DeepExtrema: A Deep Learning Approach for Forecasting Block Maxima in Time Series Data

Anonymous submission

Abstract

Accurate forecasting of extreme values in time series is critical due to the significant impact of extreme events on human and natural systems. This paper presents DeepExtrema, a novel framework that combines a deep neural network (DNN) with generalized extreme value (GEV) distribution to forecast the block maximum value of a time series. Implementing such a network is a challenge as the framework must preserve the inter-dependent constraints among the GEV model parameters even when the DNN is initialized. We describe our approach to address this challenge and present an architecture that enables both conditional mean and quantile prediction of the block maxima. The extensive experiments performed on both real-world and synthetic data demonstrated the superiority of DeepExtrema compared to other baseline methods.

1 Introduction

Extreme events such as droughts, floods, and severe storms occur when the values of the corresponding geophysical variables (such as temperature, precipitation, or wind speed) reach their highest or lowest point during a period or surpass a threshold value. Extreme events have far-reaching consequences for both humans and the environment. For example, four of the most expensive hurricane disasters in the United States since 2005—Katrina, Sandy, Harvey, and Irma—have each incurred over $50 billion in damages with enormous death tolls [US GAO, 2020]. Accurate forecasting of the extreme events is therefore crucial as it not only helps provide timely warnings to the public but also enables emergency managers and responders to better assess the risk of potential hazards caused by future extreme events.

Despite its importance, forecasting time series with extremes can be tricky as the extreme values may have been ignored as outliers during training to improve the generalization performance of the model. Furthermore, as current approaches are mostly designed to minimize the mean-square prediction error, their fitted models focus on predicting the conditional expectation of the target variable rather than its extreme values [Bishop, 2006]. Extreme value theory (EVT) offers a statistically well-grounded approach to derive the

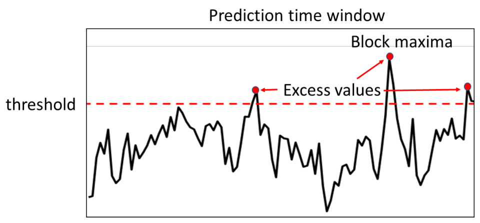

Figure 1: Types of extreme values in a given time window.

limiting distribution governing a sequence of extreme values [Coles *et al.*, 2001]. The two most popular distributions studied in EVT are the generalized extreme value (GEV) and generalized Pareto (GP) distributions. Given a prediction time window, GEV governs the distribution of its block maxima, whereas GP is concerned with the distribution of excess values over a certain threshold, as shown Figure 1. This paper focuses on forecasting the block maxima as it allow us to assess the worst-case scenario in the given time window and avoids making ad-hoc decisions related to the choice of excess threshold to use for the GP distribution. Unfortunately, classical EVT has limited capacity in terms of modeling highly complex, nonlinear relationships present in time series data. For example, [Kharin and Zwiers, 2005] uses a simple linear model with predictors to infer the parameters of a GEV distribution.

Deep learning methods have grown in popularity in recent years due to their ability to capture nonlinear dependencies in the data. Previous studies have utilized a variety of deep neural network architectures for time series modeling, including long short-term memory networks [Sagheer and Kotb, 2019; Masum *et al.*, 2018; Laptev *et al.*, 2017], convolutional neural networks [Bai *et al.*, 2018; Zhao *et al.*, 2017; Yang *et al.*, 2015], encoder-decoder based RNN [Peng *et al.*, 2018], and attention-based models [Zhang *et al.*, 2019; Aliabadi *et al.*, 2020]. However, these works are mostly focused on predicting the conditional mean of the target variable. While there have some recent attempts to incorporate EVT into deep learning [Wilson *et al.*, 2022; Ding *et al.*, 2019; Polson and Sokolov, 2020], they are primarily focused on modeling the tail distribution, i.e., excess values over a threshold, using the GP distribution, rather than forecasting the block maxima using the GEV distribution. Furthermore, instead of inferring the distribution parameters from data, some methods [Ding *et al.*, 2019] assume that the parameters are fixed at all times and can be provided as user-specified hyperparameters while others [Polson and Sokolov, 2020] do not enforce the necessary constraints on parameters of the extreme value distribution.

Incorporating the GEV distribution into the deep learning formulation presents many technical challenges. First, the GEV parameters must satisfy certain positivity constraints to ensure that the predicted distribution has a finite bound [Coles *et al.*, 2001]. Maintaining these constraints throughout the training process is a challenge since the model parameters depend on the observed predictor values in a mini-batch. Another challenge is the scarcity of data since there is only one block maximum value per time window. This makes it hard to accurately infer the GEV parameters for each window from a set of predictors. Finally, the training process is highly sensitive to model initialization. An improper initialization may lead to violations of the positivity constraints as the initial DNN outputs may produce an unbounded log likelihood function or an ill-defined expected value of block maxima when the GEV shape parameter, ξ, is beyond certain range of values [Coles *et al.*, 2001], i.e., $\xi < -1$ or $\xi > 1$.

To overcome these challenges, we propose a novel framework called `DeepExtrema` that utilizes the GEV distribution to characterize the distribution of block maximum values for a given forecast time window. The parameters of the GEV distribution are estimated using a deep neural network (DNN), which is trained to capture the nonlinear dependencies in the time series data. This is a major departure from previous work by [Ding *et al.*, 2019], where the distribution parameters are assumed to be a fixed, user-specified hyperparameter. `DeepExtrema` reparameterizes the GEV formulation to ensure that the DNN output is compliant with the GEV positivity constraints. In addition, `DeepExtrema` offers a novel, model bias offset mechanism in order to ensure that the GEV constraints are preserved at all times, even when the weights of the DNN are randomly initialized.

In summary, the main contributions of the paper are:

1. We present a novel framework to predict the block maxima of a given time window by incorporating GEV distribution into the training of a DNN.
2. We propose a reformulation of the GEV constraints to ensure they can be enforced using activation functions in the DNN.
3. We introduce a model bias offset mechanism to ensure that the DNN output preserves the GEV constraints in spite of its random initialization.
4. We perform extensive experiments on both real-world and synthetic data to demonstrate the effectiveness of `DeepExtrema` compared to other baseline methods.

2 Related Work

Deep learning architectures are widely used in time series forecasting due to their immense success in learning complex, nonlinear relationships in data. For example, long short-term memory (LSTM) networks [Hochreiter and Schmidhuber, 1997] have been successfully applied to various application domains such as electric load prediction [Masum *et al.*, 2018]. [Peng *et al.*, 2018] employed an encoder-decoder-based GRU, which is a variation of the LSTM architecture, to predict host workloads in a cloud computing environment while [Sagheer and Kotb, 2019] proposed a variation of LSTM to predict petroleum production. [Zerveas *et al.*, 2021] proposed a Transformer-based architecture for time series forecasting.

Convolutional neural networks have also been successfully applied to time series modeling problems. For example, [Yang *et al.*, 2015] proposed a CNN architecture for activity recognition while [Zhao *et al.*, 2017] employed a CNN-based architecture for a variety of time series classification problems.

Kharin et al [Kharin and Zwiers, 2005] is representative of much of the traditional statistical work utilizing EVT. They analyzed GEV parameters assuming there was a simple relationship between those parameters and a single predictor, i.e., time. As previously mentioned, prior work combining deep learning with EVT suffers from significant limitations. For instance, [Ding *et al.*, 2019] incorporated the GP distribution in their loss function to forecast excesses over a threshold. However, instead of predicting the GP parameters from covariates, they assume the GP parameters values are known *a-priori* and can be provided as hyperparameters of their algorithm. Thus, the GP parameters are assumed to be fixed (constant) for all-time series, which is a strong assumption especially if the time series is generated for different locations. [Polson and Sokolov, 2020] proposed to combine deep learning model with extreme value theory to model tail behavior of a time series. They applied the GP distribution for modeling excess values and used its negative log-likelihood as the loss function. However, a major issue with their proposed framework is that it does not incorporate a mechanism to enforce constraints on parameters of the learned GP distribution, which are essential to ensure the predicted distribution is well-behaved.

For uncertainty quantification in the time series problem, [Wang *et al.*, 2020] proposed a distribution-free novel approach called DeepPIPE. It simultaneously predicted point estimations as well as quantile estimations without any prior assumption about the data distribution. They proposed a hybrid loss function based on point estimations and point intervals to leverage point and quantile estimations. Though it might be a sound baseline for quantile estimation, it did not incorporate EVT theory. [Laptev *et al.*, 2017] proposed a LSTM-based architecture for time series extreme event forecasting where they combined Bootstrap and Bayesian approaches for uncertainty estimation.

3 Preliminaries

3.1 Problem Statement

Let $z_1, z_2, \cdots, z_T$ be a time series of length T. Assume the time series is partitioned into a set of time windows, where each window $[t-\alpha, t+\beta]$ contains a sequence of predictors, $x_t = (z_{t-\alpha}, z_{t-\alpha+1}, \cdots, z_t)$, and target, $\tilde{y}_t = (z_{t+1}, z_{t+2}, \cdots, z_{t+\beta})$. Note that β is known as the forecast horizon of the prediction. For each time window, let $y_t = \max_{\tau \in \{1,\cdots,\beta\}} z_{t+\tau}$ be the block maxima of the target variable at time t. Our time series forecasting task is to estimate the block maxima, $\hat{y}_t$, as well as its upper and lower quantile estimates, $\hat{y}_U$ and $\hat{y}_L$, of a future time window based on current and past data, x_t.

3.2 Generalized Extreme Value Distribution

The GEV distribution governs the distribution of block maxima in a given window. Let $Y = \max\{z_1, z_2, \cdots, z_t\}$. If there exist sequences of constants $a_t > 0$ and b_t such that

$$Pr(Y - b_t)/a_t \leq y \rightarrow G(y) \quad \text{as } t \rightarrow \infty$$

for a non-degenerate distribution G, then the cumulative distribution function G belongs to a family of GEV distribution of the form [Coles *et al.*, 2001]:

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

The GEV distribution is characterized by the following parameters: μ (location), σ (scale), and ξ (shape). The expected value of the distribution is given by

$$y_{mean} = \mu + \frac{\sigma}{\xi}\left[\Gamma(1-\xi) - 1\right] \tag{2}$$

where $\Gamma(x)$ denotes the gamma function of a variable $x > 0$. This means y_{mean} is only well-defined for $\xi < 1$. Furthermore, the *pth* quantile of the GEV distribution, y_p, can be also calculated as follows:

$$y_p = \mu + \frac{\sigma}{\xi}\left[(-\log p)^{-\xi} - 1\right] \tag{3}$$

Given n independent block maxima values, $\{y_1, y_2, \cdots, y_n\}$, with the distribution function given by Equation (1) and assuming $\xi \neq 0$, its log-likelihood function is given by:

$$\log L_{GEV}(\mu,\sigma,\xi) = -n\log\sigma - (\frac{1}{\xi}+1)\sum_{i=1}^{n}\log(1+\xi\,\frac{y_i-\mu}{\sigma}) - \sum_{i=1}^{n}(1+\xi\,\frac{y_i-\mu}{\sigma})^{-1/\xi} \tag{4}$$

The GEV parameters (μ, σ, ξ) can be estimated using the maximum likelihood (ML) approach by maximizing (4) subject to the following positivity constraints:

$$\sigma > 0 \quad \text{and} \quad \forall\, i: 1 + \frac{\xi}{\sigma}(y_i - \mu) > 0 \tag{5}$$

In addition to the above positivity constraints, the shape parameter ξ must be within certain range of values in order for the ML estimators to exist and have regular asymptotic properties [Coles *et al.*, 2001]. Specifically, the ML estimators have regular asymptotic properties as long as $\xi > -0.5$. Otherwise, if $-1 < \xi < -0.5$, then the ML estimators may exist but will not have regular asymptotic properties. Finally, the ML estimators do not exist if $\xi < -1$ [Smith, 1985].

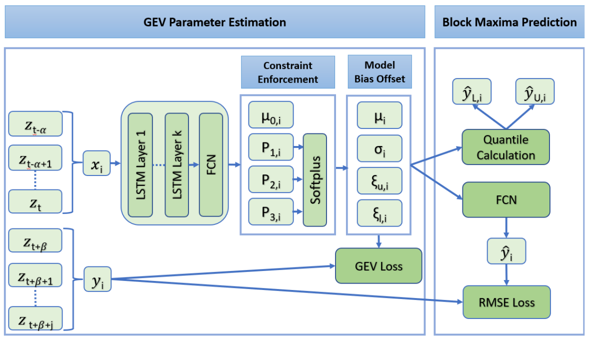

Figure 2: Proposed DeepExtrema framework for predicting block maxima using GEV distribution.

4 Proposed Framework: DeepExtrema

This section presents the proposed DeepExtrema framework for predicting the block maxima of a given time window. The predicted block maxima $\hat{y}$ follows a GEV distribution, whose parameters are conditioned on observations of the predictors x. Figure 2 provides an overview of the DeepExtrema architecture. Given the input predictors x, the framework uses a stacked LSTM network to learn a representation of the time series. The LSTM will output a latent representation, which will used by a fully connected layer to generate the GEV parameters:

$$(\mu, \sigma, \xi_u, \xi_l) = LSTM(x) \tag{6}$$

where μ, σ, and ξ's are the location, shape, and scale parameters of the GEV distribution. Note that ξ_u and ξ_l are the estimated parameters due to reformulation of the GEV constraints, which will be described in the next subsection.

The proposed Model Bias Offset (MBO) component performs bias correction on the estimated GEV parameters to ensure that the LSTM outputs preserve the GEV constraints in (5) and generate a feasible value for ξ irrespective of how the network was initialized. The GEV parameters are subsequently provided to a fully connected layer to obtain point estimates of the block maxima, which include its expected value $\hat{y}$ as well as upper and lower quantiles, $\hat{y}_U$ and $\hat{y}_L$, using the equations given in (2) and (3), respectively. The GEV parameters are then used to compute the negative log-likelihood of the estimated GEV distribution, which will be combined with the root-mean-square error (RMSE) of the predicted block maxima to determine the overall loss function. Details of the different components are described in the subsections below.

4.1 GEV Parameter Estimation

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ be a set of training examples, where each x_i denotes the predictor time series and y_i is the corresponding block maxima for time window i. A naïve approach is to assume that the GEV parameters (μ, σ, ξ) are constants for all time windows. This can be done by fitting a global GEV distribution to the set of block maxima values y_i's using the maximum likelihood approach given in (4). Instead of using a global GEV distribution with fixed parameters, our goal is to learn the parameters (μ_i, σ_i, ξ_i) of each window i using the predictors x_i. The added flexibility enables the model to improve the accuracy of its block maxima prediction, especially for non-stationary time series.

The estimated GEV parameters generated by the LSTM must satisfy the two positivity constraints given by the inequalities in (5). While the first positivity constraint on σ_i is straightforward to enforce, maintaining the second one is harder as it involves a nonlinear relationship between y_i and the estimated GEV parameters, ξ_i, μ_i, and σ_i. The GEV parameters may vary from one input x_i to another, and thus, learning them from the limited training examples is a challenge. Worse still, some of the estimated GEV parameters could be erroneous, especially at the initial rounds of the training epochs, making it difficult to satisfy the constraints throughout the learning process.

To address these challenges, we propose a reformulation of the second constraint in (5). This allows the training process to proceed even though the second constraint in (5) has yet to be satisfied especially in the early rounds of the training epochs. Specifically, we relax the hard constraint by adding a small tolerance factor, $\tau > 0$, as follows:

$$\forall i : 1 + \frac{\xi}{\sigma}(y_i - \mu) + \tau \geq 0. \tag{7}$$

The preceding soft constraint allows for minor violations of the second constraint in (5) as long as $1 + \frac{\xi}{\sigma}(y_i - \mu) > -\tau$ for all time windows i. Furthermore, to ensure that the inequality holds for all y_i's, we reformulate the constraint in (7) in terms of $y_{\min} = \min_i y_i$ and $y_{\max} = \max_i y_i$ as follows:

Theorem 1. *Assuming $\xi \neq 0$, the soft constraint in* (7) *can be reformulated into the following bounds on ξ:*

$$-\frac{\sigma}{y_{\max} - \mu}(1 + \tau) \leq \xi \leq \frac{\sigma}{\mu - y_{\min}}(1 + \tau) \tag{8}$$

where τ is the tolerance on the constraint in 5.

Proof. For the lower bound on ξ, set y_i to be $y_{\max}$ in (7):

$$1 + \frac{\xi}{\sigma}(y_{\max} - \mu) + \tau \geq 0 \implies \frac{\xi}{\sigma}(y_{\max} - \mu) \geq -(1 + \tau)$$
$$\implies \xi \geq -\frac{\sigma}{(y_{\max} - \mu)}(1 + \tau)$$

To obtain the upper bound on ξ, set y_i to be $y_{\min}$ in (7):

$$1 + \frac{\xi}{\sigma}(y_{\min} - \mu) + \tau \geq 0 \implies \frac{\xi}{\sigma}(\mu - y_{\min}) \leq (1 + \tau)$$
$$\implies \xi \leq \frac{\sigma}{(\mu - y_{\min})}(1 + \tau)$$

□

Following Theorem 1, the upper and lower bound constraints on ξ in (8) can be restated as follows:

$$\frac{\sigma}{\mu - y_{\min}}(1 + \tau) - \xi \geq 0$$
$$\xi + \frac{\sigma}{y_{\max} - \mu}(1 + \tau) \geq 0 \tag{9}$$

The reformulation imposes lower and upper bounds on ξ, which can be used to re-parameterize the second constraint in (5). Next, we describe how the reformulated constraints in (9) can be enforced by `DeepExtrema` in a DNN.

Given an input x_i, `DeepExtrema` will generate the following four outputs: μ_i, P_{1i}, P_{2i}, and P_{3i}. A softplus activation function, $softplus(x) = \log(1 + \exp(x))$, which is a smooth approximation to the ReLU function, is used to enforce the non-negativity constraints associated with the GEV parameters. The scale parameter σ_i can be computed using the softplus activation function on P_{1i} as follows:

$$\sigma_i = softplus(P_{1i}) \tag{10}$$

This ensures the constraint $\sigma_i \geq 0$ is met. The lower and upper bound constraints on ξ_i given by the inequalities in (9) are enforced using the softplus function on P_{2i} and P_{3i}:

$$\frac{\sigma_i}{\mu_i - y_{\min}}(1 + \tau) - \xi_{u,i} = softplus(P_{2i})$$
$$\frac{\sigma_i}{y_{\max} - \mu_i}(1 + \tau) + \xi_{l,i} = softplus(P_{3i}) \tag{11}$$

By re-arranging the above equation, we obtain

$$\xi_{u,i} = \frac{\sigma_i}{\mu_i - y_{\min}}(1 + \tau) - softplus(P_{2i})$$
$$\xi_{l,i} = softplus(P_{3i}) - \frac{\sigma_i}{y_{\max} - \mu_i}(1 + \tau) \tag{12}$$

`DeepExtrema` computes the upper and lower bounds on ξ_i using the formulas in (12). During training, it will minimize the distance between $\xi_{u,i}$ and $\xi_{l,i}$ and will use the value of $\xi_{u,i}$ as the estimate for ξ_i. Note that the two ξ_i's converge rapidly to a single value after a small number of training epochs.

4.2 Model Bias Offset (MBO)

Although the constraint reformulation approach described in the previous subsection ensures that the DNN outputs will satisfy the GEV constraints, the random initialization of the network can produce estimates of ξ that violate the regularity conditions described in Section 3.2. Specifically, the ML-estimated distribution may not have the asymptotic GEV distribution when $\xi < -0.5$ while its conditional mean is not well-defined when $\xi > 1$. Additionally, the estimated location parameter μ may not fall within the desired range between $y_{\min}$ and $y_{\max}$ when the DNN is randomly initialized. Thus, without proper initialization, the DNN will struggle to converge to a good solution and produce acceptable values of the GEV parameters.

One way to address this challenge is to repeat the random initialization of the DNN until a reasonable set of initial GEV parameters, i.e., $y_{\min} \leq \mu \leq y_{\max}$ and $-0.5 < \xi < 1$, is found. However, this approach is infeasible given the size of the parameter space of the DNN. A better strategy is to control the initial output of the neural network in order to produce an acceptable set of GEV parameters, (μ, σ, ξ) during initialization. Unfortunately, controlling the initial output of a neural network is difficult given its complex architecture.

We introduce a simple but effective technique called Model Bias Offset (MBO) to address this challenge. The key insight here is to view the GEV parameters as a biased output due to the random initialization of the DNN and then perform bias correction to alleviate the effect of the initialization. To do

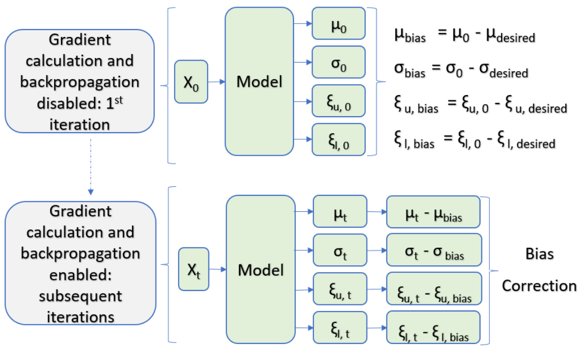
Gradient calculation and backpropagation disabled: 1st iteration
X0
Model
μ0
σ0
ξu, 0
ξl, 0
μbias = μ0 - μdesired
σbias = σ0 - σdesired
ξu, bias = ξu, 0 - ξu, desired
ξl, bias = ξl, 0 - ξl, desired
Gradient calculation and backpropagation enabled: subsequent iterations
Xt
Model
μt
σt
ξu, t
ξl, t
μt - μbias
σt - σbias
ξu, t - ξu, bias
ξl, t - ξl, bias
Bias Correction

Figure 3: Model Bias Offset (MBO) to ensure the initial estimates of the GEV parameters are reasonable even when the DNN is randomly initialized.

this, let $\mu_{\text{desired}}, \sigma_{\text{desired}}$, and ξ_{desired} be an acceptable set of initial GEV parameters. The values of these initial parameters must satisfy the constraints $-0.5 < \xi_{\text{desired}} < 1$, $\sigma_{\text{desired}} > 0$, and $y_{\min} \le \mu_{\text{desired}} \le y_{\max}$. This can be done by randomly choosing a value from the preceding range of acceptable values, or more intelligently, using the GEV parameters estimated from a global GEV distribution fitted to the block maxima y_i's in the training data via the ML approach given in (4), without considering the input predictors (see the discussion at the beginning of Section 4.1). We find that the latter strategy works well in practice as it can lead to faster convergence especially when the global GEV parameters are close to the final values after training.

When the DNN is randomly initialized, let $\mu_0, \sigma_0, \xi_{u,0}$, and $\xi_{l,0}$ be the initial DNN output for the GEV parameters. These initial outputs may not necessarily fall within their respective range of acceptable values. We consider the difference between the initial DNN output and the desired GEV parameters as a *model bias* due to the random initialization:

$$\begin{aligned} \mu_{\text{bias}} &= \mu_0 - \mu_{\text{desired}} & \sigma_{\text{bias}} &= \sigma_0 - \sigma_{\text{desired}} \\ \xi_{u,\text{bias}} &= \xi_{u,0} - \xi_{u,\text{desired}} & \xi_{l,\text{bias}} &= \xi_{l,0} - \xi_{l,\text{desired}} \end{aligned} \quad (13)$$

The model bias terms in (14) can be computed during the initial forward pass of the algorithm. The gradient calculation and back-propagation are disabled during this step to prevent the DNN from computing the loss and updating its weight with the unacceptable GEV parameters. After the initial iteration, the gradient calculation will be enabled and the bias terms will be subtracted from the DNN estimate of the GEV parameters in all subsequent iterations t:

$$\begin{aligned} \mu_t &\to \mu_t - \mu_{\text{bias}} & \sigma_t &\to \sigma_t - \sigma_{\text{bias}} \\ \xi_{u,t} &\to \xi_{u,t} - \xi_{u,\text{bias}} & \xi_{l,t} &\to \xi_{l,t} - \xi_{l,\text{bias}} \end{aligned} \quad (14)$$

Observe that, when μ_t is set to μ_0, then the debiased output $\mu_t - \mu_{\text{bias}}$ will be equal to μ_{desired}. By debiasing the output of the DNN in this way, we guarantee that the initial GEV parameters are reasonable and satisfy the GEV constraints.

4.3 Block Maxima Prediction

Given an input x_i, the DNN will estimate the GEV parameters needed to compute the block maxima $\hat{y}_i$ along with its upper and lower quantiles, $\hat{y}_{U,i}$ and $\hat{y}_{L,i}$, respectively. The quantiles are estimated using the formula given in (3). The GEV parameters are provided as input to a fully connected network (FCN) to generate the block maxima prediction, $\hat{y}_i$.

DeepExtrema employs a combination of the negative log-likelihood function of the GEV distribution and a least-square loss function to train the model. This enables the framework to simultaneously learn the GEV parameters and make accurate block maxima predictions. The loss function to be minimized by DeepExtrema is:

$$\begin{aligned} \mathcal{L} = \lambda_1 \mathcal{L}_{GEV} + \lambda_2 \sum_{i=1}^{n} (\xi_{u,i} - \xi_{l,i})^2 \\ + (1 - \lambda_1 - \lambda_2) \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \end{aligned} \quad (15)$$

where the first term corresponds to the negative log-likelihood function given in Equation (4), the second term minimizes the difference between the upper and lower-bound estimates of ξ, while the last term minimizes the prediction error of the block maxima. Here, λ_1 and λ_2 are hyperparameters to manage the trade-off between different factors of the loss function.

5 Experimental Evaluation

This section presents our experimental results comparing DeepExtrema against the various baseline methods. The code and datasets are available at https://anonymous.4open.science/r/DeepExtrema-IJCAI22--EE78/README.md.

5.1 Data

Synthetic Data

As the ground truth GEV parameters are often unknown, we have created a synthetic dataset to evaluate the performance of various methods in terms of their ability to correctly infer the parameters of the GEV distribution. The data is generated assuming the GEV parameters are functions of some input predictors $x \in \mathbb{R}^6$. We first generate x by random sampling from a uniform distribution. We then assume a non-linear mapping from x to the GEV parameters μ, σ, and ξ, via the following nonlinear equations:

$$\begin{aligned} \mu(x) &= w_\mu^T (\exp(x) + x) \qquad \sigma(x) = w_\sigma^T (\exp(x) + x) \\ \xi(x) &= w_\xi^T (\exp(x) + x) \end{aligned} \quad (16)$$

where w_μ, w_σ, and w_ξ are generated from a standard normal distribution. Using the generated μ, σ, and ξ parameters, we then randomly sample y from the GEV distribution governed by the GEV parameters. Here, y denotes the block maxima as it is generated from a GEV distribution. We created 8,192 block maxima values for our synthetic data.

Real-world data

We consider the following 3 datasets for our experiments.

Hurricane: This corresponds to tropical cyclone intensity data obtained from the HURDAT2 database [Landsea and Franklin, 2013]. There are altogether 3,111 hurricanes spanning the period between 1851 and 2019. For each hurricane,

Negative Log-likelihood		
DeepExtrema	Ground Truth	Global GEV Estimate
4410	4451	4745

Table 1: Negative log-likelihood of `DeepExtrema` with respect to ground truth and global parameter estimation using synthetic data

wind speeds (intensities) were reported at every 6-hour interval. We consider only hurricanes that have at least 24-time steps at minimum for our experiments. For each hurricane, we have created non-overlapping time windows of length 24 time steps (6 days). We use the first 16 time steps (4 days) in the window as the predictor variables and the block maxima of the last 8 time steps (2 days) as the target variable.

Solar: This corresponds to half-hourly energy use (kWh) for 55 families over the course of 284 days from Ausgrid database [Aus, 2013]. We preprocess the data by creating non-overlapping time windows of length 192 time steps (4 days). We use the first 144 time steps (3 days) in the window as the predictor variables and the block maxima of the last 48 time steps (1 day) as the target variable.

Weather: We have used a weather dataset from the Kaggle competition [Muthukumar, 2017]. The data is based on hourly temperature data for a city over a ten-year period. We use the first 16 time steps (16 hours) in the window as the predictor variables and the block maxima of the last 8 time steps (8 hours) as the target variable.

5.2 Experimental Setup

For evaluation purposes, we split the data into separate training, validation, testing with a ratio of 7:2:1. The data is standardized to have zero mean and unit variance. We compare `DeepExtrema` against the following baseline methods: (1) Persistence, which uses the block maxima value from the previous time step as its predicted value, (2) fully-connected network (FCN), (3) LSTM, (4) Transformer, (5) DeepPIPE [Wang *et al.*, 2020], and (6) EVL [Ding *et al.*, 2019]. We will use the following metrics to evaluate the performance of the methods: (1) Root mean squared error (RMSE) and correlation between the predicted and ground truth block maxima and (2) Negative log-likelihood (for synthetic data). Finally, hyperparameter tuning is performed by assessing the model performance on the validation set. The hyperparameters of the baseline and the proposed methods are selected using Ray Tune, a tuning framework with ASHA (asynchronous successive halving algorithm) scheduler for early stopping.

5.3 Experimental Results

Results on Synthetic Data

In this experiment, we have compared the performance of `DeepExtrema` against using a single (global) GEV parameter to fit the data. Based on the results shown in Table 1, `DeepExtrema` achieves a significantly lower negative log-likelihood of 4410 compared to the negative log-likelihood for global GEV estimate, which is 4745. This result supports the assumption that each block maxima comes from different GEV distributions rather than a single (global) GEV distribution. The results also suggest that the negative log likelihood estimated by `DeepExtrema` is lower than that for the ground truth.

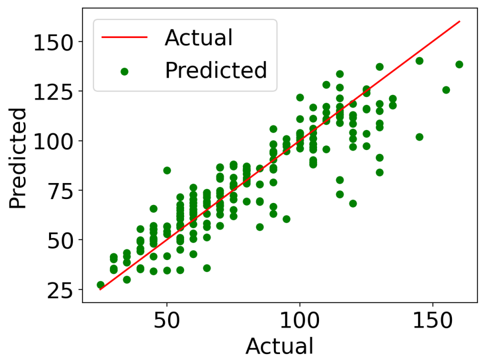

Figure 4: Comparison between actual and predicted block maxima of hurricane intensities for `DeepExtrema`.

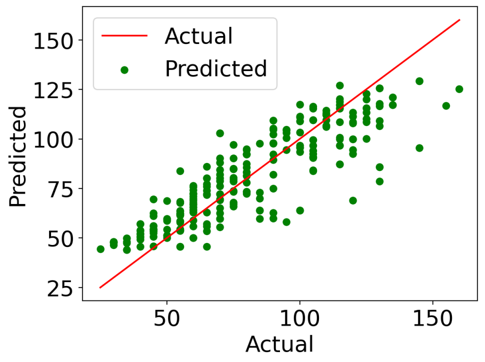

Figure 5: Comparison between actual and predicted block maxima of hurricane intensities for `EVL` [Ding *et al.*, 2019].

Methods	Hurricanes		Ausgrid		Weather	
	RMSE	Correlation	RMSE	Correlation	RMSE	Correlation
Persistence	28.6	0.6	0.84	0.65	4.16	0.96
FCN	14.14	0.87	0.69	0.65	2.5	0.97
LSTM	13.31	0.88	0.65	0.64	2.53	0.97
Transformer	13.89	0.88	0.68	0.62	2.43	**0.98**
DeepPIPE	13.67	0.87	0.71	0.59	2.59	0.94
EVL	15.72	0.83	0.75	0.54	2.71	0.90
DeepExtrema	**12.81**	**0.90**	**0.63**	**0.67**	**2.27**	0.97

Table 2: Performance comparison on real world data.

Results on Real-world Data

Evaluation on real-world data shows that `DeepExtrema` outperforms other baseline methods used for comparison for all data sets (see Table 2). For RMSE, `DeepExtrema` generates lower RMSE compared to all the baselines on all 3 datasets, whereas for correlation, `DeepExtrema` outperforms the baselines on 2 of the 3 datasets.

To demonstrate how well the model predicts the extreme values, Figure 4 shows a scatter plot of the actual versus predicted values generated by `DeepExtrema` on the test set of the hurricane intensity data. The results suggest that `DeepExtrema` can accurately predict the hurricane intensities for a wide range of values, especially those below 140

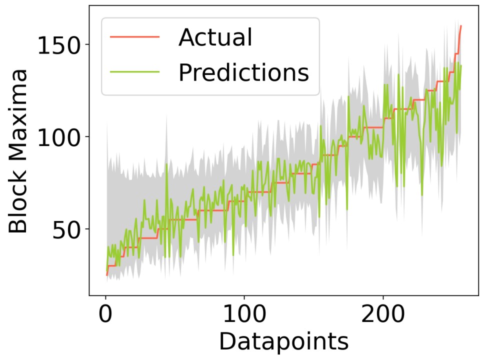

Figure 6: 90% confidence interval of the hurricane intensity predictions for `DeepExtrema`, sorted in increasing block maxima values. Ground truth values are shown in red.

knots. `DeepExtrema` also does a better job at predicting the high intensity hurricanes compared to `EVL` [Ding *et al.*, 2019], as illustrated in Figure 5. Figure 6 shows the 90% confidence interval of the predictions. Apart from the point and quantile estimations, `DeepExtrema` can also estimate the GEV parameter values for each hurricane.

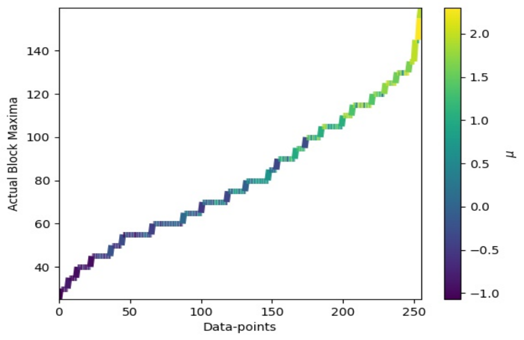

Figure 7: Magnitude of μ with respect increasing intensities of hurricane.

According to Figure 8, the magnitude of ξ decreases in the upper tail part of the distribution, which is consistent our expectation. Similarly, Figures 7 and 9 show an increasing trend in μ and a decreasing trend in σ as the block maxima value of the hurricane intensity increases. So, tail end extremes, i.e., upper extremes have much larger μ, smaller σ, and lower ξ. The observed trend in GEV parameters enable us to better characterize properties of the distribution of hurricane intensities.

Ablation Studies

The hyperparameter $\lambda_1 + \lambda_2$ in the objective function of `DeepMaxima` denotes the trade-off between GEV loss and RMSE loss. Experimental results show that with the decrease of GEV loss weight, the RMSE of block maxima also decreases (Table 3). It suggests incorporation of GEV theory plays a positive role in block maxima estimations rather than using mean squared-based loss function.

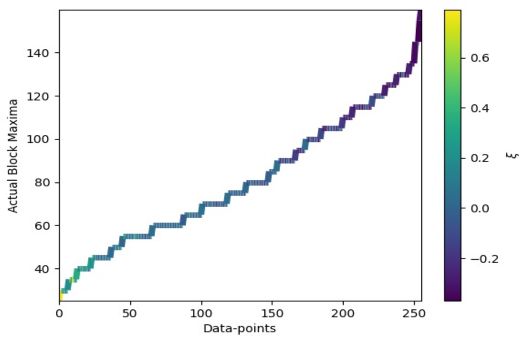

Figure 8: Magnitude of ξ with respect increasing intensities of hurricane.

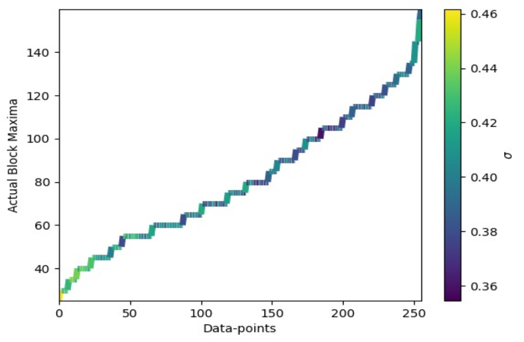

Figure 9: Magnitude of σ with respect increasing intensities of hurricane.

Weights on GEV Loss	RMSE of Block maxima		
	Hurricanes	Ausgrid	Weather
$\lambda_1 + \lambda_2 = 0.0$	13.28	0.68	2.52
$\lambda_1 + \lambda_2 = 0.5$	13.03	**0.63**	2.41
$\lambda_1 + \lambda_2 = 0.9$	**12.81**	0.64	**2.27**

Table 3: Effects of different weights on GEV Loss.

6 Conclusion

This paper presents a novel deep learning framework called `DeepExtrema` that combines extreme value theory with deep learning to address the challenges of predicting extremes in time series. We offer a reformulation and reparameterization technique for satisfying constraints as well as a model bias offset technique for proper model initialization. We evaluated our framework on synthetic and real-world data and showed its effectiveness. For future work, we plan to extend the formulation to enable more complex deep learning architectures such as those using an attention mechanism. In addition, the framework will be extended to model extremes in spatio-temporal data.

References

[Aliabadi *et al.*, 2020] Majid Moradi Aliabadi, Hajar Emami, Ming Dong, and Yinlun Huang. Attention-based recurrent neural network for multistep-ahead prediction of process performance. *Computers & Chemical Engineering*, 140:106931, 2020.

[Aus, 2013] Solar home electricity data, Dec 2013.

[Bai *et al.*, 2018] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv:1803.01271 [cs]*, April 2018. arXiv: 1803.01271.

[Bishop, 2006] Christopher M Bishop. *Pattern recognition*. Springer, New York, 2006.

[Coles *et al.*, 2001] Stuart Coles, Joanna Bawa, Lesley Trenner, and Pat Dorazio. *An introduction to statistical modeling of extreme values*, volume 208. Springer, 2001.

[Ding *et al.*, 2019] Daizong Ding, Mi Zhang, Xudong Pan, Min Yang, and Xiangnan He. Modeling extreme events in time series prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1114–1122, 2019.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Kharin and Zwiers, 2005] Viatcheslav V Kharin and Francis W Zwiers. Estimating extremes in transient climate change simulations. *Journal of Climate*, 18(8):1156–1173, 2005.

[Landsea and Franklin, 2013] Christopher W Landsea and James L Franklin. Atlantic hurricane database uncertainty and presentation of a new database format. *Monthly Weather Review*, 141(10):3576–3592, 2013.

[Laptev *et al.*, 2017] Nikolay Laptev, Jason Yosinski, Li Erran Li, and Slawek Smyl. Time-series extreme event forecasting with neural networks at uber. In *International conference on machine learning*, volume 34, pages 1–5, 2017.

[Masum *et al.*, 2018] Shamsul Masum, Ying Liu, and John Chiverton. Multi-step time series forecasting of electric load using machine learning models. In *International conference on artificial intelligence and soft computing*, pages 148–159. Springer, 2018.

[Muthukumar, 2017] J Muthukumar. Weather dataset, Dec 2017.

[Peng *et al.*, 2018] Chenglei Peng, Yang Li, Yao Yu, Yu Zhou, and Sidan Du. Multi-step-ahead host load prediction with gru based encoder-decoder in cloud computing. In *2018 10th International Conference on Knowledge and Smart Technology (KST)*, pages 186–191. IEEE, 2018.

[Polson and Sokolov, 2020] Michael Polson and Vadim Sokolov. Deep learning for energy markets. *Applied Stochastic Models in Business and Industry*, 36(1):195–209, 2020.

[Sagheer and Kotb, 2019] Alaa Sagheer and Mostafa Kotb. Time series forecasting of petroleum production using deep lstm recurrent networks. *Neurocomputing*, 323:203–213, 2019.

[Smith, 1985] Richard L. Smith. Maximum likelihood estimation in a class of nonregular cases. *Biometrika*, 72(1):67–90, 1985.

[US GAO, 2020] US GAO. Natural disasters: Economic effects of hurricanes katrina, sandy, harvey, and irma. https://www.gao.gov/products/gao-20-633r, 2020. Accessed: 2021-04-09.

[Wang *et al.*, 2020] Bin Wang, Tianrui Li, Zheng Yan, Guangquan Zhang, and Jie Lu. Deeppipe: A distribution-free uncertainty quantification approach for time series forecasting. *Neurocomputing*, 397:11–19, 2020.

[Wilson *et al.*, 2022] Tyler Wilson, Pang-Ning Tan, and Lifeng Luo. Deepgpd: A deep learning approach for modeling geospatio-temporal extreme events. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, 2022.

[Yang *et al.*, 2015] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[Zerveas *et al.*, 2021] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2114–2124, 2021.

[Zhang *et al.*, 2019] Xuan Zhang, Xun Liang, Aakas Zhiyuli, Shusen Zhang, Rui Xu, and Bo Wu. At-lstm: An attention-based lstm model for financial time series prediction. In *IOP Conference Series: Materials Science and Engineering*, volume 569, page 052037. IOP Publishing, 2019.

[Zhao *et al.*, 2017] Bendong Zhao, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169, 2017.