

## A APPENDIX

### A.1 OVERVIEW OF CHEBYSHEV APPROXIMATION

The Chebyshev polynomials denoted as mappings  $T_k: [-1, 1] \rightarrow \mathbb{R}$  are given by the recurrent formula

$$\begin{aligned} T_0(t) &:= 1, \\ T_1(t) &:= t, \\ T_{k+1}(t) &:= 2tT_k(t) - T_{k-1}(t), \quad k \geq 1. \end{aligned}$$

Alternatively, the recurrence is equivalent to  $T_k(t) = \cos(k \cos^{-1}(t))$ .

Chebyshev polynomials have been widely studied, see (Dahlquist & Björck (2008)) for a general overview, and their zeros in the interval  $[-1, 1]$  are well-known. In fact, the zeros of the  $k$ th Chebyshev polynomial are located at the points

$$t_k := \cos\left(\frac{\pi(k + \frac{1}{2})}{d}\right), \quad 0 \leq k \leq d. \quad (15)$$

An explicit computation shows that they are orthogonal in the interval  $[-1, 1]$  over a weight  $(1 - t^2)^{-1/2}$ . For all indices  $i \neq j$ , when introducing the change of variable  $t = \cos x$  and trigonometric properties of the cosine, we see that

$$\int_0^\pi \cos(ix) \cos(jx) dx = \frac{1}{2} \int_0^\pi \cos((j-i)x) + \cos((j+i)x) dx = 0.$$

This orthogonality between Chebyshev polynomials leads to a linearly independent system. Thus,  $\{T_k(t)\}_{k=0}^{d-1}$  is a vectorial subspace of dimension  $d$ , meaning that we can easily apply the normal equations interpreted in the discrete case as follows: if  $f: [-1, 1] \rightarrow \mathbb{R}$  is a continuous function and we consider the discrete points  $\{f(t_k)\}_{k=0}^{d-1}$  where  $t_k$  is the  $k$ th zero of the  $T_d$  Chebyshev polynomial (see Equation 15), then  $f$  can be approximated by

$$f(t) \approx p(t) := c_0 T_0(t) + \dots + c_{d-1} T_{d-1}(t), \quad \text{for all } t \in [-1, 1]. \quad (16)$$

with

$$c_j = \sum_{k=0}^{d-1} f(t_k) \cos\left(\frac{j\pi(k + \frac{1}{2})}{d}\right), \quad 0 \leq j < d. \quad (17)$$

Note that Equation 17 is simply a matrix times a vector, where the entries of the matrix are the cosines and the vector is  $\{f(t_k)\}_{k=0}^{d-1}$ . Therefore, a direct computation of Equation 17 has a complexity equal to the matrix-vector multiplication, i.e.  $\Theta(d^2)$ . However, there is a fast computation for an expression like the one in Equation 17.

In fact, an expression like the one in Equation 17 is denoted in signal theory as a Discrete Cosine Transform (DCT). If values are given to  $j$  and  $k$  in Equation 17, then one immediately realises that many of the cosine values are the same due to the trigonometric properties of the cosine. Following the principles in the Fast Fourier Transform (FFT), we obtain an efficient algorithm with complexity  $\Theta(d \log d)$ , which is commonly implemented in numerical libraries and known as type 2 DCT (DCT-II).

As happens with the classical FFT, the DCT is not normalised and to avoid this lack of normalisation, i.e. to make the  $c_j$  quantitatively independent of  $d$ , it is common to consider the  $c_j$  multiplied by a factor like  $2/d$ .

**Remark A.1.** The approximation in Equation 16 exhibits some properties and advantages. One of them is to approximate, uniformly in  $[-1, 1]$ , an arbitrary function  $f$  with a finite collection of number  $c_k$  rather than a table of values in  $t_k$ .

**Remark A.2.** The assumption that  $f$  is defined in  $[-1, 1]$  is not a restriction, since a bijection  $\varphi_{a,b}: [-1, 1] \rightarrow [a, b]$  can always be considered. Indeed, if  $g: [a, b] \rightarrow \mathbb{R}$  is a mapping, then we can consider  $f := g \circ \varphi_{a,b}$ . Therefore, given that in the main text we have  $\tau \in [0, 1]$ , then we can consider the interval  $[0, 1]$  using a  $\varphi_{0,1}$ .

**Remark A.3.** It is important to query the accuracy of the approximation in Equation 16. This question is equivalent to quantifying the norm  $\|f - p\|$  or the pointwise distance  $|f(t) - p(t)|$ . This question is crucial in determining how well  $p$  approximates  $f$ . Numerous references can be found in this respect, mostly by numerical analysts, e.g. in (Dahlquist & Björck (2008); Trefethen (2008); Majidian (2017)), or (Elliott (1968)). As a general conclusion, we can say that Equation 16 will be a good (uniform) approximation when  $d$  is large enough. The quantification of “enough” here depends on how regular the function  $f$  is. If  $f$  is analytic, the  $d$  will be smaller than the  $C^\infty$  case and even smaller than the finite differentiable case.

**Remark A.4.** The expression in Equation 16 can be evaluated at a given  $t$  without constructing the polynomials  $T_k(t)$ . In fact, such an evaluation can be done in linear time with  $d$ . Algorithm 1 shows which steps to follow to evaluate a Chebyshev expansion in the ChePAN case when the  $[0, 1]$  representation is considered.

## A.2 THE CLENSHAW-CURTIS NETWORK (CCN)

The Clenshaw-Curtis Network (CCN), which provides a partially monotonic function with respect to the quantile input  $\tau$ , comprises the step between the UMNN, referenced in Section 4, and the main proposal of the current article, the ChePAN, presented in Section 5. However, given that this model cannot be generally applied to constrained black-box uncertainty modelling, we have decided to keep its definition for the Appendix.

The CCN contains a neural network,  $\phi: [0, 1] \times \mathbb{R}^D \rightarrow \mathbb{R}_+$ , which models the derivative with respect to the quantile  $\tau$  of the Quantile Regression loss function in Equation 1, similar to the ChePAN. Furthermore, the CCN also uses the Chebyshev approximation described in Section A.1. In contrast with the ChePAN, however, it restarts the computation of the Chebyshev coefficients at each given  $\tau$ , as explained below.

With the CCN, it is sometimes appropriate to change the Chebyshev nodes of Equation 3 for the points that are the extrema of the Chebyshev polynomials. There are  $k + 1$  extrema of the  $k$ th Chebyshev polynomial located at the points

$$\cos\left(\frac{\pi k}{d}\right), \quad 0 \leq k < d.$$

for  $k = 0, \dots, d - 1$ . After the bijection  $\varphi_{0,\tau}$  from Remark A.2, they become

$$\bar{t}_k^d := \frac{\tau}{2} \cos\left(\frac{\pi k}{d}\right) + \frac{\tau}{2}, \quad 0 \leq k \leq d. \quad (18)$$

They have the property that  $\bar{t}_k^{2d} = \bar{t}_{k/2}^d$ , which means that information can be reused when the value of  $d$  needs to be increased. Besides this small detail, the rest of the content is adaptable to these new roots. Indeed, if  $f: [0, \tau] \rightarrow \mathbb{R}$  is continuous, we consider the discrete points  $\{f(\bar{t}_k^d)\}_{k=0}^d$ , then  $f$  can be approximated by

$$f(t) \approx \bar{p}(t) := \bar{c}_0 T_0(\varphi_{0,\tau}(t)) + \dots + \bar{c}_d T_d(\varphi_{0,\tau}(t)), \quad \text{for all } t \in [0, \tau],$$

with  $\varphi_{0,\tau}(t) := 2t/\tau - 1$  and

$$\bar{c}_j = \sum_{k=0}^d f(\bar{t}_k^d) \cos\left(\frac{j\pi k}{d}\right), \quad 0 \leq j \leq d, \quad (19)$$

which, as in Equation 17, is a matrix-vector multiplication. Similarly, as in Section A.1, there is a fast algorithm that performs this matrix-vector with complexity  $\Theta(d \log d)$ . This algorithm is known as the type 1 Discrete Cosine Transform, or DCT-I. This algorithm produces the unnormalised coefficients Equation 19, which as with the case of Equation 17, must be normalised by a factor, for instance,  $2/d$ .

Similarly to the ChePAN in Section 5, we use Equation 6 to deduce the Chebyshev coefficients of the integral by the recurrence given in Equation 8, but now the  $C_0$  is chosen such that the value of the integral of the small polynomial  $p$  is equal to a  $\beta(\mathbf{x})$ , i.e.,  $P(0, \mathbf{x}; d) = \beta(\mathbf{x})$ . All of this leads to the final Clenshaw-Curtis expression used in the CCN,

$$P(\tau, \mathbf{x}; d) = \tau \left( \frac{c_0(\tau, \mathbf{x})}{2} - \sum_{k=1}^{\lfloor d/2 \rfloor} \frac{c_{2k}(\tau, \mathbf{x})}{4k^2 - 1} \right) + \beta(\mathbf{x}), \quad (20)$$

where  $\lfloor d/2 \rfloor$  denotes the floor value of the integer division  $d/2$ .

Note that Equation 20 has a  $\tau$  dependency on all the Chebyshev coefficients of  $c_k(\tau, \mathbf{x})$ , which comes from the fact that  $P$  is not uniform on  $\tau$ .

### A.3 A BRIEF COMPARISON OF THE CCN AND THE CHEPAN

Both methods are based on the UMNN described in Section 4 and both have been modified in such a way as to provide a partially monotonic function in the input  $\tau$ . However, the ChePAN gives a uniform representation of the NN across all of the quantile domain  $[0, 1]$ , while the CCN only gives pointwise values of the NN. This crucial difference derives from the fact that the nodes in Equation 18 explicitly depend on  $\tau$  with the CCN, which does not happen with the ChePAN nodes (see Equation 3).

Another consequence of this dependency on  $\tau$  in the CCN is that the final expression in Equation 20 obviously depends on the quantile  $\tau$  as well. This makes it impossible to impose any condition for the black box, except in the case of the quantile 0. In contrast, the ChePAN provides for much more freedom in the choice of the constant of integration and is therefore more flexible in adding uncertainty estimation via a black-box predictive system.

### A.4 DEDUCTION OF THE $C_0$ COEFFICIENT IN THE CHEPAN

In this subsection, we explain how to deduce all the formulas proposed in Section 5.1 in more detail.

Equation 9 is obtained by considering  $P(0, \mathbf{x}; d) = \beta(\mathbf{x})$  in Equation 7. This is straightforward because  $T_k(-1) = (-1)^k$ .

Similarly, Equation 10 is deduced by imposing  $P(1, \mathbf{x}; d) = \beta(\mathbf{x})$ , which is immediate due to the fact that  $T_k(1) = 1$ .

Equation 11 is proven in a similar manner because  $T_k(0) = 0$  when  $k$  is odd and  $T_k(0) = (-1)^{k/2}$  when  $k$  is even.

Finally, Equation 12 considers the definition for the mean of a random variable. Changing coordinates leads to the computation of the integral

$$\int_{-1}^1 t T_k(t) dt. \quad (21)$$

Following this, the symmetries of the Chebyshev polynomials leads to the equality

$$\int_{-1}^1 t T_k(t) dt = (1 + (-1)^{k+1}) \int_0^1 t T_k(t) dt.$$

Then if  $k$  is even, Equation 21 is zero. If  $k$  is now assumed to be an odd integer, then by defining the Chebyshev polynomials,  $2T_k(t) = T_{k+1}(t) + T_{k-1}(t)$  and then according to Equation 6,

$$\int_{-1}^1 t T_k(t) dt = \left. \frac{T_{k-2}(t)}{2(k-2)} - \frac{T_{k+2}(t)}{2(k+2)} \right|_0^1 = \frac{2}{k^2 - 4}.$$

By now imposing  $\int_{-1}^1 t P(t, \mathbf{x}; d) dt = \beta(\mathbf{x})$ , we can deduce Equation 12.

### A.5 THE PSEUDO-CODE OF THE CHEPAN

The Clenshaw method, see (Clenshaw (1955)), or its stable numerical error version (Elliott (1968); Newbery (1974)), can be used to evaluate Equation 4 or Equation 7 for a value  $\tau$  in  $[0, 1]$ . Let us briefly summarise the evaluation at  $\tau$  of  $p(\tau, \mathbf{x}; d)$  in Equation 4 (or  $P(\tau, \mathbf{x}; d)$  in Equation 7), whose numerical complexity is  $\Theta(d)$ .

The pseudo-code of this framework is shown in Algorithm 4 and can be implemented using any automatic differentiation library. Furthermore, a TensorFlow (Abadi et al. (2016)) and Keras (Chollet et al. (2019)) implementation, following this pseudo-code, will be found in the Github repository that will be made public in the camera-ready version of this paper.

**Algorithm 1** Evaluation of Eq. 4 or Eq. 7 at  $\tau \in [0, 1]$ 


---

```

1: procedure EVAL_CHEB( $\tau, c_0(\mathbf{x}), \dots, c_{d-1}(\mathbf{x})$ )
2:    $d_1(\mathbf{x}) \leftarrow d_2(\mathbf{x}) \leftarrow d_3(\mathbf{x}) \leftarrow 0$ .
3:    $\sigma \leftarrow 2\tau - 1$ .
4:   for  $k = d - 1, d - 2, \dots, 1$  do
5:      $d_3(\mathbf{x}) \leftarrow d_1(\mathbf{x})$ .
6:      $d_1(\mathbf{x}) \leftarrow 2\sigma d_1(\mathbf{x}) - d_2(\mathbf{x}) + c_k(\mathbf{x})$ .
7:      $d_2(\mathbf{x}) \leftarrow d_3(\mathbf{x})$ .
8:   return  $\sigma d_1(\mathbf{x}) - d_2(\mathbf{x}) + 0.5c_0(\mathbf{x})$ .

```

---

**Algorithm 2** Definitions and functions used for the following algorithms

---

- $\triangleright \mathbf{x}$  has batch size and number of features as shape, i.e.  $[bs, D]$ .
- $\triangleright \text{RS}(\text{tensor}, \text{shape})$ : reshape *tensor* to *shape*.
- $\triangleright \text{RP}(\text{tensor}, n)$ : repeats  $n$  times the last dimension of *tensor*.
- $\triangleright \text{CC}(T_1, T_2)$ : concatenate  $T_1$  and  $T_2$  using their last dimension.

---

**Algorithm 3** Obtain all the ChePAN coefficients

---

```

1: procedure CHEB_CS( $\mathbf{x}, d, \phi, \beta$ )
2:    $\{t_k\}_{k=0}^{d-1} \leftarrow \text{Apply Eq. 3}$ 
3:    $\{t'_k\}_{k=0}^{d-1} \leftarrow \text{RS}(\text{RP}(\{t_k\}_{k=0}^{d-1}, bs), [bs \cdot d, 1])$ 
4:    $\mathbf{x}' \leftarrow \text{RS}(\text{RP}(\mathbf{x}, d), [bs \cdot d, D])$ 
5:    $\mathbf{i} \leftarrow \text{CC}(\{t'_k\}_{k=0}^{d-1}, \mathbf{x}')$   $\triangleright [bs \cdot d, D + 1]$ .
6:    $\mathbf{o} \leftarrow \phi(\mathbf{i})$   $\triangleright$  Apply any NN function  $\phi: [0, 1] \times \mathbb{R}^D \rightarrow \mathbb{R}^+$ .
7:    $\{c_k(\mathbf{x})\}_{k=0}^{d-1} \leftarrow \text{DCT-II}(\mathbf{o}, d)$   $\triangleright$  Transformation.
8:    $\{C_k(\mathbf{x})\}_{k=1}^{d-1} \leftarrow \text{Integration step wrt } \{c_k(\mathbf{x})\}_{k=0}^{d-1}$ 
9:    $C_0(\mathbf{x}) \leftarrow 2\beta(\mathbf{x}) - 2 \sum_{k=1}^{d-1} C_k(\mathbf{x})(-1)^k$   $\triangleright$  Eq. 9
10:  return  $\{C_k(\mathbf{x})\}_{k=0}^{d-1}, \{c_k(\mathbf{x})\}_{k=0}^{d-1}$ .

```

---

**Algorithm 4** How to build the ChePAN model by using any deep learning architecture for regression

---

```

1: procedure BUILD_CHEPAN_GRAPH( $\mathbf{x}, y, d, \phi, \beta, N_\tau$ )
2:    $\triangleright N_\tau$  is the number of non-roots to evaluate.
3:    $\{C_k(\mathbf{x})\}_{k=0}^{d-1}, \{c_k(\mathbf{x})\}_{k=0}^{d-1} \leftarrow \text{CHEB\_CS}(\mathbf{x}, d, \phi, \beta)$ 
4:    $\boldsymbol{\tau} \leftarrow \mathcal{U}(0, 1)$   $\triangleright \boldsymbol{\tau}$  must has  $[bs \cdot N_\tau, 1]$  shape.
5:    $\mathbf{o}_P \leftarrow \text{EVAL\_CHEB}(\boldsymbol{\tau}, C_0(\mathbf{x}), \dots, C_{d-1}(\mathbf{x}))$ 
6:    $\mathcal{L} \leftarrow (y - \mathbf{o}_P) \cdot (\boldsymbol{\tau} - \mathbb{1}[y < \mathbf{o}_P])$   $\triangleright$  Eq. 1 loss.
7:   return  $\mathcal{L}$ 

```

---

**A.6 DATA SETS AND MODEL HYPER-PARAMETERS USED DURING THE EVALUATION**

To ensure the strictly positive output values of  $\phi$  in the CCN and the ChePAN, the final output will have a softplus function (Zheng et al. (2015)) with a certain shift, specifically  $\phi(\tau, \mathbf{x}) = 10^{-3} + \text{softplus}(NN(\tau, \mathbf{x}) + 10^{-5})$ , where  $NN(\tau, \mathbf{x})$  is the output of the neural network. All internal activation functions will be ReLU for all of the models. Furthermore, all experiments will be trained using an early stopping training policy, with 100 epochs of patience for all compared methods.

**The Synthetic Heterogeneous Data Set** To demonstrate the aleatoric enhancement resulting from Section 5.1, we selected a recently published synthetic data set proposed in (Brando et al. (2019)) and shown in Figure 3. It has 4 different synthetic distributions depending on the input value. Specifically, the data set has 3,800 points that came from a Beta( $\alpha = 0.5, \beta = 1$ ) distribution, a  $\mathcal{N}(\mu = 3 \cdot \cos x_i - 2, \sigma = |3 \cdot \cos x_i - 2|)$  distribution, an increasing uniform random distribution and a mixture of three uniform distributions  $\mathcal{U}(8, 0.5)$ ,  $\mathcal{U}(1, 3)$  and  $\mathcal{U}(-4.5, 1.5)$ .

Following (Brando et al. (2019)), the architecture used for  $\mu$  and  $\sigma$  in the N model and  $\phi$  and  $\beta$  functions of the ChePAN consists of 4 dense layers with output dimensions 120, 60, 10 and 1 for each function, respectively (see the pseudo-code in Section A.5).

**The Year Prediction Million Song Data set (Year-MSD)** This consists of 515,345 songs comprising 90 input audio features to predict the release year of each song from 1922–2011. To evaluate this data set, 80% was considered for the training set, 10% for the validation set and, finally, 10% for the test set. All years were normalised between 0 and 1. Furthermore, each audio feature was normalised subtracting by the minimum value and dividing by the new maximum value considering each feature of the training set.

**Room price forecasting (RPF)** Following the procedure to obtain the data described in (Brando et al. (2019)), the publicly available information from the Inside Airbnb platform (Cox (2019)) corresponds to the cities of Barcelona (BCN) with 36,367 flats, and Vancouver (YVC) with 11,497 flats. The goal of the regression problem is to predict the rental price per night of each flat in their respective currency using the properties available for it. These properties are: district number, neighbourhood number, room type and property type, number of bathrooms, accommodation values, and longitude and latitude normalised according to the minimum and maximums of the corresponding city. The same training, validation and test sets were used as the original proposal to make the evaluation.

Similarly to (Brando et al. (2019)), the selected architecture for the  $\mu$  and  $\sigma$  in the N model,  $\mu$  and  $b$  in the LP and  $\phi$  and  $\beta$  functions of the ChePAN consists of 6 dense layers with output dimensions 120, 120, 60, 60, 10 and 1 for each function, respectively. On the other hand, for the Year-MSD problem we used the same architecture for all the proposed models (see the pseudo-code in Section A.5 for further details).

**Improving an Inaccurate Predictive System** Given a 5-depth Random Forest Regressor (RF) with 100 estimators, we obtained an imprecise approximation of the distribution to be predicted minimising the mean square error, as shown in Figure 3. Alternatively, an equivalent implementation with an XGBoost model (Chen & Guestrin (2016)) is considered.

Since this statistic corresponds to the mode of a normal distribution, we can consider two different approaches to tackle the aleatoric uncertainty estimation of the RF or the XGBoost (the black box). Firstly, we can consider a conditional normal distribution, fixing the mean as the predicted values of the black box and adjusting the variance as a  $\sigma(\mathbf{x})$  neural network function that minimises the corresponding likelihood (which we shall call *RF-N*). Similarly, we can perform the same optimisation process with a Laplace distribution. Alternatively, we can use the proposed ChePAN model considering the black-box predictor as  $\beta$  and adjusting  $\phi$  (*RF-ChePAN* or *XGBoost-ChePAN*). Table 1 shows the mean and standard deviation of 10 executed models of the quantile regression loss values for the same 10,000 randomly sampled quantiles.

Enriching a pointwise prediction involves relating the aleatoric estimation of the distribution to the previous pointwise prediction. Thus, we do not consider approximating an IQN or SQR model because we need to link the pointwise prediction of the RF and the predicted quantiles. As Figure 3 shows, there is a noisy behaviour in the predicted quantiles similar to the RF prediction. This indicates that the quantile values depend on the RF prediction, as expected.

**Enriching an Accurate Predictive System** For comparison with an optimal black-box estimator, we can repeat the previous experiment learning the  $\beta$  function (or  $\mu$  in the exponential power distributions). The results are shown in the lower part of Figure 3 and the last 3 rows of Table 1.

**UCI Data Sets** In order to verify the results in other well-known data sets, we applied all the proposed models to 9 different UCI Machine learning data sets (Dua & Graff (2017b)). These data sets are commonly used for various regression tasks. Specifically, we used the splits proposed in (Hernández-Lobato & Adams (2015a)) and widely used in later works (Gal & Ghahramani (2015); Lakshminarayanan et al. (2017)).

Regarding the trained models, all share the same number of parameters for all of the models: a single dense hidden layer of 200 neurons. However, as N, LP and ChePAN require two different

architectures for their functions, we decided to assign half of the neurons of the single hidden layer to each of them. The ChePAN was trained with a fixed degree of 20.

#### A.7 DIMENSION OF UCI DATA SETS

	Housing	Concrete	Energy	Kin8nm	Naval	Power	Protein	Wine	Yacht
Train	364	741	552	5898	8592	6888	32925	1151	221
Val	91	186	139	1475	2149	1723	8232	288	56
Test	51	103	77	819	1193	957	4573	160	31

Table 2: Number of data points for each UCI data set proposed in (Hernández-Lobato & Adams (2015a)).

Table 2 contains the number of points for each UCI dataset used in the article.

#### A.8 CALIBRATION IN PREDICTION INTERVALS

	$XGBoost$ -N	$XGBoost$ -LP	$XGBoost$ -ChePAN
Concrete	$46.55 \pm 6.7(3.34 \pm 0.42)$	$54.66 \pm 5.5(4.38 \pm 0.3)$	<b><math>85.58 \pm 4.7(17.7 \pm 21.)</math></b>
Power	$79.44 \pm 3.5(6.99 \pm 0.4)$	$88.23 \pm 3.7(9.19 \pm 1.2)$	<b><math>92.93 \pm 2.8(12.03 \pm 1.6)</math></b>
Wine	<b><math>95.78 \pm 1.8(2.66 \pm 0.2)</math></b>	<b><math>96.7 \pm 1.8(2.77 \pm 0.2)</math></b>	<b><math>95.81 \pm 1.89(3.86 \pm 0.6)</math></b>
Yacht	<b><math>93.22 \pm 5.1(2.42 \pm 0.4)</math></b>	<b><math>94.03 \pm 4.6(2.45 \pm 0.4)</math></b>	<b><math>99.19 \pm 1.7(8.16 \pm 1.3)</math></b>
Naval	<b><math>99.52 \pm 1.3(0.05 \pm 0.0)</math></b>	<b><math>99.68 \pm 1.3(0.08 \pm 0.0)</math></b>	<b><math>96.95 \pm 4.2(0.11 \pm 0.0)</math></b>
Energy	<b><math>95.5 \pm 4.4(2.16 \pm 0.5)</math></b>	<b><math>95.8 \pm 3.7(2.11 \pm 0.3)</math></b>	$99.48 \pm 0.64(6.17 \pm 1.1)$
Boston	$51.9 \pm 10.(3.25 \pm 0.4)$	$63.73 \pm 6.6(4.2 \pm 0.3)$	<b><math>89.12 \pm 3.7(10.4 \pm 2.0)</math></b>
Kin8nm	<b><math>94.77 \pm 1.0(0.67 \pm 0.0)</math></b>	$98.33 \pm 0.7(0.87 \pm 0.0)$	<b><math>95.51 \pm 1.8(1.39 \pm 0.1)</math></b>

Table 3: Mean and standard deviation, mean  $\pm$  std, of the 100-PICP (MPIW) value between the 0.975 and 0.025 quantile of the `black box`-uncertainty wrapper for the different test set folds presented in (Hernández-Lobato & Adams (2015b)).

Table 3 shows the calibration results of using the Prediction Interval Coverage Probability (PICP) and the Mean Prediction Interval Width (MPIW) for all the data sets used in (Tagasovska & Lopez-Paz (2019)).

#### A.9 EXTRA GENERAL CALIBRATION FIGURES

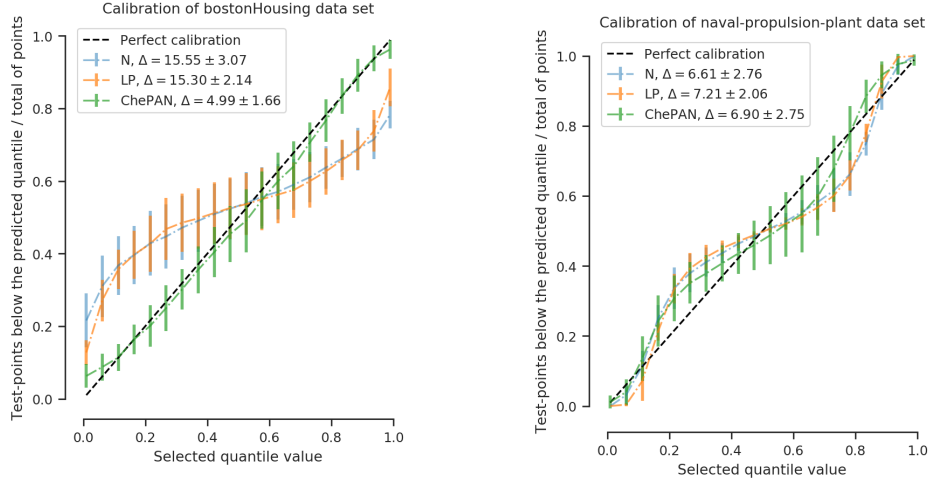
Figure 5 shows two examples from Table 4 based on a metric such as that presented in Equation 14. Specifically, the concrete metric for calculating each point of the curve corresponds to the non-distance evaluation for each selected quantile value,

$$\text{specific-Cal}(f; X_{test}, Y_{test}, \tau) = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \mathbb{1}[y_i < f(\tau_j, \mathbf{x}_i)] \quad (22)$$

where the black line corresponds to a perfect calibration and the further away from that curve, the worse the calibration is. We see that the ChePAN achieves an overall strong performance compared to the other baselines.

#### A.10 SELECTING DIFFERENT STATISTICS

Figure 6 shows the result of approximating the heterogeneous synthetic data set considering  $\beta$  as the other statistics proposed in Section 5.1.



(a) Example where the ChePAN clearly outperforms others.

(b) Example where all methods behave similarly.

Figure 5: Plot with performance in terms of calibration. The table contains the mean and standard deviation of all the folds using the mean absolute error between the empirical predicted calibration and the perfect ideal calibration of 980 equidistant quantiles.

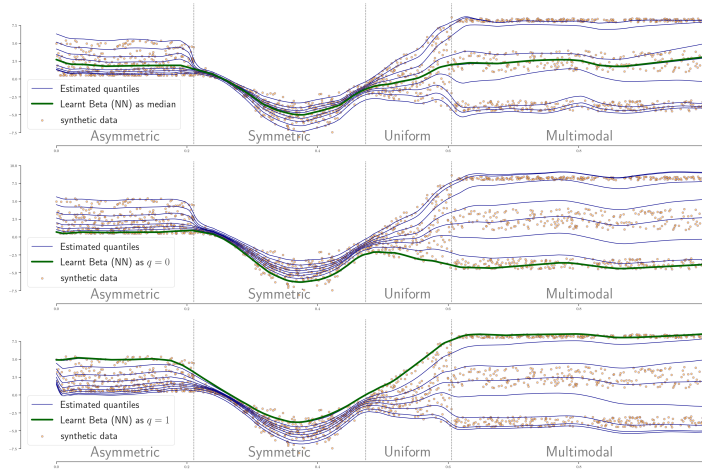


Figure 6: Heterogeneous synthetic distribution proposed by (Brando et al. (2019)). In all cases,  $\phi$  and  $\beta$  are learnt but  $\beta$  correspond to a different statistic in each case. In the upper part of the figure,  $\beta$  approximates the median, following Equation 11. In the central figure,  $\beta$  regresses to the lower quantile 0, following Equation 9. In the lower part,  $\beta$  corresponds to the higher quantile 1, following Equation 10.