are longitude, latitude and altitude.

**What temperature and humidity should we take into account ?**

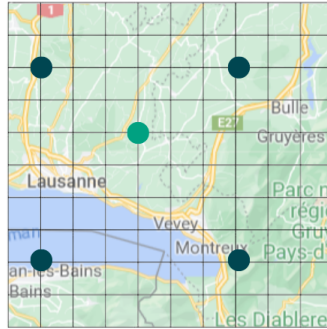


Figure 17: Boxplots of the maxima precipitation over July and August

Since GLM, VGLM, VGAM and ANNs can't take a high number of covariates, we can't give them the information of all the 12 km resolution grid-points as input. **The selection of neighboring data points to incorporate is a crucial parameter we will explore starting when choosing which model to choose.**

#### 4.5 Statistical models

**Using R :** The models described in 3.1 are fitted by using R, and different packages : **SpatialExtremes** and **VGAM**. The GLM model is fitted by using the *fitspatgev* function which calculates the linear coefficient by maximizing the maximum likelihood.



More precisely, we tried the following relationships between the location parameter and the predictors.  $\text{temp}_1$  is a reference to the temperature of the closest neighbors,  $\text{temp}_2$  to the second closest, ...

| Model's name                          | Formula                                                                                                                                                                                                                                                                     |
|---------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Basic stationary model                | $\text{loc} \sim \text{rlon} + \text{rlat} + \text{alt}$                                                                                                                                                                                                                    |
| Adding temperature                    | $\text{loc} \sim \text{rlon} + \text{rlat} + \text{alt} + \text{temp}_1$                                                                                                                                                                                                    |
| One neighbor                          | $\text{loc} \sim \text{rlon} + \text{rlat} + \text{alt} + \text{temp}_1 + \text{hum}_1$                                                                                                                                                                                     |
| One neighbor and normalized variables | $\text{loc} \sim \text{rlon} + \text{rlat} + \text{alt} + \text{temp\_norm}_1 + \text{hum\_norm}_1$                                                                                                                                                                         |
| One neighbor and splines              | $\text{loc} \sim \text{s}(\text{rlon}) + \text{s}(\text{rlat}) + \text{s}(\text{alt}) + \text{s}(\text{temp}_1) + \text{s}(\text{hum}_1)$                                                                                                                                   |
| Two neighbors                         | $\text{loc} \sim \text{rlon} + \text{rlat} + \text{alt} + \text{temp}_1 + \text{hum}_1 + \text{temp}_2 + \text{hum}_2$                                                                                                                                                      |
| Two neighbors and splines             | $\text{loc} \sim \text{s}(\text{rlon}) + \text{s}(\text{rlat}) + \text{s}(\text{alt}) + \text{s}(\text{temp}_1) + \text{s}(\text{hum}_1) + \text{s}(\text{temp}_2) + \text{s}(\text{hum}_2)$                                                                                |
| Four neighbors                        | $\text{loc} \sim \text{rlon} + \text{rlat} + \text{alt} + \text{temp}_1 + \text{hum}_1 + \text{temp}_2 + \text{hum}_2 + \text{temp}_3 + \text{hum}_3 + \text{temp}_4 + \text{hum}_4$                                                                                        |
| Four neighbors and splines            | $\text{loc} \sim \text{s}(\text{rlon}) + \text{s}(\text{rlat}) + \text{s}(\text{temp}_1) + \text{s}(\text{hum}_1) + \text{s}(\text{temp}_2) + \text{s}(\text{hum}_2) + \text{s}(\text{temp}_3) + \text{s}(\text{hum}_3) + \text{s}(\text{temp}_4) + \text{s}(\text{hum}_4)$ |

Table 1: Summary of tested formulas with corresponding models

All the models are tried with the GLM, VGLM and VGAM.

Here is for example a relation with the predictors that  $\mu$  can have, like for a GAM :

$$\mu(x) = \beta_0 + \beta_1 f_1(\text{lon}) + \beta_2 f_2(\text{lat}) + \beta_3 f_3(\text{alt}) + \beta_4 f_4(\text{temp}) + \beta_5 f_5(\text{hum})$$

**Adding scale depending of predictors :** The location parameter was the most important in the first approach. However in a case of parameter distribution prediction, it would be interesting to capture both the central tendency (mean) and the spread (standard deviation). That is what we are going to do with the neural networks, so to be able to compare them with statistical approaches we will also predict the scale parameter with statistical models.

**Criteria for model selection :** The different models are then compared by using the TIC criteria. The **Takeuchi Information Criterion (TIC)** [15] is used to evaluate and compare the quality of two statistical models. The TIC is defined by:

$$\text{TIC} = -2 \log(L) + k \log(n)$$

where  $L$  is the likelihood function of the model (that should be the highest possible),  $k$  is the number of estimated parameters in the model and  $n$  is the sample size, i.e., the number of observations.

The TIC is a goodness-of-fit measure that takes into account both the model's fit to the data (with the likelihood) and its complexity (with the second term). The goal is to **minimize the TIC**. A model with a lower TIC is preferred as it manages to explain the data well with a reasonable number of parameters.

**CRPS** After finding the best statistical model we calculate the associated CRPS on the validation set. We get the location, shape, and scale from the model and have to be careful at the link function used before getting them from R to Python.

## 4.6 Navigating Neural Networks: challenges, insights, and solutions to establish a protocol

With the dataset prepared and the theoretical framework established, another challenge was devising a suitable protocol for training the neural networks. Several issues emerged that lead to discussions.

**Choice of outputs** Initially, our objective centered around predicting the three generalized extreme value (GEV) distribution parameters and using them to sample from the distribution to apply the CRPS.

My initial approach involved using a LNN for parameter prediction. This approach produced NaN (Not-a-Number) values rapidly because of the lack of activation function to contain the value of  $\xi$ . Indeed as we have :

$$F^{-1}(p) = \begin{cases} \mu + \frac{\sigma}{\xi} [(-(-\log p)^{-\xi} - 1)] , & \text{if } \xi \neq 0 \\ \mu - \sigma \log(-\log p), & \text{if } \xi = 0 \end{cases}$$

the value of  $\xi$  could easily lead the samples to be infinite and the operations to conduct to NaN.

However it turns out that  $\xi$  parameter is usually set to a constant in statistical model due to the huge incertitude over this parameter. The difficulty to obtain  $\xi$  with maximum likelihood estimation is showed when trying to predict the shape parameter in R, it leads to warnings or even bugs.

```
> gev_fit <- gev.fit(mat_precip, show = FALSE)
Error in optim(init, gev.lik, hessian = TRUE, method = method, control = list(maxit = maxit,
  valeur non-finie fournie par optim
In addition: Warning message:
In sqrt(6 * var(xdat)) : NaNs produced
```

Figure 18: Error while using R

This graph shows that it has little effect on the mean and standard deviation of the distribution.

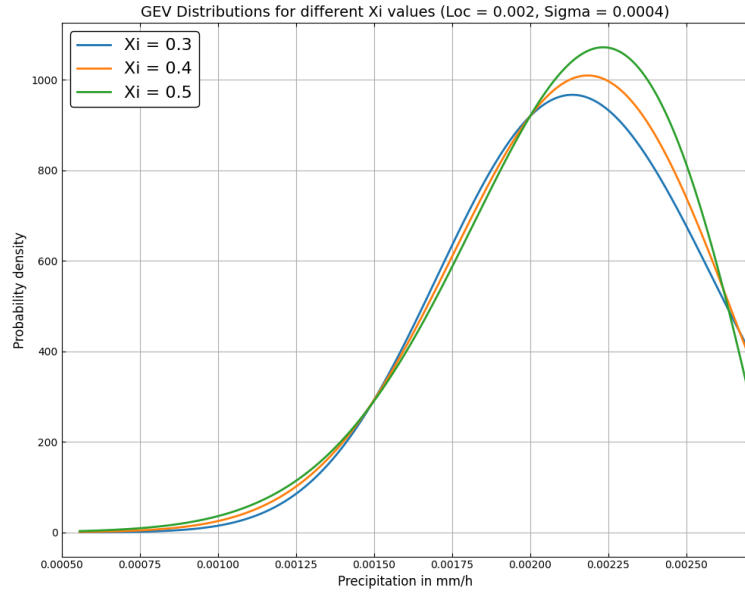


Figure 19: Behaviour of the distribution for different shape parameters

Given this information, what mostly matters is the prediction of the location and scale parameters and set the shape parameter at its stationary value (when we look at the monthly maxima over all the month of the 10 years data).

The ReLU activation function was put at the end of the NN to ensure that the scale parameter is positive and the first results came out. However sometimes the outputs were fully null tensors. In fact the CRPS happens to be a bit tricky :

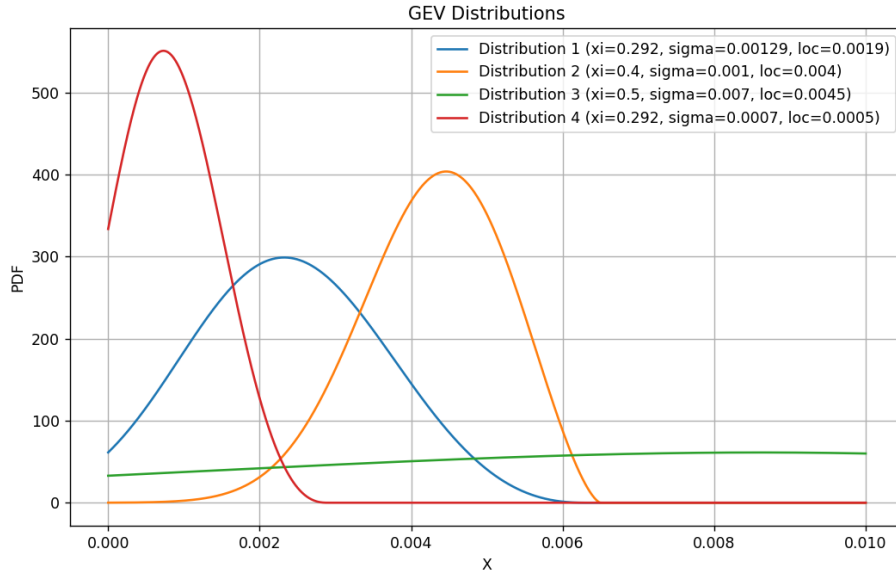


Figure 20: Different types of distribution

The blue distribution is said to be the reference distribution. I compare the obtained CRPS with samples from the same distribution, the second, the third, the fourth and with samples that are only zeros.

|      | Distribution 1 | Distribution 2 | Distribution 3 | Distribution 4 | Null samples |
|------|----------------|----------------|----------------|----------------|--------------|
| CRPS | 0.00052        | 0.00135        | 0.00365        | 0.00168        | 0.00304      |

Table 2: CRPS Compared to Different Distributions

The set of null samples achieved a higher score than the set derived from the third distribution. While it holds true that the third distribution lacks any resemblance to the blue one, in the context of precipitation, preference leans towards having non-zero samples. The fact that the zero-sample got a better score affects the training of the neural network. Specifically, due to the application of the ReLU activation function, the initial samples may assume partial null values. Given the satisfactory score achieved, the optimization phase may struggle to guide these samples away from the minimum associated with the zero-samples.

To overcome the problem, we decide to predict two quantiles of the distribution and then get back to the parameters :

$$q_1 = \exp \left\{ - \left[ 1 + \xi \left( \frac{x_1 - \mu}{\sigma} \right) \right]^{-1/\xi} \right\}$$

$$q_2 = \exp \left\{ - \left[ 1 + \xi \left( \frac{x_2 - \mu}{\sigma} \right) \right]^{-1/\xi} \right\}$$

So let's say we are taking the 90<sup>th</sup> and the 95<sup>th</sup> quantiles (which will be useful in an extreme event context).

For the location parameter  $\mu$ :

$$\mu = q_2 - \frac{\sigma}{\xi} \left[ 1 - (-\log(1 - 0.9))^{-\xi} \right]$$

For the scale parameter  $\sigma$ :

$$\sigma = \frac{q_2 - q_1}{(-\log(1 - 0.9))^{-\xi} - (-\log(1 - 0.95))^{-\xi}}$$

In this context, the training works.

#### 4.6.1 Training, validation and test set



Figure 21: First training -validation - test sets

The training-validation-test split derives from a partitioning of Switzerland into three distinct geographical regions. The training set represents a significant portion of the nation's surface, focusing on the western region. This allocation ensures a comprehensive incorporation of prevalent patterns : lakes, mountains, city, ... An additional 15% of the territory is assigned to the validation set. The remaining constitutes the test set, reserved for the final evaluation of the model's generalizability to unseen data instances.

#### 4.7 Tested architectures arising from these reflections

For the baseline we start by only using topological variables : rlon, rlat and altitude to predict the location and the scale parameters. Then we add the closest neighbor and its temperature and humidity conditions, then add the second and the fourth.

**With the following activation functions : ReLU - Sigmoid - ReLU - Abs.**

As I said before 3.2.1, they are essential to keep the parameters of the GEV in the right set  $z : 1 + \xi\left(\frac{z-\mu}{\sigma}\right) > 0$ .

**The optimizer used is Adam and the learning rate is 0.0001.**

| Model   | Input                                                                                             |
|---------|---------------------------------------------------------------------------------------------------|
| Model 0 | rlon, rlat, alt                                                                                   |
| Model 1 | rlon, rlat, alt, temp <sub>1</sub> , hum <sub>1</sub>                                             |
| Model 2 | rlon, rlat, alt, temp <sub>1</sub> , hum <sub>1</sub> , temp <sub>2</sub> , hum <sub>2</sub>      |
| Model 3 | rlon, rlat, alt, temp <sub>1</sub> ...,temp <sub>4</sub> , hum <sub>1</sub> ..., hum <sub>4</sub> |

Table 3: ANN architectures

Here is the representation of Model 1

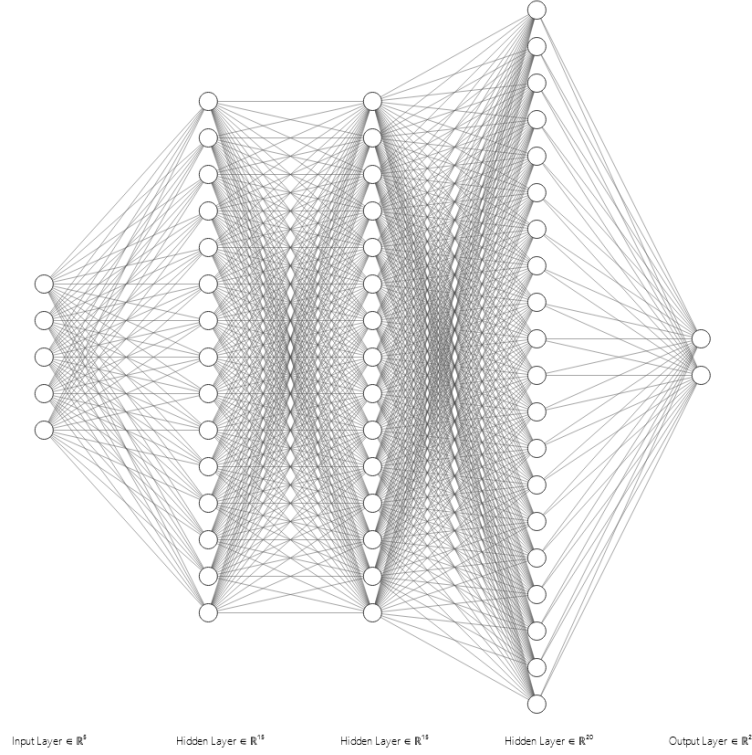


Figure 22: Model 1 architecture

**U-Net inspiration** To build the CNN I used the U-Net architecture as inspiration and it gave this resulting architecture, where the input is the 12km resolution map of temperature and humidity and the output is a 2km resolution map of the location and scale parameter of the GEV.

| Layer                       | Description                                                                                                  |
|-----------------------------|--------------------------------------------------------------------------------------------------------------|
| Encoder - Conv2D 1          | Input: <code>in_channels</code> , Output: 20, Kernel Size: 4, Stride: 2, Padding: 1<br>Activation: ReLU      |
| Encoder - Conv2D 2          | Input: 20, Output: 20, Kernel Size: 4, Stride: 1, Padding: 1                                                 |
| Decoder - ConvTranspose2D 1 | Input: 20, Output: 20, Kernel Size: 4, Stride: 2, Padding: 0                                                 |
| Decoder - ConvTranspose2D 2 | Input: 20, Output: <code>out_channels</code> , Kernel Size: 4, Stride: 3, Padding: 1                         |
| Decoder - ConvTranspose2D 3 | Input: <code>out_channels</code> , Output: <code>out_channels</code> , Kernel Size: 4, Stride: 2, Padding: 0 |

Table 4: Description of Layers in Conv-UNet Architecture

### 4.7.1 Hyperparameter settings

Hyperparameters are the settings of a machine learning model that are not learned from the data. Tuning hyperparameters is the process of finding the best set of hyperparameters for a given model and dataset. It is done thanks to the validation set that the model has never encountered before. This can be a time-consuming process, but it is important to do it in order to get the best possible performance from the model.

Here are the tunable parameters in this particular case

- The learning rate, usually between 0.1 and 0.00001, that controls the speed of the learning part. It determines how much the model's weights are updated after each training iteration.
- The number of layers and neurons. It determines the complexity of the model however too many layers can lead to overfitting and difficult training.
- The number of neighbors taken into account. As for the statistical models, it is possible to try a different number of neighbors to increase the complexity of the problem.
- The quantiles. The first quantiles idea was to take the 90<sup>th</sup> and the 95<sup>th</sup> for extreme studies purposes but others can be tested : 50<sup>th</sup> and 75<sup>th</sup>, 25<sup>th</sup> and 75<sup>th</sup>.
- The activation functions.

To find the best set one can use random search : trying first to find the best learning rate with random values, then the number of layers, ... It is the fastest way but not the most efficient. To enhance the tuning, one can do grid search and thus find the best set of hyperparameters but it's computationally expensive.

## 4.8 Transformers thoughts

Although I did not have enough time to train the transformer neural network I coded, I will share some thoughts we had about positional encoding implementation. I found that some people on GitHub and YouTube use embedding layers with learnable parameters to do positional encoding. However, I believe that this is not the best way to do it, as it requires the network to learn the positional encoding. Instead, I stucked to the method used by the original authors of the transformer neural network, which is to use sinusoidal positional encoding. In our case, we have a 2D map, so we can use a 2D sinusoidal positional encoding. We can also try using the product of sine and cosine to encode the number of points in the grid. We also looked at diffusion models and LSTM models to downscale the image. However, I decided to try a encoder-MLP architecture for my first attempt.

## 5 Results

### 5.1 Statistical models

**GLM with fitspatgev :** Here, I present a selection of outcomes generated by R during the model fitting process. It shows the spectrum of coefficients within the models, while also displaying the consistent values assigned to scale and shape parameters.

```

Model: Spatial GEV model
Deviance: -1384692
TIC: -1383911

Location Parameters:
locCoeff1 locCoeff2 locCoeff3 locCoeff4
5.622e-03 5.249e-05 -6.137e-05 -6.487e-09
Scale Parameters:
scaleCoeff1
0.003339
Shape Parameters:
shapeCoeff1
0.1548

```

Figure 23:  $\mu \sim \text{rlon} + \text{rlat} + \text{alt}$

```

Model: Spatial GEV model
Deviance: -1384694
TIC: -1383908

Location Parameters:
locCoeff1 locCoeff2 locCoeff3 locCoeff4 locCoeff5 locCoeff6
7.175e-03 4.540e-05 -2.927e-05 -5.425e-09 -6.083e-06 2.152e-06
Scale Parameters:
scaleCoeff1
0.003338
Shape Parameters:
shapeCoeff1
0.155

```

Figure 24:  $\mu \sim \text{rlon} + \text{rlat} + \text{alt} + \text{temperature} + \text{humidity}$

```

Model: Spatial GEV model
Deviance: -1384726
TIC: -1383926

Location Parameters:
locCoeff1 locCoeff2 locCoeff3 locCoeff4 locCoeff5 locCoeff6 locCoeff7 locCoeff8
5.440e-03 4.612e-04 1.290e-05 2.472e-04 -6.604e-05 5.348e-05 -9.301e-05 -9.492e-05
locCoeff9 locCoeff10
-1.014e-04 1.741e-05

```

Figure 25: Model results for  $\mu \sim s(\text{rlon}) + s(\text{rlat}) + s(\text{temperature})$

The first TIC results illustrate well the positive impact of using splines.

| Formula                            | TIC      |
|------------------------------------|----------|
| One neighbor and splines           | -1383909 |
| One neighbor                       | -1383908 |
| Basic stationary model             | -1383911 |
| Basic stationary model without alt | -1383908 |
| Two neighbors                      | -1383901 |
| Two neighbors and splines          | -1383901 |

We start seeing in these first results that more neighbors taken into account for the prediction of  $\mu$  doesn't necessarily leads to better score. Let's see if VGAM/VGLM have the same issues.



**VGLM - VGAM results with VGAM package** Adding covariates at 12 km resolution normally gives us valuable information to predict the location parameter. While our objective is to maximize the information available, taking all the 12 km grid points as covariates in the models is impractical, it would have too much parameters and requires too much data. For that reason we agreed to stop at 4 neighbors and I've got these results.

| Formula                         | Model | TIC      |
|---------------------------------|-------|----------|
| One neighbor and splines        | VGLM  | -1393776 |
| One neighbor                    | VGAM  | -1383908 |
| One neighbor (without altitude) | VGAM  | -1392369 |
| Basic stationary model          | VGLM  | -1393570 |
| Basic stationary model          | VGAM  | -1393570 |
| Two neighbor                    | VGLM  | -1393913 |
| Two neighbor                    | VGAM  | -1393913 |
| Two neighbor splines            | VGLM  | -1392544 |
| Two neighbor splines            | VGLM  | -1392544 |
| Four neighbor                   | VGLM  | -1392823 |
| Four neighbor                   | VGAM  | -1392823 |
| Four neighbor splines           | VGLM  | -1392823 |
| Four neighbor splines           | VGAM  | -1394313 |

Splines definitely leads to better scores and again sometimes less information happens to perform better in terms of TIC as for the 'one neighbor and splines' and 'two neighbors and splines' with VGLM.

**The link functions used are : identity for location, loglink <sup>3</sup> for the scale and logofflink <sup>4</sup>for the shape.**

**Scale determination with VGAM** Adding the determination of the scale parameter leads to a TIC of **-1396196** for the VGAM - four neighbor splines model. Here are an example of the changes of value it can lead to for location and scale :

<sup>3</sup><https://search.r-project.org/CRAN/refmans/VGAM/html/loglink.html>

<sup>4</sup><https://www.rdocumentation.org/packages/VGAM/versions/1.1-6/topics/logofflink>

```

VGLM - Two neighbors with splines
      rlon rlat      loc      scale      shape
0      -2.60 -0.60  0.004318  0.003205  0.404865
1      -2.60 -0.58  0.004379  0.003205  0.404865
2      -2.60 -0.56  0.004371  0.003205  0.404865
3      -2.60 -0.54  0.004362  0.003205  0.404865
4      -2.60 -0.52  0.005254  0.003205  0.404865
...      ...      ...      ...      ...      ...
7795 -0.02  0.50  0.006013  0.003205  0.404865
7796 -0.02  0.52  0.006005  0.003205  0.404865
7797 -0.02  0.54  0.005996  0.003205  0.404865
7798 -0.02  0.56  0.005987  0.003205  0.404865
7799 -0.02  0.58  0.005978  0.003205  0.404865

[7800 rows x 5 columns]
VGAM - Four neighbors with splines - Location and Scale
      rlon rlat      loc      scale      shape
0      -2.60 -0.60  0.003728  0.002705  0.406069
1      -2.60 -0.58  0.004380  0.003086  0.406069
2      -2.60 -0.56  0.004476  0.003136  0.406069
3      -2.60 -0.54  0.004469  0.003134  0.406069
4      -2.60 -0.52  0.004435  0.003054  0.406069
...      ...      ...      ...      ...      ...
7795 -0.02  0.50  0.005383  0.003068  0.406069
7796 -0.02  0.52  0.005313  0.003072  0.406069
7797 -0.02  0.54  0.005242  0.003077  0.406069
7798 -0.02  0.56  0.005243  0.003092  0.406069
7799 -0.02  0.58  0.005171  0.003097  0.406069

```

Figure 26: Location and scale determining

Predicting location and scale leads to better adjustment :

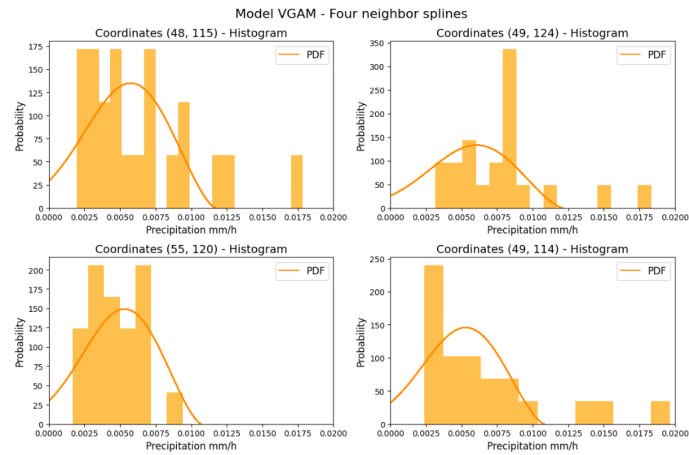


Figure 27: Histogram and predicted PDF for VGAM with four neighbors and splines on trained data (here all the map was used for training)

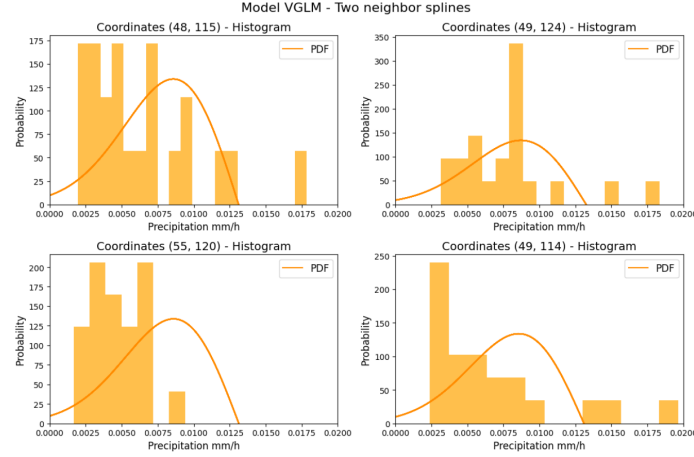


Figure 28: Histogram and predicted PDF for VGLM with two neighbors and splines on trained data (here all the map was used for training)

Despite looking to fit more the data, there is a slight change in the CRPS scores of the two models : 0.0026 for VGAM - 4 neighbors and splines and 0.0027 for VGLM - 2 neighbors and splines

Seeing the results made me think that I should have taken 3-hours accumulated precipitation maxima or daily accumulated precipitation maxima to have maxima that have higher values and maybe I would have seen the tendency more clearly. Also the lack of maxima number appears here.

## 5.2 Hyperparameters choice

**ANNs :** I employed a grid search methodology to find the optimal set of hyperparameters from the previously mentioned options. I printed the model, the hyperparameters and the score for each architecture tried :

```
Model with quantiles q50-q75 : CustomANN(
  (fc_layers): ModuleList(
    (0): Linear(in_features=11, out_features=20, bias=True)
    (1-4): 4 x Linear(in_features=20, out_features=20, bias=True)
    (5): Linear(in_features=20, out_features=2, bias=True)
  )
) with a learning rate of 0.0001, a number of layers : 5, a number of neurons in each layer : 20
End of CRPS validation list 0.0032710159339143573
Model with quantiles q50-q75 : CustomANN(
  (fc_layers): ModuleList(
    (0): Linear(in_features=11, out_features=30, bias=True)
    (1-4): 4 x Linear(in_features=30, out_features=30, bias=True)
    (5): Linear(in_features=30, out_features=2, bias=True)
  )
) with a learning rate of 0.0001, a number of layers : 5, a number of neurons in each layer : 30
End of CRPS validation list 0.002870001833336676
Model with quantiles q50-q75 : CustomANN(
  (fc_layers): ModuleList(
    (0): Linear(in_features=11, out_features=45, bias=True)
    (1-4): 4 x Linear(in_features=45, out_features=45, bias=True)
    (5): Linear(in_features=45, out_features=2, bias=True)
  )
) with a learning rate of 0.0001, a number of layers : 5, a number of neurons in each layer : 45
End of CRPS validation list 0.003127983435159496
```

Figure 29: Part of the hyperparameters search

The architecture that achieved the best CRPS on the validation set is as follows:

- Learning rate: 0.001
- Predicted quantiles: 50 and 75.
- Number of hidden layers: 3
- Number of neurons in each layer: 20

**This architecture results in a CRPS of 0.0022 on the validation set.**

**U-Net :** This U-Net inspired architecture has less hyperparameters to find as it needs to be able to go from a 12 km map to a 2 km map so it has a setup of kernel dimension, padding, stride that ensures to have a higher resolution map of 2km. As a result, the conventional hyperparameters found in standard CNNs, including kernel dimensions, padding strategies, and stride configurations, are non-adjustable with this framework.

I also often displayed the training and validation losses of the models to look for potential overfitting.

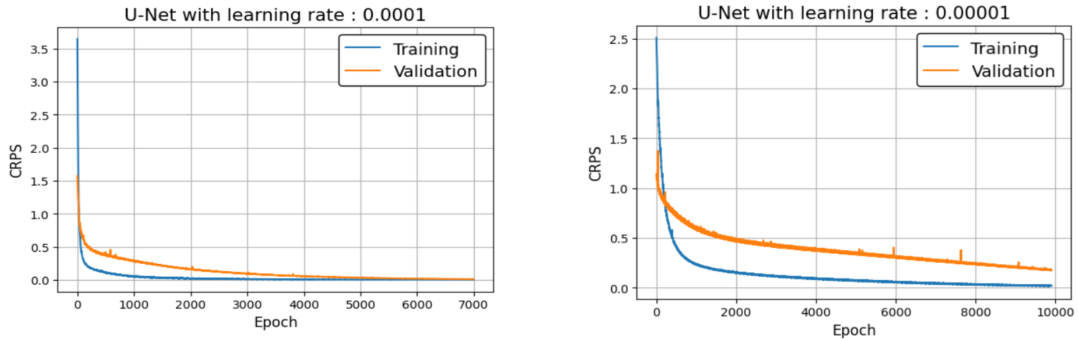


Figure 30: Checking the training - validation losses to detect potential overfitting

I also showed a comparison of the validation curves for different learning rates to see the impact on the convergence.

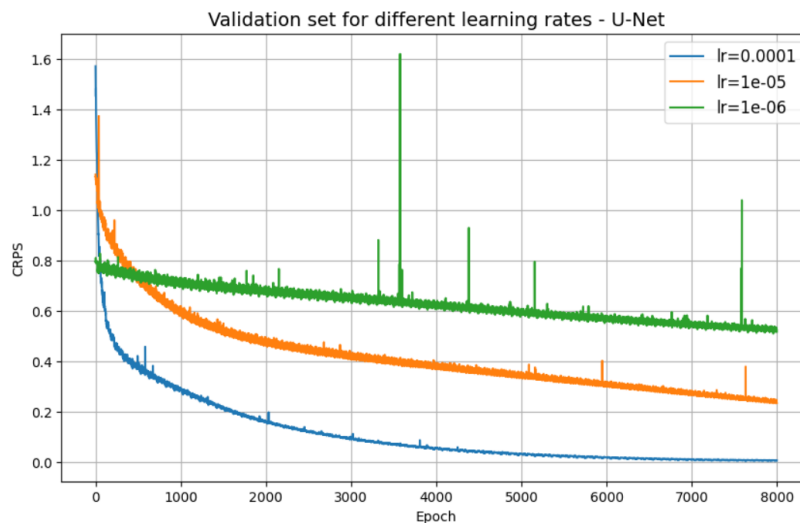


Figure 31: Validation curves

The two hyperparameters tested are the learning rate and the quantiles and here are some runs results.

```

U-Net with quantiles q90-q95 : MyConv(
  (encoder): Sequential(
    (0): Conv2d(2, 20, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(20, 20, kernel_size=(4, 4), stride=(1, 1), padding=(1, 1))
  )
  (decoder): Sequential(
    (0): ConvTranspose2d(20, 20, kernel_size=(4, 4), stride=(2, 2))
    (1): ConvTranspose2d(20, 2, kernel_size=(4, 4), stride=(3, 3), padding=(1, 1))
    (2): ConvTranspose2d(2, 2, kernel_size=(4, 4), stride=(2, 2))
  )
) with a learning rate of 0.001
End of CRPS validation list 0.002452364362141808
U-Net with quantiles q90-q95 : MyConv(
  (encoder): Sequential(
    (0): Conv2d(2, 20, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(20, 20, kernel_size=(4, 4), stride=(1, 1), padding=(1, 1))
  )
  (decoder): Sequential(
    (0): ConvTranspose2d(20, 20, kernel_size=(4, 4), stride=(2, 2))
    (1): ConvTranspose2d(20, 2, kernel_size=(4, 4), stride=(3, 3), padding=(1, 1))
    (2): ConvTranspose2d(2, 2, kernel_size=(4, 4), stride=(2, 2))
  )
) with a learning rate of 0.0001
End of CRPS validation list 0.0026325714831337897
U-Net with quantiles q90-q95 : MyConv(
  (encoder): Sequential(
    (0): Conv2d(2, 20, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(20, 20, kernel_size=(4, 4), stride=(1, 1), padding=(1, 1))
  )
  (decoder): Sequential(
    (0): ConvTranspose2d(20, 20, kernel_size=(4, 4), stride=(2, 2))
    (1): ConvTranspose2d(20, 2, kernel_size=(4, 4), stride=(3, 3), padding=(1, 1))
    (2): ConvTranspose2d(2, 2, kernel_size=(4, 4), stride=(2, 2))
  )
) with a learning rate of 1e-05
End of CRPS validation list 0.10458413770420269

```

Figure 32: Trying different learning rates for quantiles 90 and 95th

**The best setup is the quantiles 25 and 75 and a learning rate of 0.001.**

### 5.3 CRPS results

On the training set we have the following results :

| Model | GLM    | VGAM   | ANNs   | U-Net  |
|-------|--------|--------|--------|--------|
| CRPS  | 0.0028 | 0.0026 | 0.0022 | 0.0022 |

Table 5: CRPS scores for the training set

On the validation set (only for ML architecture) we have the following results :

| Model | ANNs   | U-Net  |
|-------|--------|--------|
| CRPS  | 0.0022 | 0.0021 |

Table 6: CRPS scores for the validation set

On the testing set for temperature and humidity conditions of August, we have the following results :

| Model | VGAM   | ANNs   | U-Net  |
|-------|--------|--------|--------|
| CRPS  | 0.0033 | 0.0034 | 0.0031 |

Table 7: CRPS scores for the testing set

A better CRPS is a smaller one so we could say that ANNs happens to not improve predictions but doesn't seem to change that much as the change in the CRPS is quite small as there is a significant change of 0.001 between the scores of the training and the testing sets. For that reason we need to look at the predicted distributions to visualize the changes.

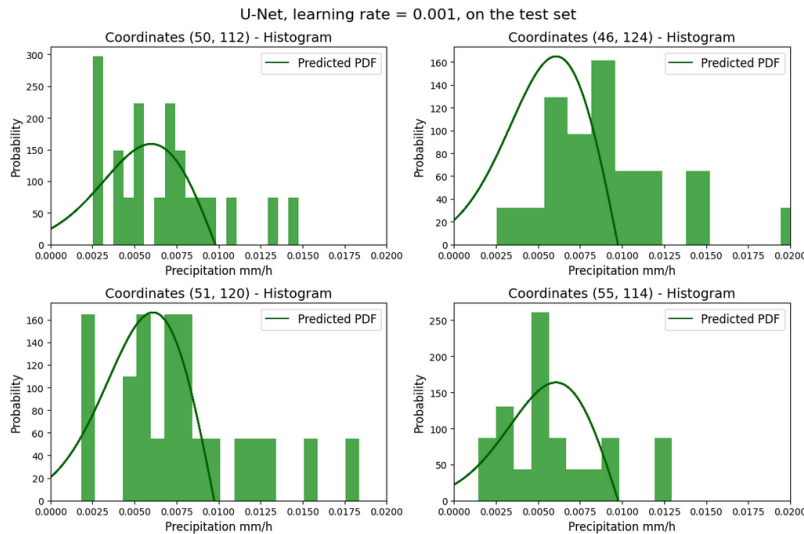


Figure 33: U-Net

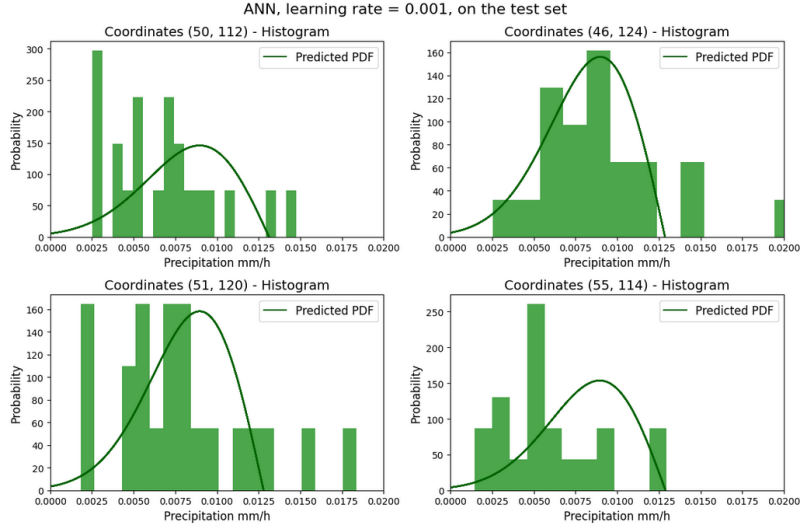


Figure 34: ANN performance on the test set

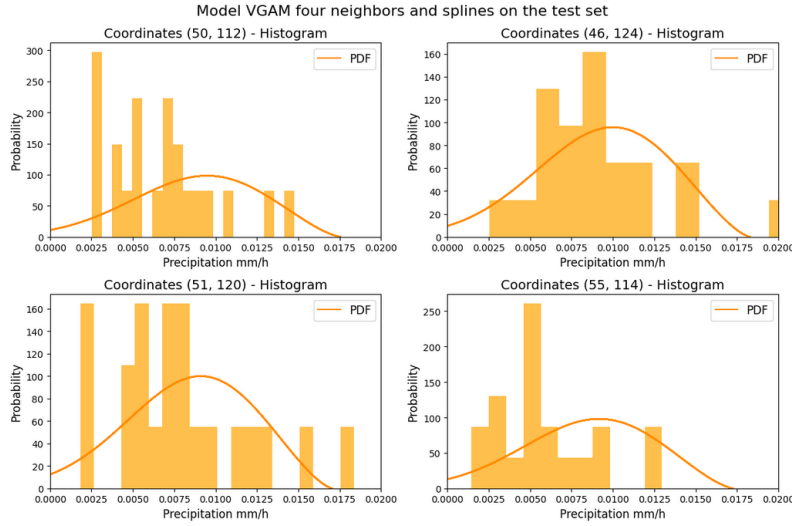


Figure 35: Best statistical model performance on the test set

What we see on these combined histograms and PDFs emphasize the need to consider both the CRPS and the shape of the distribution when evaluating model performance. While U-Net exhibits better performance on the CRPS scale, it generates a more tightly concentrated distribution (as evidenced by its smaller standard deviation). Therefore, in scenarios where our focus lies on predicting unprecedented extremes from the tail of the distribution, statistical models might offer improved results.

**Explaining better results for statistical models :** VGAM is a powerful tool as the functions are polynomials and determined by splines. The obtained results show that in a case of prediction of distribution parameter, the statistical model is complex enough not to be outperform by ANNs : this might not have been the

case if we were working on maxima directly. The CNN inspired by U-Net performs slightly better than VGAM as it can capture spatial dependencies without relying on VGAM or ANN that takes lon, lat as input. This is something that will have to be explored.



## 6 Further work

**Enhancing storm prediction :** Other meteorological variables can be used. Atmospheric instability can be measured with the CAPE (Convective Available Potential Energy) which indicates the potential for upward motion and the development of powerful updrafts.

**Better model training :** We could use patches to train U-Net and then have more training inputs. Patches could have these form : or could intersect be sliding from left to right.

We could also try cross validation with patches/fold where one or some rectangles could be used for validation and the others for the training.

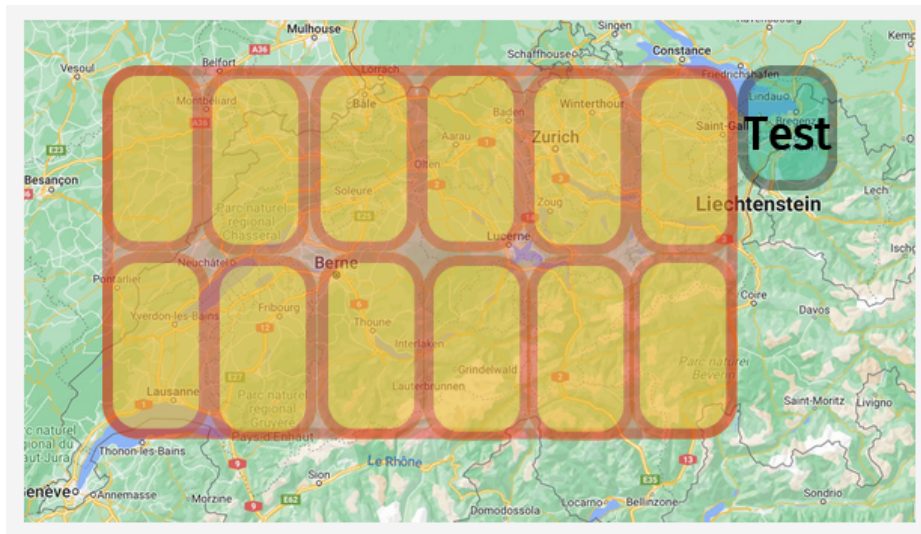


Figure 36: Cross validation set-up

The transformer also needs to be trained.

**Testing robustness to climate change :** What motivates the use of temperature and humidity as predictors and the parameters as output is that this approach might help the model being robust in a future climate by having learned relationship between storm, temperature and humidity.

**Using more the past intern work :** dig into super resolution to compare them better metrics : MSE, RMSE, Perkins score

I hope that this report will be useful to future people working on the subject.

## 7 Conclusion

This internship gave us the opportunity to work with the Continuous Ranked Probability Score, CRPS, within the context of predicting the GEV parameters. Through this process, I encountered issues associated with the use of this scoring method. In conclusion, my findings underscore that by capturing nuanced spatio-temporal patterns, the CNN-U-Net inspired network could help achieving a good prediction. However, when it comes to predicting parameters and not directly the maxima, statistical models can offer great performance. It's crucial then to consider the application when looking at the prediction distribution and give enough consideration to the distribution tail for a comprehensive exploration of extreme scenarios. By doing this internship and this report we build a methodology that holds potential for future researchers to work on that topic.

On a personal matter, I gradually get familiar with neural networks, hyperparameters search, and training-validation-testing set, allowing me to have a better practice and understanding of machine learning basics. I learned how demanding machine learning is and the internship has given me a solid foundation to build upon as I continue my journey in ML related to climate. I also had the opportunity to work extensively with Python and R. These skills are super practical and I can already see how useful they'll be in the future.

One of the most inspiring aspects of this internship was witnessing the potential impact of climate-related research with interdisciplinary work. The prospect of contributing to the establishment of climate-focused centers and engaging in cross-disciplinary collaborations made me really enthusiast for the future of this field and I look forward to seeing the next ECCE's projects (hoping to be part of).

In conclusion, these three months have been an awesome research experience and I'm very thankful to ECCE for giving me this opportunity.

## References

- [1] Sándor Baran and Ágnes Baran. Calibration of wind speed ensemble forecasts for power generation. *Quarterly Journal of the Hungarian Meteorological Service*, 125(4):609–624, October–December 2021.
- [2] Elizabeth A. Barnes, Randal J. Barnes, and Nicolas Gordillo. Adding uncertainty to neural network regression tasks in the geosciences. *Atmospheric and Oceanic Physics*, 2021. Submitted on 15 Sep 2021.
- [3] Jorge Baño-Medina, Rodrigo Manzananas, and José Manuel Gutiérrez. Configuration and intercomparison of deep learning neural models for statistical downscaling. *Geoscientific Model Development*, 2020.
- [4] R. E. Benestad. Empirical-statistical downscaling in climate modeling. *Eos Trans. AGU*, 85(42):417–422, 2004.
- [5] Siddharth Bhatia, Arjit Jain, and Bryan Hooi. Exgan: Adversarial generation of extreme samples. *Association for the Advancement of Artificial Intelligence*, 2021.
- [6] Stuart Coles. *An Introduction to Statistical Modeling of Extreme Values*. Springer Science & Business Media, August 2001.
- [7] H. J. Fowler, S. Blenkinsop, and C. Tebaldi. Linking climate change modelling to impacts studies: Recent advances in downscaling techniques for hydrological modelling. *International Journal of Climatology*, 27:1547–1578, 2007. Published online 14 September 2007 in Wiley InterScience.
- [8] Manuel Gebetsberger, Jakob W. Messner, Georg J. Mayr, and Achim Zeileis. Estimation methods for nonhomogeneous regression models: Minimum continuous ranked probability score versus maximum likelihood. *Monthly Weather Review*, 146(12):4323–4338, 2018.
- [9] Firas Gerges, Michel C. Boufadel, Elie Bou-Zeid, Hani Nassif, and Jason T. L. Wang. A novel deep learning approach to the statistical downscaling of temperatures for monitoring climate change. *Association for Computing Machinery*, 2022.
- [10] Katherine Haynes, Ryan Lagerquist, Marie McGraw, Kate Musgrave, and Imme Ebert-Uphoff. Creating and evaluating uncertainty estimates with neural networks for environmental-science applications. *Artificial Intelligence for the Earth Systems (AIES)*, 2023.
- [11] Laureline Hentgen, Nikolina Ban, Nico Kröner, David Leutwyler, and Christoph Schär. Clouds in convection-resolving climate simulations over europe. *Journal of Geophysical Research: Atmospheres*, 124(7):3849–3870, 2019. Research Article, Free Access.
- [12] Alexander Jordan, Fabian Krüger, and Sebastian Lerch. Evaluating probabilistic forecasts with scoringrules. *Journal of Statistical Software*, 2019.

- [13] Jie Xiang\*, Lifeng Zhang, Fuhan Zhang, Li Xiang†, Jiping Guan†. Spatiotemporal model based on transformer for bias correction and temporal downscaling of forecasts. *Frontiers in Environmental Science*, 10, November 2022.
- [14] Yingkai Sha, David John Gagne II, Gregory West, and Roland Stull. Deep-learning-based gridded downscaling of surface meteorological variables in complex terrain. part ii: Daily precipitation. *American Meteorological Society*, August 2020. Department of Earth, Ocean and Atmospheric Sciences, The University of British Columbia, Vancouver, British Columbia, Canada; National Center for Atmospheric Research, Boulder, Colorado; BC Hydro, Burnaby, British Columbia Canada.
- [15] R. Shibata. Statistical aspects of model selection. In J.C. Willems, editor, *From Data to Model*. Springer, Berlin, Heidelberg, 1989.
- [16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, A. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

## 8 Appendix

### Internship Schedule

| Week         | Activities                                                                                                                                                                                                |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Week 1 -2    | Reading papers, being able to use the cluster                                                                                                                                                             |
| Week 3       | Reproducing what the past intern did. Manipulating the data using Python. Reading about GANs and super resolution models.                                                                                 |
| Week 4 - 5   | Reading about GAN/Transformer/Diffusion models and trying implementation of the CRPS for the parameter prediction approach.                                                                               |
| Week 6       | Writing my first NN (linear) and trying to train it : problems of the shape parameter value.                                                                                                              |
| Week 7       | Struggles with the CRPS and first statistical models : trying GLM and VGAM with R.                                                                                                                        |
| Week 8 - 9   | Adding complexity to the Neural Networks : activation functions, more layers. Implementation of the U-Net and of the Transformer. Changes in dataset : from small part of Switzerland to all the country. |
| Week 10 - 11 | Managing the use of both Python and R. Training the Neural Networks, start working on final results                                                                                                       |
| Week 12 - 13 | Report writing, results showing : PDF and histograms. Discussing the futur of the project : writing an article.                                                                                           |

### 8.1 CRPS Formula proof

#### 8.1.1 First lemma

Let  $X$  and  $Y$  be independent real random variables with finite expectations.  $F$  is the distribution function of  $X$ , and  $G$  is the distribution function of  $Y$ . We use

$$\begin{aligned}
 |X - Y| &= \int_{-\infty}^{+\infty} (\mathbb{1}_{(X \leq u < Y)} + \mathbb{1}_{(Y \leq u < X)}) du \\
 \mathbb{E}[|X - Y|] &= \int_{x=-\infty}^{-\infty} \int_{y=-\infty}^{+\infty} |x - y| dF(x) dG(y) \\
 \mathbb{E}[|X - Y|] &= \int_{x=-\infty}^{+\infty} \int_{y=-\infty}^{+\infty} \int_{u=-\infty}^{+\infty} (\mathbb{1}_{(X \leq u < Y)} + \mathbb{1}_{(Y \leq u < X)}) du dF(x) dG(y)
 \end{aligned}$$

Fubini's theorem enables us to write

$$\int_{x=-\infty}^{+\infty} \int_{y=-\infty}^{+\infty} \int_{u=-\infty}^{+\infty} (\mathbb{1}_{(X \leq u < Y)}) du dF(x) dG(y) = \int_{x=-\infty}^{+\infty} \int_{u=-\infty}^{+\infty} \mathbb{1}_{x \leq u} dF(x) \int_{y=v}^{+\infty} dG(y)$$

$$= \int_{u=-\infty}^{+\infty} \left( \int_{x=-\infty}^u dF(x) \right) (1 - G(u)) = \int_{u=-\infty}^{+\infty} F(u)(1 - G(u))du$$

Likewise :

$$\int_{x=-\infty}^{+\infty} \int_{y=-\infty}^{+\infty} \int_{u=-\infty}^{+\infty} (\mathbb{1}_{(Y \leq u < X)}) du dF(x) dG(y) = \int_{u=-\infty}^{+\infty} G(u)(1 - F(u))du$$

Since the two parts of the integrand are integrable on their respective sets of definitions, the integral separates and we obtain :

$$\mathbb{E}[|X - Y|] = \int_{u=-\infty}^{+\infty} F(u)(1 - G(u))du + \int_{u=-\infty}^{+\infty} G(u)(1 - F(u))du$$

For X and Y having the same distribution function :

$$\mathbb{E}[|X - Y|] = 2 \int_{u=-\infty}^{+\infty} F(u)(1 - F(u))$$

### 8.1.2 From the original definition of the CRPS

$$\text{CRPS}(F(z), y) = \int_{-\infty}^{\infty} (F(z) - \mathbb{1}_{(z \geq y)})^2 dx = \int_{-\infty}^{\infty} (F(z)^2 - 2F(z)\mathbb{1}_{(z \geq y)} + \mathbb{1}_{(z \geq y)})dz$$

Let's show that

$$\text{CRPS}(F, y) = \mathbb{E}_F |X - y| - \frac{1}{2} \mathbb{E}_F |X - X'|$$

with X and X' which has the same distribution function F

$$\begin{aligned} \mathbb{E}_F |X - y| &= \int_{x=-\infty}^{\infty} \left( \int_{u=-\infty}^{+\infty} (\mathbb{1}_{(x \leq u < y)} + \mathbb{1}_{(y \leq u < x)}) du \right) dF(x) \\ &= \int_{u=-\infty}^{\infty} \left( \int_{x=-\infty}^{+\infty} \mathbb{1}_{(x \leq u < y)} dF(x) \right) du + \int_{u=-\infty}^{\infty} \left( \int_{x=-\infty}^{+\infty} \mathbb{1}_{(y \leq u < x)} dF(x) \right) du \\ &= \int_{u=-\infty}^{\infty} \left( \int_{x=-\infty}^u \mathbb{1}_{(u < y)} dF(x) \right) du + \int_{u=-\infty}^{\infty} \left( \int_{x=u}^{+\infty} \mathbb{1}_{(y \leq u)} dF(x) \right) du \\ &= \int_{u=-\infty}^{\infty} F(u) \mathbb{1}_{(u < y)} du + \int_{u=-\infty}^{\infty} \mathbb{1}_{(u > y)} - F(u) \mathbb{1}_{(u > y)} du \\ &= \int_{-\infty}^{\infty} (F(u) - 2F(u) \mathbb{1}_{u \geq y} + \mathbb{1}_{u \geq y}) du \end{aligned}$$

Hence :

$$\begin{aligned}
\mathbb{E}_F |X - y| - \frac{1}{2} \mathbb{E}_F |X - X'| &= 2 \int_{u=-\infty}^{+\infty} F(u)(1-F(u))du + \int_{-\infty}^{\infty} (F(u) - 2F(u)\mathbb{1}_{u \geq y} + \mathbb{1}_{u \geq y})du \\
&= \int_{-\infty}^{\infty} (Fu^2 - 2F(u)\mathbb{1}_{(u \geq y)} + \mathbb{1}_{(u \geq y)})du \\
&= CRPS(F(u), y)
\end{aligned}$$

## 8.2 Hyperparameters search : U-Net

```

Model with quantiles q50-q75 : CustomANN(
  (fc_layers): ModuleList(
    (0): Linear(in_features=11, out_features=20, bias=True)
    (1-4): 4 x Linear(in_features=20, out_features=20, bias=True)
    (5): Linear(in_features=20, out_features=2, bias=True)
  )
) with a learning rate of 0.0001 , a number of layers : 5 , a number of neurons in each layer : 20
End of CRPS validation list 0.0032710159339143573
Model with quantiles q50-q75 : CustomANN(
  (fc_layers): ModuleList(
    (0): Linear(in_features=11, out_features=30, bias=True)
    (1-4): 4 x Linear(in_features=30, out_features=30, bias=True)
    (5): Linear(in_features=30, out_features=2, bias=True)
  )
) with a learning rate of 0.0001 , a number of layers : 5 , a number of neurons in each layer : 30
End of CRPS validation list 0.002870001833336676
Model with quantiles q50-q75 : CustomANN(
  (fc_layers): ModuleList(
    (0): Linear(in_features=11, out_features=45, bias=True)
    (1-4): 4 x Linear(in_features=45, out_features=45, bias=True)
    (5): Linear(in_features=45, out_features=2, bias=True)
  )
) with a learning rate of 0.0001 , a number of layers : 5 , a number of neurons in each layer : 45
End of CRPS validation list 0.003127983435159496

```

Figure 37: Testing the learning rate for quantiles 50 and 75