Extremely Randomized Neural Networks for Constructing Prediction Intervals

Tullio Mancini[*] Hector Calvo-Pardo[†] Jose Olmo[‡]

July 19, 2021

Abstract

The aim of this paper is to propose a novel prediction model based on an ensemble of deep neural networks adapting the extremely randomized trees method originally developed for random forests. The extra-randomness introduced in the ensemble reduces the variance of the predictions and improves out-of-sample accuracy. As a byproduct, we are able to compute the uncertainty about our model predictions and construct interval forecasts. Some of the limitations associated with bootstrap-based algorithms can be overcome by not performing data resampling and thus, by ensuring the suitability of the methodology in low and mid-dimensional settings, or when the *i.i.d.* assumption does not hold. An extensive Monte Carlo simulation exercise shows the good performance of this novel prediction method in terms of mean square prediction error and the accuracy of the prediction intervals in terms of out-of-sample prediction interval coverage probabilities. The advanced approach delivers better out-of-sample accuracy in experimental settings, improving upon state-of-the-art methods like MC dropout and bootstrap procedures.

Keywords: Neural networks, ensemble methods, prediction interval, uncertainty quantification, dropout

[*]University of Southampton. Corresponding address: Department of Economics, University of Southampton. Highfield Campus, SO17 1BJ, Southampton. E-mail: T.Mancini@soton.ac.uk. Tullio Mancini acknowledges financial support from The University of Southampton Presidential Scholarship.

[†]University of Southampton, Centre for Economic and Policy Research (CEPR), ESRC Centre for Population Change (CPC), and Institut Louis Bachelier (ILB). Hector Calvo-Pardo acknowledges financial support from the ESRC funding ES/R009139/1.

[‡]University of Southampton and Universidad de Zaragoza. Jose Olmo acknowledges financial support from Fundación Aragonesa para la Investigación y el Desarrollo (ARAID).

1 Introduction

A popular and fruitful strategy in the prediction literature is model averaging. Steel (2020) distinguishes two main categories: Bayesian model averaging (see Leamer, 1978), where the model index is treated as unknown (and thus a prior is specified on both model and model parameters); and frequentist model averaging (see Wang et al., 2009; Dormann et al., 2018), where the predictions of a battery of different prediction models are ensembled. There is a well established theoretical and empirical literature analyzing the predictive advantages of both Bayesian and frequentist approaches.

When focusing on Bayesian model averaging (BMA), Min and Zellner (1993) show that the expected squared errors are minimized by Bayesian ensembles as long as the model underlying the data generating process is included in the model space. Raftery et al. (1997) introduce BMA for linear regression models. These authors show how Bayesian model averaging improves over single model predictions. [1] One of the main advantages of Bayesian ensembling is the possibility of integrating the prior structure analytically. Thus, the literature on BMA models has studied the suitability of prior density functions extensively. Reviews of the most popular options, both in linear and generalized linear frameworks, can be found in Fernandez et al. (2001), Li and Clyde (2018), and Steel (2020). Nonetheless, a vast model space may constitute a computational challenge due to the impossibility of a complete model enumeration (this problem is often solved with Markov chain Monte Carlo (MCMC) methods).

Steel (2020) also highlights that MCMC methods are not implementable for frequentist model averaging approaches (no estimation of the model probabilities), ultimately limiting the possibility of applying frequentist approaches to a large number of models. As a result, the literature focusing on frequentist ensembles tries to propose methods directed to reducing the model space (see, for example, Claeskens et al., 2006; Zhang et al., 2016; and Zhang et al., 2013). Stock and Watson (2004) provide a seminal contribution within the frequentist approach. After examining different weighting schemes found in the literature, these authors show how ”optimally” estimated weights perform worse than equal weights in terms of out-of-sample mean squared error. Smith and Wallis (2009) explain that this ”forecast combination puzzle” can be explained starting from the double estimation uncertainty associated with estimating the ”optimal” weights.

Similarly, model averaging in machine learning (e.g., boosting, bagging, random forest, and extremely randomized trees) aims at constructing a predictive model by combining ”weak” learners to obtain a strong learner. As opposed to the aforementioned literature, model averaging in

[1]Rafter et al. (1997) offer two alternative approached for model selection in a Bayesian setting. First, the "Occams window" procedure, which indicates a small set of models over which a model average can be computed. Second, the authors describe a Markov chain Monte Carlo approach that directly approximates the exact solution.

machine learning does not allow the estimation of the uncertainty on the parameter estimates and the identification of the model structure. Instead, it is mainly focused on point prediction/-forecasting and associated uncertainty estimation.

Neural networks are increasingly popular prediction models in the machine learning literature. These models are widely used in prediction tasks due to their unrivaled performance and flexibility in modeling complex unknown functions of the data. A plethora of literature has been focusing on ensembling techniques for neural networks[2]. As also explained in Krogh and Vedelsby (1995), an ensemble of neural networks ensures gains in prediction accuracy when there exists disagreement (or what they call *ambiguity*) among the single learners. Implementing strategies that lead to diversity in the error distributions across single learners ensures improvement in the out-of-sample performance of the ensemble. As an example, when a frequentist approach is used, it is possible to introduce diversity in the ensemble by training individual neural networks on resampled versions of the datasets. When instead Bayesian model averaging is implemented, independence can be achieved by creating a probability distribution of all feasible models from which one can sample using MCMC methods (Neal, 2012).

The literature has reported numerous justifications for the success of ensembles of neural networks characterized by a sufficient degree of *ambiguity*. Lee et al. (2015) summarize them as follows: (i) Ensembling ensures enlarging the hypotheses space considered by the single learner; (ii) It ensures reduction in the estimation and optimization errors that may arise from variations due to non-convex loss functions, random weight initialization, or stochastic learning; and (iii) From a Bayesian perspective, ensembling ensures a finite sample approximation of the model space.

Although neural networks provide accurate predictions, the development of tools to estimate the uncertainty around their predictions is still in its infancy. As explained in Hüllermeier and Waegeman (2020) and Pearce et al. (2018), out-of-sample pointwise accuracy is not enough[3]. The predictions of deep neural network (DNN) models need to be supported by measures of uncertainty in order to provide satisfactory answers for prediction in high-dimensional regression models, pattern recognition, biomedical diagnosis, and others (see Schmidhuber (2015) and LeCun et al. (2015) for overviews of the topic).

The present paper focuses on a machine learning approach for model prediction. We propose an ensemble of neural network models with the aim of improving the accuracy of existing model predictions from individual neural networks. A second main contribution of the present study

[2]The interested reader is referred to Lee et al. (2015), and Krogh and Vedelsby (1995) for a detailed review.

[3]A trustworthy representation of uncertainty can be considered pivotal when machine learning techniques are applied to medicine (Yang et al., 2009; Lambrou et al., 2011), or to anomaly detection, optimal resource allocation and budget planning (Zhu and Laptev, 2017), or cyber-physical systems (Varshney and Alemzadeh, 2017) defined as surgical robots, self-driving cars and the smart grid.

is to assess the uncertainty about the predictions of these ensembles of neural network models and construct interval forecasts. Our novel approach extends the Extra-trees algorithm (Geurts et al., 2006) to ensembles of deep neural networks using a fixed Bernoulli mask. To do this, we estimate T different subnetworks with randomized architectures (each network will have different layer-specific widths) that are independently trained on the same dataset. Thus, the fixed Bernoulli mask introduces an additional randomization scheme to the prediction obtained from the ensemble of neural networks that ensures independence between the components of the ensemble, reducing the variance associated with the prediction and delivering accurate prediction intervals. Additionally, based on the findings of Lee et al. (2015) and Lakshminarayanan et al. (2017), the novel procedure is expected to outperform bootstrap based approaches in terms not only of estimation accuracy but also of uncertainty estimation. This is confirmed in our simulation experiments.

The competitors of our ensemble prediction model are found in the machine learning literature. In particular, we consider Monte Carlo dropout and bootstrap procedures as the benchmark models to beat in out-of-sample prediction exercises. Monte Carlo dropout approximates the predictive distribution of a target variable by fitting a deep or shallow network with dropout both at train and test time. Conversely, both extra-neural network and bootstrap based approaches approximate the target predictive distribution via ensemble methods. When comparing classical bootstrap approaches to the extra-neural network approach proposed in this paper, we notice that (i) both methods guarantee conditional randomness of the predicted outputs, the extra-neural network method does it through the Bernoulli random variables with probability p and random weight initialization, whereas the bootstrap does it through the nonparametric data resampling and random weight initialization; (ii) by performing data resampling, the naive (nonparametric) bootstrap approach requires the assumption that observations are independent and identically distributed ($i.i.d$). Importantly, each single model is trained with only 63% unique observations of the original sample due to resampling with replacement; (iii) by randomizing the neural network structures, the extra-neural network approach increases the diversity among the individual learners (see Zhou (2012) for an analysis of diversity and ensemble methods); and (iv) the extra-neural network will benefit from the generalization gains associated with dropout (one can think of the dropout approach of Srivastava et al. (2014) as an ensemble of subnetworks trained for one gradient step).

To analyze the out-of-sample performance and the prediction interval coverage probability (PICP) of the proposed methodologies, we carry out an extensive Monte Carlo exercise that evaluates the Monte Carlo dropout, the bootstrap approach, and extra-neural network for both deep and shallow neural networks, given different dropout rates and data generating processes. The simulation results show that all three procedures return prediction intervals approximately equal to the theoretical ones for nominal values equal to 0.01 and 0.05; for prediction intervals constructed at 0.10 significance level, the extra-neural network is shown to outperform both Monte Carlo dropout and bootstrap. Additionally, the simulation findings show that the extra-neural network approach returns prediction intervals with correct empirical PICP for different dropout rates (within a reasonable range) as opposed to MC dropout, that only returns correct prediction intervals for specific values of the dropout rate. These findings show the robustness of the extra-neural network to the choice of the dropout rate, and complete the results of Levasseur et al. (2017) by showing that Monte Carlo dropout returns correct prediction intervals when the dropout rate that yields the highest out-of-sample accuracy is adopted.

The novel methodology is also evaluated on real world datasets. In order to allow for comparability with other approaches found in the literature, the experimental settings of Hernández-Lobato and Adams (2015) are adopted. The empirical results show that extra-neural network methods outperform other state-of-the-art approaches used in the literature. These results complete the conclusions drawn from the Monte Carlo simulation by showing the generalization of the extra-neural network methodology when applied to datasets of different dimensions.

The rest of the paper is organized as follows: Section 2 introduces the ensemble prediction model in a deep neural network and discusses the different sources of uncertainty. Section 3 reviews extant methodologies to construct prediction intervals that can be applied to DNNs. Section 4 introduces a novel methodology to construct prediction intervals based on an adaptation of Extra-trees for random forests. Section 5 presents the simulation setup including linear and nonlinear models along with the choice of parameters and hyperparameters for the implementation of neural network methods. Section 6 discusses the results of the empirical study. Section 7 concludes. Appendix A provides the detailed derivations of the prediction intervals obtained using the delta method (Hwang and Ding, 1997), the naive bootstrap approach (Heskes, 1997), and the Monte Carlo dropout (Gal and Ghahramani, 2016a). Appendix B contains a brief note discussing random weight initialization and uncertainty for extra-neural networks.

2 Ensemble predictors for DNN models

We propose the following additive model for predicting the output variable y_i, for $i = 1, \ldots, n$:

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \tag{1}$$

with $f(\mathbf{x}_i)$ a real-valued function used to predict the outcome variable using a set of covariates $\mathbf{x}_i$. The choice of the functional form $f(\mathbf{x}_i)$ depends on the loss function penalizing the difference between the outcome variable and the prediction. For example, it is well known that if the loss function is quadratic then the best predictive model is $f(\mathbf{x}_i) = E[y_i \mid \mathbf{x}_i]$. The error term

ϵ defines the noise in the output variable that cannot be explained by the covariates $\mathbf{x}$ and satisfies the conditional independence assumption $E[\epsilon_i \mid \mathbf{x}_i] = 0$.

In this paper, we consider $f(\mathbf{x}_i)$ to be modeled by a deep neural network. As explained in Farrell et al. (2021), when focusing on deep neural networks, the choice of the specific network class to be adopted becomes a crucial aspect. Different classes will be characterized by different complexities and thus, by different approximating power. In the present paper we focus on feedforward neural networks. More precisely, in the sub-category of fully connected (between consecutive layers) feedforward neural networks. By considering the broader class of fully connected sequential feedforward neural networks, it is possible to extend the results of the present paper to other special cases (e.g., convolutional neural networks are a subclass obtained by introducing a convolutional layer, as explained in LeCun et al., 2015) found in the literature. For any two natural numbers d, $n_1 \in \mathbb{N}$, which are called input and output dimension respectively, a $\mathbb{R}^d \to \mathbb{R}^{n_1}$ DNN is given by specifying a natural number $N \in \mathbb{N}$, a sequence of N natural numbers $Z_1, Z_2, \ldots, Z_N$, and a set of $N+1$ affine transformations $\mathbf{T}_1 : \mathbb{R}^d \to \mathbb{R}^{Z_1}, \mathbf{T}_i : \mathbb{R}^{Z_{i-1}} \to \mathbb{R}^{Z_i}$, for $i = 2, \ldots, N$, and $\mathbf{T}_{N+1} : \mathbb{R}^{Z_N} \to \mathbb{R}^{n_1}$. Such a DNN is called a $(N+1)$-layer DNN, and is said to have N hidden layers. The function $f : \mathbb{R}^d \to \mathbb{R}^{n_1}$ is the output of this DNN that is constructed as

$$f(\mathbf{x}_i; \boldsymbol{\omega}) = \mathbf{T}_{N+1} \circ \boldsymbol{\theta} \circ \mathbf{T}_N \circ \ldots \circ \mathbf{T}_2 \circ \boldsymbol{\theta} \circ \mathbf{T}_1, \tag{2}$$

with $\circ$ indicating function composition; $\mathbf{T}_n = \mathbf{W}^n \mathbf{h}_{n-1} + \mathbf{b}_n$, where - for $N = 1$ - $\mathbf{W}^n \in \mathbb{R}^{Z_1 \times d}$; $\mathbf{h}_0 \equiv \mathbf{x}$, with $\mathbf{x} \in \mathbb{R}^{d \times 1}$ the input layer, and $\mathbf{b}_n \in \mathbb{R}^{Z_1}$ is an intercept or bias vector. For $N \neq 1$, $\mathbf{W}^n \in \mathbb{R}^{Z_n \times Z_{n-1}}$ is a matrix with the deterministic weights determining the transmission of information across layers; and $\mathbf{h}_{n-1} \in \mathbb{R}^{Z_{n-1}}$ is a vector defined as $\mathbf{h}_{n-1} = \boldsymbol{\theta}(\mathbf{T}_{n-1})$ where $\boldsymbol{\theta}$ is the activation function. Typical choices for $\boldsymbol{\theta}$ are: (i) the Rectified linear unit (ReLu), $\boldsymbol{\theta}(\mathbf{T}_{n-1}) = \max\{0, \mathbf{T}_{n-1}\}$; (ii) leaky ReLu, $\boldsymbol{\theta}(\mathbf{T}_{n-1}) = \max\{\lambda \mathbf{T}_{n-1}, \mathbf{T}_{n-1}\}$ with λ defining how much the function "leaks"; (iii) Sigmoid, $\boldsymbol{\theta}(\mathbf{T}_{n-1}) = (1 + e^{-\mathbf{T}_{n-1}})^{-1}$; or Hyperbolic tangent, $\boldsymbol{\theta}(\mathbf{T}_{n-1}) = (e^{2\mathbf{T}_{n-1}} - 1)/(e^{2\mathbf{T}_{n-1}} + 1)$. Finally, $\boldsymbol{\omega} = (\boldsymbol{W}^n, \mathbf{b}_n)$ collects the set of estimable features of the model. The *depth* of a DNN is defined as $N+1$. The width of the n^{th} hidden layer is Z_n and the *width* of a DNN is defined by $\max\{Z_1, \ldots, Z_N\}$. The *size* of the DNN is $Z_{\text{tot}} = Z_1 + Z_2 + \ldots + Z_N$. The number of active weights (different from zero) - in a fully connected DNN - of the n^{th} hidden layer is $w_n = (Z_n \times Z_{n-1}) + Z_n$. The *number of active weights* in a fully connected DNN is $w_1 + w_2 + \ldots + w_N$.

To reduce the tendency to overfitting when training DNNs, a regularization technique is usually adopted. Among many, training with dropout (Srivastava et al., 2014) is a popular approach that ensures good generalization performance. The present section discusses training with dropout in detail because it is a powerful regularization technique and is closely related to

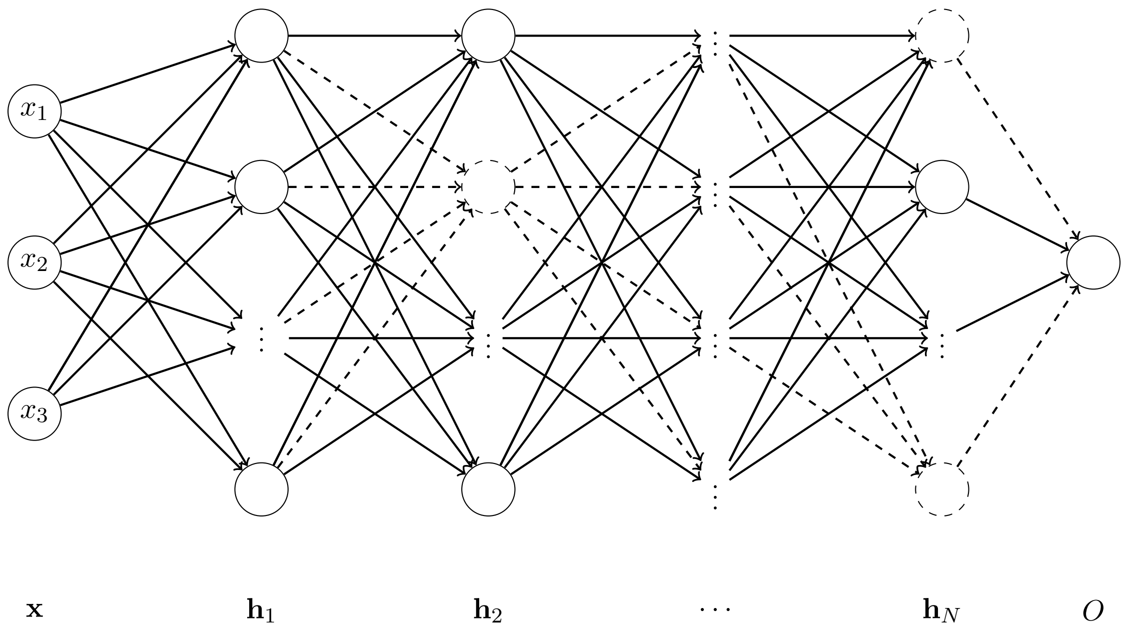

Figure 1: ReLu Deep Neural Network with bias terms 0 and dropout mask.

model averaging and uncertainty estimation in neural network settings.

Training with dropout (*dropout training* - Figure 1) implies that for each iteration of the learning algorithm different random subnetworks (or *thinned* networks) will be trained.[4] Let h_{zn} denote the elements of the vector $\mathbf{h}_n$ for a given node $z = 1, \ldots, Z_n$ and layer $n = 1, \ldots, N$. Srivastava et al. (2014) develop a dropout methodology that is applied to each function h_{zn} to obtain a transformed variable $\overline{h}_{zn}$. This variable is obtained by pre-multiplying h_{zn} by a random variable r_{zn} with distribution function $F(r_{zn})$, such that $\overline{h}_{zn} = r_{zn} \cdot h_{zn}$, for all (z, n), prior to being fed forward to the activation function of the next layer, h_{zn+1}, for all $z = 1, \ldots, Z_{n+1}$. For any layer n, $\mathbf{r}_n$ is then a vector of independent random variables, $\mathbf{r}_n = [r_{1n}, \ldots, r_{Z_n n}] \in \mathbb{R}^{Z_n}$. In this paper we consider only the Bernoulli probability distribution $F(r_{zn})$, where each r_{zn} has probability p of being 1 (and $q = 1-p$ of being 0). The vector $\mathbf{r}_n$ is then sampled and multiplied element-wise with the outputs of that layer, h_{zn}, to create the thinned outputs, $\overline{h}_{zn}$, which are then used as input to the next layer, h_{zn+1}.

When this process is applied at each layer $n = 1, \ldots, N$, it amounts to sampling a subnetwork from a larger network at each forward pass (or gradient step). At test time, the weights are scaled down as $\overline{\mathbf{W}}^n = p\mathbf{W}^n, n = 1, \ldots, N$, returning a deterministic output. We then identify $\mathbf{r}^\star = [\mathbf{r}_1, \ldots, \mathbf{r}_N]$ as the collection of independent random variables applied to a feedforward neural network of depth $N + 1$[5]. Figure 1 shows how the dropout mask works; at each training

[4]Warde-Farley et al. (2014) explain how each subnetwork is usually trained for only one gradient step.

[5]In practice, an inverted dropout methodology is applied when implementing this methodology in Keras for RStudio. In this case, instead of scaling-down the weights at test time, the weights are scaled-up during train time as $\overline{\mathbf{W}}^n = (1/p)\mathbf{W}^n, n = 1, \ldots, N$. At test time, a single deterministic forward pass on the unscaled

step (forward and backward pass), every neuron of each hidden layer will randomly not be considered when training the network and thus be "dropped out" (Géron, 2019). It is now possible to understand why dropout training is also closely related to model ensembling. In particular, Goodfellow et al. (2016) explain how dropout involves training an ensemble of subnetworks –obtained by removing units from a "parent" neural network, at random– which are trained for only one gradient step. As opposed to traditional ensembling techniques, the individual models are not independent due to parameter sharing. Finally, as explained in Srivastava et al. (2014), scaling down the weights of the "parent" neural network at test time is equivalent to performing an approximation of model averaging across the different random subnetworks.

In the remainder of the paper, we will consider ReLu DNNs for comparison with the relevant literature (e.g., Hernández-Lobato and Adams (2015) that consider ReLu Bayesian neural networks; and Farrell et al. (2021) that establish valid causal inference on finite-dimensional parameters following a first-step estimation using deep ReLu neural networks). Under these premises, universal approximation theorems developed for ReLu DNN models (Lu et al., 2017) guarantee that $f(\mathbf{x}_i;\boldsymbol{\omega})$ approximates the true function $f(\mathbf{x}_i)$ in (1) arbitrarily well. See also Cybenko (1989), Leshno et al. (1993), Hornik (1991), Lu et al. (2017), and Mei et al. (2018) for universal approximation theorems in similar contexts.

We should note the presence of an approximation error due to replacing $f(\mathbf{x}_i)$ by $f(\mathbf{x}_i;\boldsymbol{\omega})$ in model (1), where $f(\mathbf{x}_i;\boldsymbol{\omega})$ denotes a feasible version of the DNN model that can be estimated from the data. The model that we consider in practice is

$$y_i = f(\mathbf{x}_i;\boldsymbol{\omega}) + u_i, \tag{3}$$

where $u_i = \epsilon_i + f(\mathbf{x}_i) - f(\mathbf{x}_i;\boldsymbol{\omega})$. In the related literature the effect of the approximation error is usually neglected, see Pearce et al. (2018) and Heskes (1997). In practice, we estimate model (3) using a training sample to obtain parameter estimates $\widehat{\boldsymbol{\omega}}$, such that the relevant empirical model is

$$y_i = f(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}) + e_i, \tag{4}$$

with $f(\mathbf{x}_i;\widehat{\boldsymbol{\omega}})$ a function that is estimated from the data and $\widehat{\boldsymbol{\omega}}$ the parameter estimates of the matrices of weights $\mathbf{W^n}$ and bias parameters $\mathbf{b_n}$ defining the DNN; e_i is the residual of the model. Combining expressions (1) to (4), the error term in (4) can be decomposed as

$$e_i = \underbrace{f(\mathbf{x}_i;\boldsymbol{\omega}) - f(\mathbf{x}_i;\widehat{\boldsymbol{\omega}})}_{\text{estimation error}} + \underbrace{f(\mathbf{x}_i) - f(\mathbf{x}_i;\boldsymbol{\omega})}_{\text{bias effect}} + \underbrace{\epsilon_i}_{\text{aleatoric error}} \tag{5}$$

such that the conditional variance of the predicted output $f(\mathbf{x}_i;\widehat{\boldsymbol{\omega}})$ given the set of covariates $\mathbf{x}$ satisfies $\sigma_e^2 = \sigma_{\widehat{\boldsymbol{\omega}}}^2(\mathbf{x}_i) + \sigma_\epsilon^2$, with $\sigma_{\widehat{\boldsymbol{\omega}}}^2(\mathbf{x}_i)$ the epistemic uncertainty due to the estimation of

weights $\mathbf{W}^n$ is performed.

the model parameters and hyperparameters (estimation effect) and σ_ϵ^2 the variance due to the aleatoric error. The bias term does not have an effect on the variance of the predictor but introduces an error in the model forecast. More formally, $\mu_i = f(\mathbf{x}_i) - f(\mathbf{x}_i; \boldsymbol{\omega})$ is a constant that captures the approximation error (bias) due to the deployed ReLu DNN being a universal approximator asymptotically. In this paper we concentrate on estimating the uncertainty around the predictions, given by σ_e^2, however, when possible, we will also discuss the bias effect due to the approximation of the ReLu DNN model.

The distinction between epistemic and aleatoric uncertainty is extremely relevant when DNNs are considered. It has been shown that deep models, notwithstanding the high confidence in their predictions, fail on specific instances due to parameter uncertainty (see Hüllermeier and Waegeman, 2020). Additionally, deep learning models are subject to drastic changes in their performance when minor changes to the dataset are engineered (well known problem of *adversarial examples* in Papernot et al., 2017) implying variability in the parameter estimates. For this reason, the literature focusing on deep learning and uncertainty quantification proposes algorithms that allow capturing all sources of uncertainty (see Zhu and Laptev, 2017; Hüllermeier and Waegeman, 2020; Senge et al., 2014; Kull and Flach, 2014; and Varshney and Alemzadeh, 2017).

Our main objective is to study the performance of ensembles of deep neural network models and to propose a novel approach that is inspired by the extremely randomized trees method originally developed for random forests. An ensemble of predictors in our context is then given by:

$$\bar{y}(\mathbf{x}_i) = \frac{1}{T} \sum_{t=1}^{T} f_t(\mathbf{x}_i; \widehat{\boldsymbol{\omega}}^{(t)}), \text{ for } i = 1, \ldots, n, \tag{6}$$

where $f_t(\mathbf{x}_i; \widehat{\boldsymbol{\omega}}^{(t)})$ denotes a set of T different prediction models based on deep neural network models, and $\widehat{\boldsymbol{\omega}}^{(t)}$ denotes the estimates of the DNN model parameters. Second, we construct interval forecasts for the ensemble model predictions.

3 Prediction intervals for DNN models

The current study is related to recent literature on prediction intervals for neural networks. A pioneering contribution is provided by Hwang and Ding (1997) that construct asymptotically valid prediction intervals for neural networks (delta method, see subsection A.1 for a complete analysis and derivation). Despite providing asymptotically valid prediction intervals, the delta method is not widely adopted by the literature focusing on uncertainty quantification and deep learning due to problems associated with the computation of the Jacobian matrix ($\mathbf{J}$). In particular, due to the high number of parameters in $\boldsymbol{\omega}$, the complex calculation of $\mathbf{J}$ is prone to error (Tibshirani, 1996); the near singularities in the model due to overfitting (Tibshirani,

1996) or due to the small sample size (De vieaux et al., 1998) make the computation of the gradient **J** unreliable or unfeasible. Additionally, being their research focused only on single layer feedforward neural networks with sigmoidal activation function, it does not find applicability in some widely adopted neural network structures (e.g. convolutional neural networks, recurrent neural networks, and deep feedforward neural networks).

An alternative approach to asymptotic prediction intervals is to construct a finite-sample approximation of the prediction interval. Bootstrap procedures have become increasingly popular, despite their computational requirements, as they provide a reliable solution to obtain the predictive distribution of the output variable in both shallow and deep neural networks. As also highlighted by Tibshirani (1996), bootstrapping prediction intervals provide a feasible alternative that does not suffer from the matrix inversion problem and does not depend on the existence of derivatives (see subsection A.2 for a complete analysis and derivation). The literature has developed many different forms of bootstrapping methods. One of its simplest and most popular forms is the percentile or naive bootstrap proposed by Efron (1979). Under this method, observations are drawn from an independent and identically distributed sample with replacement, and each observation has the same probability of being extracted (see, for example, Carney et al., 1999; and Errouissi et al., 2015 for implementation of the naive bootstrap methodology for uncertainty estimation). However, recent advances in the neural network literature (Pearce et al., 2018; Lee et al., 2015; and Lakshminarayanan et al., 2017) have also shown how resampling with replacement has a negative impact not only on the prediction accuracy but also on the correct quantification of the predictive uncertainty of the ensemble itself. Additionally, El Karoui and Purdom (2018) show how naive bootstrapping becomes completely unreliable in estimating confidence intervals in low and mid-dimensional settings.

Thus, the literature focusing on uncertainty estimation has proposed alternatives to the bootstrap approach that allow estimating the uncertainty around DNNs predictions without performing data resampling. Levasseur et al. (2017) notice that one of the main obstacles for assessing uncertainty around the outputs of neural network models is the fact that the weights characterizing the predictions are usually fixed, implying that the output is deterministic. In contrast, Bayesian neural networks (Denker and LeCun, 1991) - instead of defining deterministic weights - allow the network weights to be defined by a given probability distribution and can capture the posterior distribution of the output, providing a probabilistic measure of uncertainty around the model predictions. Being the approximation of the posterior distribution a difficult task, the literature focusing on deep Bayesian neural networks has proposed different alternatives for the estimation of such distribution. These alternatives center around the Bayesian interpretation of dropout methods to estimate the uncertainty in the model predictions. A noteworthy example is Gal and Ghahramani (2016a); these authors develop a Monte Carlo (MC) dropout to model both parameter and data uncertainty by fitting a deep neural network with dropout implemented not only at training but also during test phase. During test time, each forward pass is multiplied by a random variable to generate a random sample of the approximated posterior distribution. Levasseur et al. (2017) analyze the PICP of the procedure proposed by Gal and Ghahramani (2016a) and conclude that the construction of prediction intervals with correct empirical PICP is highly dependent on the adequate tuning of the dropout rate.

Applying dropout during the test phase can also be regarded as an approach to estimate the uncertainty around the predicted outputs from deep neural networks that works outside the Bayesian framework (see subsection A.3 for a complete analysis and derivation). As one could notice, using dropout also at test phase allows randomizing the output of the DNN at each forward pass and thus, by performing T stochastic forward passes, it is possible to obtain the sample $\{\widehat{y}(\mathbf{x}_i; \widehat{\boldsymbol{\omega}}^{(t)})\}_{t=1}^{T}$. Recent literature focusing on the approximation of the predictive distribution of DNNs has proposed several algorithms - based on the MC dropout of Gal and Ghahramani (2016a) - for the estimation of the predictive uncertainty in deep learning outside a Bayesian framework. As an example, Serpell et al. (2019) augment the MC dropout by implementing the Mean Variance Estimation (MVE)[6] and stochastic forward passes. If the MVE approach allows modeling the data uncertainty - accommodating a varying e, the Monte Carlo dropout captures the uncertainty in the model parameters. The two procedures together allow the correct estimation of σ_e^2. Zhu and Laptev (2017) improve over the original Monte Carlo dropout by estimating the noise level using the residual sum of squares evaluated on a hold-out set[7] - Equation A.14; Kendall and Gal (2017) propose a new loss function that allows estimating the aleatoric uncertainty from the input data. Finally, Lakshminarayanan et al. (2017) explain how tuning the dropout rate on the training data implies interpreting dropout as a tool for Bayesian inference (any Bayesian posterior should be approximated starting only from the training data).

However, as also highlighted by the simulation results of Levasseur et al. (2017), the present paper shows how the estimation of the σ_e^2 depends on the choice of p. As the epistemic uncertainty in the MC dropout is determined solely by the choice of p, if p is set equal to 1, the epistemic uncertainty will be zero, returning narrower prediction intervals. We improve over the results by Levasseur et al. (2017) as we show that choosing the dropout rate that minimizes the out-of-sample prediction error ensures returning prediction intervals with empirical PICPs approximately equal to their nominal levels.

[6]The Mean Variance Estimation method - introduced by Nix and Weigend (1994) - involves fitting a neural network with two output nodes capturing the mean and the variance, respectively, of a Normal distribution.

[7]These authors precise that the approach of Gal and Ghahramani (2016a) relies on the implausible assumption of knowing the correct noise level a priori.

4 Extra-neural networks (Fixed Bernoulli Mask)

In this section we introduce a novel ensemble predictor within deep neural network models. This methodology also allows us to construct prediction intervals based upon the work of Srivastava et al. (2014) adopting the original concept of an ensemble of subnetworks, from which the dropout training is built upon. The Bernoulli mask $\mathbf{r}^{\star}$ introduces an additional randomization scheme to the predictions obtained from the ensemble of neural networks that ensures independence of the individual predictor models.

For notation purposes, we will identify the fixed Bernoulli mask as $\bar{\mathbf{r}}^{\star}$ as opposed to $\mathbf{r}^{\star}$ used in dropout training. In other words, T sets of vectors $\{\bar{\mathbf{r}}^{\star(t)}\}_{t=1}^{T}$ are sampled from the Bernoulli distribution prior to training (instead of at test time with Monte Carlo dropout) and they are kept constant during both train and test phases. This approach reduces to train and independently fit T random subnetworks on the same dataset. In this setting, generating the predictive distribution is similar, in spirit, to an ensemble approach that trains different sub-neural networks on the same dataset. The proposed algorithm - being based on the extremely randomized trees proposed by Geurts et al. (2006) - is called *extra-neural networks.*

Let $\bar{f}_{EN}(\mathbf{x}_i)$ denote the ensemble predictor obtained from the extra-neural networks approach that is constructed as

$$\bar{f}_{EN}(\mathbf{x}_i) = \frac{1}{T}\sum_{t=1}^{T} f_t(\mathbf{x}_i; \widehat{\boldsymbol{\omega}}^{(t)}), \text{ for } i = 1, \dots, n. \tag{7}$$

We consider T fitted subnetworks defined as $f_t(\mathbf{x}_i; \widehat{\boldsymbol{\omega}}^{(t)})$ with $t = 1, \dots, T$. We use f_t to note that each prediction belongs to a potentially different neural network model; $\widehat{\boldsymbol{\omega}}^{(t)}$ denotes the parameter estimates obtained from fitting each subnetwork independently.

Before analyzing the prediction intervals for the extra-neural network, it is convenient to analyze the factors that influence the prediction accuracy of the model. To do this, we compute the mean square prediction error (MSPE) of the prediction conditional on the input vector $\mathbf{x}_i$. Then,

$$MSPE(\bar{f}_{EN}(\mathbf{x}_i)) \equiv \mathbb{E}[(\bar{f}_{EN}(\mathbf{x}_i) - y_i)^2] = \text{Bias}^2(\bar{f}_{EN}(\mathbf{x}_i)) + V(\bar{f}_{EN}(\mathbf{x}_i)). \tag{8}$$

We compute the conditional bias and variance of $\bar{f}_{EN}(\mathbf{x}_i)$ as

$$\text{Bias}(\bar{f}_{EN}(\mathbf{x}_i)) \equiv \mathbb{E}[\bar{f}_{EN}(\mathbf{x}_i) - y_i] = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[f_t(\mathbf{x}_i; \widehat{\boldsymbol{\omega}}^{(t)})] - f(\mathbf{x}_i), \tag{9}$$

and

$$V(\bar{f}_{EN}(\mathbf{x}_i)) \equiv \mathbb{E}\left(\bar{f}_{EN}(\mathbf{x}_i) - \mathbb{E}[\bar{f}_{EN}(\mathbf{x}_i)]\right)^2 = \mathbb{E}\left[\bar{f}^2_{EN}(\mathbf{x}_i)\right] - \mathbb{E}^2[\bar{f}_{EN}(\mathbf{x}_i)],$$

such that

$$V(\bar{f}_{EN}(\mathbf{x}_i)) = \frac{1}{T^2}\sum_{t=1}^{T}\sum_{t'=1}^{T}\left(\mathbb{E}[f_t(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}^{(t)})f_{t'}(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}^{(t')})] - \mathbb{E}[f_t(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}^{(t)})]\mathbb{E}[f_{t'}(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}^{(t')})]\right).$$

Furthermore, assuming that the first two statistical moments of all the individual predictors indexed by $t = 1, \dots, T$ are equal, with $\mathbb{E}\left[f_t(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}^{(t)})\right] = f(\mathbf{x}_i) + \mu_i$, where μ_i is the bias term, $\mathbb{V}\left[f_t(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}^{(t)})\right] = \sigma^2_{\widehat{\boldsymbol{\omega}}}(\mathbf{x}_i)$, and $\mathrm{Cov}\left[f_t(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}^{(t)})f_{t'}(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}^{(t')})\right] = c_i$, we obtain

$$MSPE(\bar{f}_{EN}(\mathbf{x}_i)) = \mu_i^2 + \frac{1}{T}\sigma^2_{\widehat{\boldsymbol{\omega}}}(\mathbf{x}_i) + \frac{T-1}{T}c_i. \tag{10}$$

This expression extends Zhou (2012) by showing that the MSPE of the ensembler (7) depends on the variance of the individual ensemblers, their covariance and the approximation bias. The smaller the covariance, the smaller the generalization error of the ensemble. In contrast, if the different predictors are perfectly correlated (as for the MC dropout) we know that $c_i = \sigma^2_{\widehat{\boldsymbol{\omega}}}(\mathbf{x}_i)$ and thus $MSPE(\bar{f}_{EN}(\mathbf{x}_i)) = \sigma^2_{\widehat{\boldsymbol{\omega}}}(\mathbf{x}_i) + \mu_i^2$ - effectively reducing to zero the effect of ensembling. Similarly, the MSPE is minimized when the errors are perfectly uncorrelated and thus when $c_i = 0$.

This result has important implications when analyzing the epistemic uncertainty of an extra-neural network. If it is assumed that the correlation among the predictions from the subnetworks is equal to zero, then as $T \to \infty$, the $MSPE(\bar{f}_{EN}(\mathbf{x}_i))$ converges to zero, assuming that the approximation bias is negligible. Therefore, a suitable prediction interval is

$$\bar{f}_{EN}(\mathbf{x}_i) \pm z_{1-\alpha/2}\left(\frac{\widehat{\sigma}^2_{\widehat{\boldsymbol{\omega}}}(\mathbf{x}_i)}{T} + \widehat{\sigma}^2_{\epsilon}\right)^{1/2}, \tag{11}$$

with $\widehat{\sigma}^2_{\widehat{\boldsymbol{\omega}}}(\mathbf{x}_i) = \frac{1}{T}\sum_{t=1}^{T}(f_t(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}^{(t)}) - \bar{f}_{EN}(\mathbf{x}_i))^2$ and $\widehat{\sigma}^2_{\epsilon} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \bar{f}_{EN}(\mathbf{x}_i)\right)^2$, where n is the size of the test sample.[8]

As explained in Zhou (2012), the covariance term in equation (10) captures the diversity existing among the T different subnetworks identifying the extra-neural network. The aim of the extra-neural network approach proposed in this paper is to construct individual predictors that are mutually independent such that the prediction interval (11) is valid. The diversity in the model predictions depends on the variance of the Bernoulli masks generated by the random sample $\{\bar{\mathbf{r}}^{\star(t)}\}_{t=1}^{T}$. It is well known that the variance of a Bernoulli distribution is defined as $\varsigma^2 = p(1-p)$; therefore, it can be easily shown that the solution to $\partial\varsigma^2/\partial p = 0$ is $p = 1/2$ and that $\partial^2\varsigma^2/\partial p^2 = -2$ showing that ς^2 is maximized at $p = q = 0.5$. Therefore, one could conclude that the covariance in Equation 8 is minimized for $p = 0.5$ and maximized for $p = 0$ and $p = 1$.

[8]Note that for obtaining a consistent estimator of $\widehat{\sigma}^2_{\epsilon}$ we have imposed homoscedasticity of the error terms ϵ_i over the test sample.

However, a complete analysis of the covariance of an ensemble of neural networks must consider the relation existing between the number of hidden nodes and the particular data generating process analyzed. Based on the literature on approximation theory and DNNs, the number of hidden nodes defines the approximation power (or flexibility) of the neural networks (for a summary on the topic, see Calvo-Pardo et al., 2020). Farrell et al. (2021) - by comparing DNN structures to different nonparametric techniques for approximating unknown continuous functions - also make explicit the dependence between the number of hidden nodes in the DNN (Z_{tot}) and the approximation power. Therefore, if the size of the networks is such that the *ambiguity* –measure of disagreement among the different networks on a specific input– is too low, the assumption of $c = 0$ becomes unrealistic.

Algorithm 1 Extra-neural networks

INPUT: Training Data $\{x_i^{\llcorner} \equiv (\mathbf{x}_i, y_i)\}_{i=1}^M$
OUTPUT: Prediction Interval $\widehat{f}(\mathbf{x}; \boldsymbol{\omega})$.

```
 1: procedure T LEARNERS
 2:
 3:     Define depth and width of original neural network.
 4:     while (t < T) do
 5:         Generate a Bernoulli mask $\bar{\mathbf{r}}^\star$ prior to training.
 6:         Apply Bernoulli mask $\bar{\mathbf{r}}^\star$ to the original neural network.
 7:         Train random thinned network on $\mathbf{x}^{\llcorner}$ with random initialization of $\{\mathbf{W}_0^n\}_{n=1}^N$
 8:         Trained thinned network $\rightarrow$ Deterministic forward pass on test data.
 9:         Store $f_t(\mathbf{x}_i; \widehat{\boldsymbol{\omega}}^{(t)})$.
10:     Compute the ensemble estimate:
            $\bar{f}_{EN}(\mathbf{x}_i) = \frac{1}{T}\sum_{t=1}^{T} f_t(\mathbf{x}_i; \widehat{\boldsymbol{\omega}}^{(t)})$    (12)
11:     Compute the epistemic and aleatoric variance:
            $\begin{cases} \widehat{\sigma}^2_{\widehat{\boldsymbol{\omega}}}(\mathbf{x}_i) = \frac{1}{T}\sum_{t=1}^{T}[f_t(\mathbf{x}_i; \widehat{\boldsymbol{\omega}}^{(t)}) - \bar{f}_{EN}(\mathbf{x}_i)]^2 \\ \widehat{\sigma}^2_\epsilon = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \bar{f}_{EN}(\mathbf{x}_i)\right)^2 \end{cases}$    (13)
12:     Define Prediction Interval:
            $\bar{f}_{EN}(\mathbf{x}_i) \pm z_{1-\alpha/2}\widehat{\sigma}_e,$    (14)
    with $\widehat{\sigma}_e = \left(\frac{\widehat{\sigma}^2_{\widehat{\boldsymbol{\omega}}}(\mathbf{x}_i)}{T} + \widehat{\sigma}^2_\epsilon\right)^{1/2}$.
      return Prediction interval (14)
```

Based on the above paragraph, one could conclude that the analysis of the covariance in an extra-neural network must consider not only that p determines the variance of $\{\bar{\mathbf{r}}^{\star(t)}\}_{t=1}^T$ but also that the particular data generating process under study, as Z_{dropout} is also determined by p. As the two effects must be considered together when choosing the probability p, one must consider that p converging to 0.5 from above or from below may have a similar impact in terms of the reduction in c but an opposite effect on the dimension of Z_{dropout}. As p converges to 0.5 from below, the dimensions of the subnetworks will increase (higher probability for each neuron to be 1 and thus to be retained in the subnetwork). Conversely, p converging to 0.5 from above will ensure a reduction of the number of hidden nodes in the T subnetworks (higher probability of being "dropped out" - $q = 1 - p$).

Algorithm 1 reports the procedure to be used for implementing the extra-neural network. In order to generate $\{f_t(\mathbf{x}; \widehat{\boldsymbol{\omega}}^{(t)})\}_{t=1}^{T}$, we sample T vectors $\{\bar{\mathbf{r}}^{\star(t)}\}_{t=1}^{T}$ prior to training. Each fixed Bernoulli mask is applied independently to the original network returning T independent subnetworks of size $Z_{\text{dropout}}^{(t)} \leq Z_{\text{tot}}$. Each subnetwork is then trained independently on $\mathbf{x}^{\llcorner}$, and T deterministic forward passes are performed at test phase. Thus, even if the novel algorithm is based upon the original idea of dropout proposed by Srivastava et al. (2014) and introduces randomness by means of a random sample $\{\bar{\mathbf{r}}^{\star(t)}\}_{t=1}^{T}$, it is closer to classical ensemble methods than to training with dropout. This has important implications while training. In this case, performing weight scaling at test phase (or train phase if the algorithm is implemented in Keras) is not required as the Bernoulli mask is applied before training. Training T independent subnetworks identified by $\{\bar{\mathbf{r}}^{\star(t)}\}_{t=1}^{T}$ makes no longer necessary to ensure that the expected total input to the units of a DNN at test time is approximately the same as the expected value at training (see Goodfellow et al., 2016).

The procedure reported in Algorithm 1 shows that an extra-neural network is an ensemble of T neural networks with randomized weights (the interested reader is referred to Appendix B for a brief analysis on random weight initialization) and structures and no data resampling. Based on the results reported by Pearce et al. (2018), Lee et al. (2015) and Lakshminarayanan et al. (2017) regarding deep ensembles[9] it is expected that the extra-neural network algorithm will improve over a bootstrapping ensemble approach. More precisely, Lee et al. (2015) show how parameter resampling without bootstrap resampling - equivalent to training T different $f(\mathbf{x}_i; \widehat{\boldsymbol{\omega}}^{(t)})$ on $\mathbf{x}^{\llcorner}$ - outperforms a bootstrap approach (analyzed in subsection A.2) in terms of predictive accuracy; Lakshminarayanan et al. (2017) complement the results of Lee et al. (2015) by showing that data resampling in deep ensembles deteriorates not only the prediction accuracy but also the definition of the predictive uncertainty of the ensemble itself.

Therefore, the extra-neural networks by randomizing not only the weights of the T subnetworks but also their structure, and by fitting the networks on the entire training set $\{\mathbf{x}_i\}_{i=1}^{M}$, are expected to outperform the bootstrap approach in terms of both out-of-sample prediction

[9]Deep ensembles and ensembles of DNNs are considered synonym for the rest of the paper.

accuracy (Lee et al., 2015) and uncertainty quantification (Lakshminarayanan et al., 2017)[10].

The main drawback of the extra-neural network algorithm is associated to the computing power required (see Section 5). In particular, if the computational requirements of the proposed methodology are equivalent to existing bootstrapping procedures (with and without data resampling), they are significantly greater than the ones of the MC dropout methodology. However, due to the parameter sharing in the MC dropout, the extra-neural networks will ensure a lower MSPE (see equation (10)). Additionally, it is expected an improvement also in terms of hyperspace: the novel methodology allows reaching a good estimation performance without the pivotal fine-tuning that is required by the other procedures. As in the case of bootstrap based procedures, the independence among the different learners in the extra-neural networks allows parallel computing, ensuring savings in computational time. Last but not least, the extra-neural network improves over a bootstrap based approach in terms of applicability: if the bootstrap approach relies on the assumption of *i.i.d* observations, the extra-neural network does not.

All the results analyzed in Section 3 and 4 will be formally evaluated in an extensive simulation study focused on assessing if the reported procedures return correct prediction intervals (empirical PICP close to the nominal one) for different significance levels and data generating processes. Finally, the empirical experimental setting of Hernández-Lobato and Adams (2015) is implemented to compare the performance (in terms of RMSPE) of the different algorithms.

5 Monte Carlo simulation

The aim of this simulation section is twofold. First, we assess the accuracy of the pointwise predictions of the above ensemble prediction models, and we study the empirical PICPs associated to each prediction model. Second, using the same experimental settings, we evaluate, empirically, that the covariances between the individual learners in the extra-neural network approach are close to zero ($c_i = 0$ in expression 10).

We analyze the empirical PICPs obtained using the bootstrap approach (expression A.9), the MC dropout (expression A.15), and the extra-neural network (expression 11)[11]. For each prediction interval, the empirical PICPs ($\bar{\alpha}$) for three different significance levels (0.01, 0.05, and 0.10) are computed. This allows evaluating the correctness of the constructed prediction intervals for different significance levels. All three procedures are analyzed for increasing $T = [30, 50, 70]$, and for a sample size $M + n = 1200 + 300$. When the small-dimensional linear process is

[10]By considering deep ensembles the equivalent of a random forest (Breiman, 2001) where the single learners are neural networks and where the parameter uncertainty is captured not by the random subset selection of features at each node (trees) but by random weight initialization, the extra randomization introduced by extra-neural networks is comparable to the extremely randomized trees in Geurts et al. (2006). In this case, randomizing also the structure is equivalent to randomizing the cut-point at each node in a tree.

[11]The authors acknowledge the use of the IRIDIS High Performance Computing Facility, and associated support services at the University of Southampton, in the completion of this work.

considered - in order to evaluate the impact that different ps may have on the correct definition of the prediction intervals - we will consider $p = [0.995, 0.990, 0.950, 0.900, 0.800]$[12]. Subsection 5.1 reports the setting for the simulation of the small dimensional linear and nonlinear data generating processes; subsection 5.2 summarizes the results. Subsection 5.3 confirms empirically the absence of correlation between the different individual predictors in the extra-neural network approach using the same simulation settings as subsection 5.1.

5.1 Data Generating Processes

When the nonlinear data generating process (DGP) is considered, the dataset $\mathbf{x} \in \mathbb{R}^5$ is defined by a multivariate normal distribution. The desired pairwise correlation matrix of the series in $\mathbf{x}$ is defined as:

$$\mathbf{C} = \begin{bmatrix} 1 & 0.5 & 0.6 & 0.7 & 0.5 \\ 0.5 & 1 & 0.7 & 0.8 & 0.5 \\ 0.6 & 0.7 & 1 & 0.7 & 0.5 \\ 0.7 & 0.8 & 0.7 & 1 & 0.8 \\ 0.5 & 0.5 & 0.5 & 0.8 & 1 \end{bmatrix} \tag{15}$$

The choice of the pairwise correlations in $\mathbf{C}$ is random but such that the matrix is guaranteed to be positive definite. Given these conditions, the simulated multivariate normal distribution will have $\boldsymbol{\mu} = [-4, 2, 2, 2, 1]$[13], and $\boldsymbol{\Sigma} = \mathbf{C}$. The nonlinear DGP is defined by a ReLu DNN with two hidden layers of width 3 and 2 respectively, and bias equal to 1 across all hidden layers

$$\begin{aligned} \mathbf{T}_1 = & \underbrace{\theta(1 - 3\mathbf{x}_1 - 2\mathbf{x}_2 + 1\mathbf{x}_3 + 5\mathbf{x}_4 - 3\mathbf{x}_5)}_{\mathbf{h}_{11}} + \underbrace{\theta(1 + 4\mathbf{x}_1 + 5\mathbf{x}_2 + 2\mathbf{x}_3 + 2\mathbf{x}_4 - 5\mathbf{x}_5)}_{\mathbf{h}_{21}} \\ & + \underbrace{\theta(1 - 3\mathbf{x}_1 - 4\mathbf{x}_2 + 2\mathbf{x}_3 - 2\mathbf{x}_4 + 3\mathbf{x}_5)}_{\mathbf{h}_{31}} \end{aligned}$$

$$\mathbf{T}_2 = \underbrace{\theta(1 - 1\mathbf{h}_{11} + 3\mathbf{h}_{21} + 5\mathbf{h}_{31})}_{\mathbf{h}_{12}} + \underbrace{\theta(1 - 2\mathbf{h}_{11} + 3\mathbf{h}_{21} + 5\mathbf{h}_{31})}_{\mathbf{h}_{22}}$$

$$\mathbf{y} = 1 + \mathbf{h}_{12} + 2\mathbf{h}_{22} + \boldsymbol{\epsilon} \tag{16}$$

with $\epsilon \sim \mathcal{N}(0, 0.7)$, $\theta(\mathbf{x}) = \max\{0, \mathbf{x}\}$, and the coefficients (network weights) randomly sampled with replacement from $[-5, 5]$. The standard deviation of the error term is set equal to 0.7 in order to reduce the nuisance in the system by differentiating the stochastic behavior of the

[12]The choice of the dropout rate $q = 1 - p$ is dictated by a really small network size and also a fairly small simulated dataset.

[13]The means are randomly sampled with replacement from a domain defined in $[-5, 5]$.

regressors $\mathbf{x}$ and of the error term[14]. Figure 2 provides a visual representation of the underlying DGP and the obtained dependent variable y:

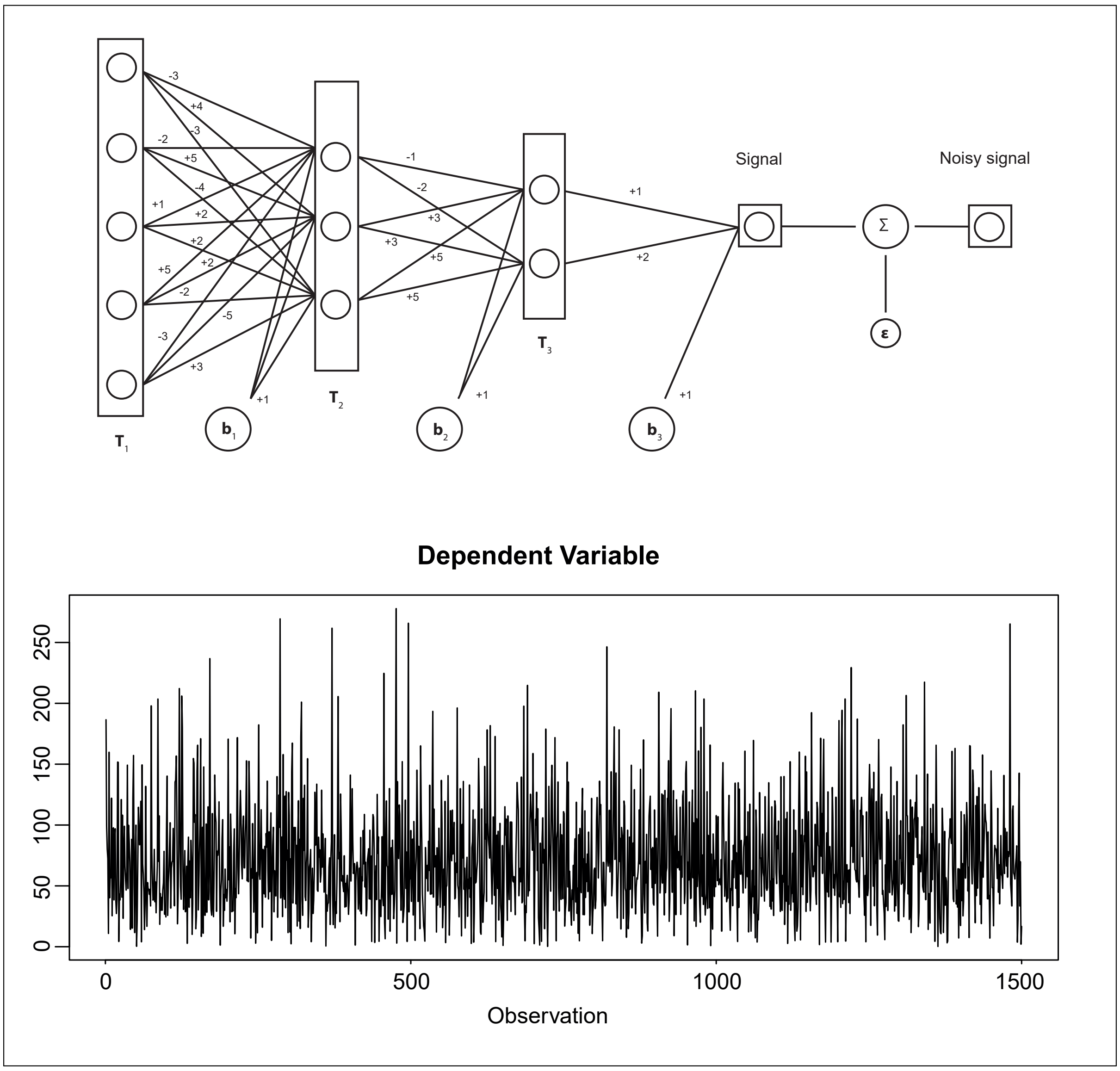

Figure 2: Data Generating Process

Next, a linear DGP that allows for interactions among the variables is also simulated. Also in this case $\mathbf{x} \in \mathbb{R}^5$, with $\mathbf{x}_1 \sim \mathcal{N}(-4,1)$, $\mathbf{x}_2 \sim \mathcal{N}(1,1)$, $\mathbf{x}_3 \sim \mathcal{N}(1,1)$, $\mathbf{x}_4 \sim \mathcal{N}(1,1)$, and $\mathbf{x}_5 \sim \mathcal{N}(5,1)$[15]. The cross-correlation matrix is defined in 15. The analyzed DGP is[16]

$$y = -8\mathbf{x}_1 + 2\mathbf{x}_2 + 2\mathbf{x}_3 + 2\mathbf{x}_4 + 7\mathbf{x}_5 + 3\mathbf{x}_1\mathbf{x}_2 - \mathbf{x}_3\mathbf{x}_5 + 2\mathbf{x}_1\mathbf{x}_4 + \epsilon \tag{17}$$

[14]A similar DGP is also simulated in Tibshirani (1996) with $\mathbf{x} \in \mathbb{R}^4$, and a shallow network with sigmoid activation functions and two hidden nodes; the Gaussian error $\boldsymbol{\epsilon}$ follows the same distribution.

[15]The vector of means is generated from $U[-5,5]$ and then rounded to the closest digit.

[16]The interaction terms are introduced in order to have an unknown network structure. In fact, if no interactions are assumed, the true network structure is a shallow network with one hidden node.

The parameters chosen for the vector of coefficients are generated from a $U[-10, 10]$ and then rounded to the closest digit; the error term is $\epsilon \sim \mathcal{N}(0, 1)$ and it is uncorrelated with the input variables.

For both linear and nonlinear DGPs, a total of 1500 observations are generated, 1200 observations are used for the training set and 300 for the test set. The datasets are normalized so that $\mathbf{x}$ has zero mean and unit variance.

When fitting the neural networks, no optimal tuning of the neural network hyper-parameters and structure is conducted[17]. The reasons for imposing the network hyper-parameters as opposed to fine-tuning them are: (i) it is ensured that the simulation results obtained are not dependent on fine-tuning; (ii) it allows conducting a comparison of the empirical PICPs across the three different methodologies analyzed, and (iii) it allows analyzing the impact that different ps may have on the empirical coverage probabilities.

When the nonlinear DGP is simulated, it is assumed that the neural network structure is known ($Z_1 = 3$ and $Z_2 = 2$). Conversely, when the linear DGP is analyzed - as the true network structure is unknown, and due to the simplicity of the DGP - a shallow network with 5 hidden nodes is considered. When a nonlinear DGP is analyzed a $p = 0.995$ is applied (the true network structure is known and thus a low dropout rate is required); conversely, when a linear DGP is fitted - by imposing $p = [0.995, 0.990, 0.950, 0.900, 0.800]$ - it is possible to analyze the impact that different ps may have on the empirical PICPs. A sensible choice of the network parameters for the linear process is to use the Adam optimizer with learning rate 0.1 and 10 epochs; for the nonlinear process, the Adam optimizer with learning rate 0.01 and 80 epochs is considered instead.

5.2 Simulation Results

Table 1 reports the out-of-sample performance, and the empirical coverages of the three procedures analyzed. When the nonlinear DGP is considered, one could notice that the three methodologies return - for the three different significance levels - prediction intervals with empirical PICPs approximately equal to the theoretical ones. Focusing on the linear DGP, one could notice that the bootstrap approach returns prediction intervals with empirical coverages approximately equal to the significance level at which they are constructed; when the extra-neural network is considered, all prediction intervals - for the different ps considered - have an empirical PICP approximately equal to the nominal one; yet, the MC dropout returns correct prediction intervals only for given values of p.

[17]For the correct choice of the network hyper-parameters, the analyst should ensure that the test set used for parameter tuning and for the aleatoric uncertainty computation is distinct - that is, a hold-out set should also be generated - otherwise, the resulting under-estimation of the aleatoric uncertainty could lead to narrower prediction intervals.

As explained in the previous sections, the epistemic uncertainty in the MC dropout is captured exclusively by dropout at test time (and thus by the dropout rate $q = 1 - p$). Conversely, when the extra-neural network is analyzed, the epistemic uncertainty depends not only on the dropout rate considered, but also on the random weight initialization used for fitting the T subnetworks. As a result, the correct construction of the prediction intervals using the MC dropout approach requires identifying the optimal dropout rate as opposed to the extra-neural network algorithm proposed in the present paper.

Table 1: The table reports the out-of-sample mean average prediction error (MAPE) and mean squared prediction error (MSPE) for the analyzed procedures. Additionally, the PICPs at the corresponding significance levels (0.99, 0.95, and 0.90) are also reported. EN_1 (EN_2) refers to the extra-neural network fitted to a nonlinear (linear) DGP. MC_1 (MC_2) refers to the MC dropout fitted to a nonlinear (linear) DGP. Finally, BOOT_1 (BOOT_2) reports the results for the bootstrap approach to a nonlinear (linear) DGP.

	Nonlinear			**Linear**										
	EN_1	**MC_1**	**BOOT_1**	**EN_2**					**MC_2**					**BOOT_2**
p	0.995	0.995	-	0.995	0.990	0.950	0.900	0.800	0.995	0.990	0.950	0.900	0.800	-
T = 30														
MAPE	1.4979	3.5218	1.8476	1.0322	1.0493	1.1113	1.1196	1.2383	1.2993	1.3152	1.5834	1.6290	2.0478	1.0451
MSPE	3.8232	19.8904	5.4190	1.7208	1.7544	2.0327	2.0315	2.6099	2.9998	2.9527	4.1508	4.0322	7.0050	1.7037
$\mathbf{PICP_{99}}$	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.02	0.01	0.00	0.01	0.00	0.01
$\mathbf{PICP_{95}}$	0.05	0.04	0.04	0.03	0.03	0.05	0.04	0.06	0.04	0.04	0.01	0.01	0.00	0.03
$\mathbf{PICP_{90}}$	0.07	0.08	0.06	0.09	0.09	0.08	0.10	0.08	0.07	0.07	0.02	0.02	0.00	0.09
T = 50														
MAPE	1.5068	3.5404	1.4480	1.0419	1.0808	1.0668	1.0940	1.2034	1.3044	1.3124	1.5337	1.5842	2.0583	1.0671
MSPE	3.6592	20.0717	3.4133	1.7332	1.8930	1.7991	1.9732	2.4329	3.0670	2.8893	3.8274	3.9688	6.7150	1.8043
$\mathbf{PICP_{99}}$	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.01	0.00	0.01
$\mathbf{PICP_{95}}$	0.04	0.04	0.04	0.04	0.04	0.04	0.05	0.04	0.04	0.04	0.01	0.01	0.00	0.04
$\mathbf{PICP_{90}}$	0.08	0.08	0.09	0.09	0.09	0.09	0.09	0.10	0.08	0.08	0.01	0.02	0.00	0.08
T = 70														
MAPE	1.4756	3.5200	1.6426	1.0423	1.0467	1.1051	1.1611	1.1980	1.3026	1.3042	1.5277	1.5603	2.0060	1.0522
MSPE	3.5096	20.1656	4.3616	1.7131	1.7315	1.9444	2.2202	2.4559	3.0330	2.9104	3.8339	3.8921	6.6385	1.7290
$\mathbf{PICP_{99}}$	0.01	0.01	0.01	0.01	0.01	0.02	0.01	0.01	0.01	0.00	0.00	0.01	0.00	0.01
$\mathbf{PICP_{95}}$	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.00	0.01	0.00	0.04
$\mathbf{PICP_{90}}$	0.09	0.08	0.08	0.10	0.10	0.09	0.09	0.10	0.07	0.08	0.01	0.02	0.00	0.08

The present research extends the results of Levasseur et al. (2017). Similarly, these authors state that the construction of prediction intervals with empirical PICPs approximately equal to the theoretical ones - using the MC dropout approach - depends on the correct choice of the dropout rate. These authors therefore suggest to tune the dropout rate to return the correct prediction intervals. The theoretical analysis in Section 3 coupled with the results in Table 1 clearly show that the prediction intervals computed from the MC dropout rely significantly on the correct choice of the dropout rate. These results also suggest that choosing the dropout rate that maximizes the out-of-sample accuracy guarantees prediction intervals with the correct $\bar{\alpha}$ (the out-of-sample error is minimized for the p that returns correct prediction intervals).

Although the results in Table 1 show that all three procedures return prediction intervals with empirical PICPs close to the theoretical ones for both linear and nonlinear DGPs, the performance of the extra-neural network approach is clearly superior in terms of PICPs and also MAPE and MSPE errors. This is particularly the case for $\alpha = 0.10$. This outperformance is especially remarkable for the linear process for which we do not impose or know a priori the true structure of the network. Focusing on the out-of-sample performance, one could notice that: (i) the out-of-sample errors decrease as T increases, and (ii) for given dropout rates, the ensemble of neural networks outperforms the bootstrap approach. When focusing on both bootstrap and extra-neural network algorithms, due to the independence among the single learners, the out-of-sample MAPE and MSPE decrease as T increases (especially for the nonlinear case); the same is not true for the Monte Carlo dropout due to the perfect correlation between model predictions. Thus, when the number of learners increases, due to the randomness introduced into the system, the reduction in variance ensures gains in the out-of-sample prediction accuracy. Similarly, the empirical PICP of the proposed algorithm, especially for $\alpha = 0.10$, improves as T increases.

It is also important to mention that the width of the prediction intervals is defined as $2z_{(1-\alpha/2)}\sigma_e^2$. This measure provides another important metric for the evaluation and comparison of the prediction intervals. Given the aforementioned assumption of unbiasedness of the pointwise model predictions, the MSPEs reported in Table 1 are consistent estimates of the conditional variance of the predicted output given the set of covariates, denoted as σ_e^2 in expression (5). Thus, by comparing the out-of-sample accuracy of the different methodologies in terms of MSPE, we are implicitly comparing the widths of the prediction intervals (i.e., lower MSPE implies a narrower prediction interval). As a result, it is also possible to conclude that the newly proposed methodology returns prediction intervals narrower than both MC dropout and bootstrap approaches when the nonlinear DGP is considered. The same is true when the linear underlying data generating process is considered and the optimal dropout rate q is selected for the extra-neural network.

As previously mentioned, both the bootstrap and the extra-neural network approaches allow distributing in parallel the fitting of the single learners. Therefore, by distributing the implementation of the algorithms across a number of cores equal to T, the execution time of the three algorithms would be equivalent. But even if the execution time can be considered as comparable, the computational load varies significantly, with the Monte Carlo dropout being the most efficient methodology. Therefore, to further compare the performance of the three different approaches analyzed in the present simulation, the average time of execution (across 10 different runs) of the extra neural network, Monte Carlo dropout, and bootstrap approaches –with a sequential job implementation– on a Kaggle notebook, is collected for increasing $T = 30, 50, 70$, reporting also the relative time of execution with respect to Monte Carlo dropout. Implementing the following analysis on a Kaggle notebook allows providing results based on system requirements that can be considered "standard" and publicly available (and thus, it provides also an objective measure of the computational load of the single algorithms publicly accessible).

When the nonlinear data generating process is considered, the ratios of the average time of execution of the extra-neural network with respect to the Monte Carlo dropout, for $T = 30, 50$ and 70, are 23.09, 34.81, and 42.47. For the bootstrap approach the ratios are 22.54, 34.34, and 41.61. When the linear data generating process and a dropout rate of 0.005 are considered, the relative execution times of the extra-neural network are 14.29, 17.55, and 19.27; for the bootstrap approach, we obtain 13.44, 16.72, and 18.84. Therefore, one can conclude that for increasing T, the gains in computational time from using the Monte Carlo dropout approach increase at a slower rate than the increase in the ensemble dimension. One also should notice that the difference, in terms of the average time of execution, between the newly proposed extra-neural network and the bootstrap approach is expected to increase for increasing neural network sizes.

5.3 Analysis of the correlation among learners

An additional exercise is carried out to test if the assumption of independence among the single learners used in implementing the extra-neural network algorithm is correct, and thus if Equation 11 is valid. The same simulation settings as described in subsection 5.1 are implemented: to approximate the nonlinear DGP defined in (16), a neural network with two hidden layers, Adam optimizer with learning rate 0.01, 80 epochs and $p = 0.995$ is trained on a sample of $M = 1200$ observations.

While studying if c_i in Equation 10 can be rightly assumed equal to zero, one should notice that it is important to respect the heteroscedasticity assumption used in computing the epistemic uncertainty, $\widehat{\sigma}^2_{\widehat{\boldsymbol{\omega}}}(\mathbf{x}_i)$. That is, the correlation among predictors should be computed for a given $\mathbf{x}_i$, returning n correlations among the T learners with $n = 300$ being the length of the test sample.

In order to do so, it is necessary to introduce variability in the ensemble predictions, while preserving the extra-neural network estimates. The strategy adopted in the present paper is to introduce uncertainty through bootstrap methods. The bootstrap procedure for T learners is as follows: given $\{\mathbf{x}_i\}_{i=1}^M$ the set of covariates and $\{y_i\}_{i=1}^M$ the target variable with M the length of the training set and $\{\mathbf{x}_i\}_{i=1}^n$ the set of covariates in the test sample, each subnetwork f_t with $t = 1, \dots, T$ is trained on $\{\mathbf{x}_i, y_i\}_{i=1}^M$ and the trained weights $\widehat{\boldsymbol{\omega}}^{(t)} = \{\mathbf{W}^{1,(t)}, \dots, \mathbf{W}^{N,(t)}, b_1^{(t)}, \dots, b_N^{(t)}\}$ are stored. Then, $\{\mathbf{x}_i\}_{i=1}^n$ is resampled with replacement B times to obtain B bootstrapped replica $\{\mathbf{x}_i^{\star,b}, \dots, \mathbf{x}_i^{\star,B}\}_{i=1}^n$ with $b = 1, \dots, B$. From the resampled datasets, B out-of-sample predictions $\{\widehat{y}_i^{\star,b}, \dots, \widehat{y}_i^{\star,B}\}_{i=1}^n$ are obtained for each f_t for a fixed set of weights $\widehat{\boldsymbol{\omega}}^{(t)}$, with $t = 1, \dots, T$. Starting from the bootstrapped predictions, the covariance among the T learners is computed as follows: for each observation $\mathbf{x}_i$ with $i = 1, \dots, n$, B predictions from each subnetwork f_t are stored. As a result, n variance-covariance matrices $(T \times T)$ are obtained and for each $\mathbf{x}_i$ it is possible to compute the average covariance between the f_t subnetworks (thus satisfying Equation 10). Below, the average covariance (average across the n obtained covariances) is reported.

Table 2: The table reports the summary statistics (mean, standard deviation, skewness, Kurtosis, and the p-value from the Shapiro test for normality) for the bootstrap replica for each of the T learners. Additionally, it also reports the true mean prediction (prior to resampling) for each subnetwork.

	True Mean Pred.	**Mean Boot.**	**Stand. Dev**	**Kurtosis**	**Skewness**	**Shapiro**
1	71.5608	71.7642	4.5904	3.2397	0.3521	0.0739
2	71.9841	71.7823	4.4271	3.0619	-0.0639	0.8900
3	71.6901	71.8064	4.6287	2.6389	0.0480	0.6274
4	71.4629	71.7165	4.7014	2.4896	0.1851	0.0229
5	71.7296	71.9098	4.7610	2.8031	-0.0253	0.6893
6	72.1818	72.4281	4.2307	3.3075	0.0607	0.2826
7	71.5742	71.2419	4.5291	3.0823	0.0541	0.7663
8	71.6073	71.6189	4.4282	2.8779	-0.0019	0.3129
9	71.5663	71.4349	4.7200	2.5119	0.1792	0.0484
10	71.4790	71.7649	4.9590	3.2489	0.3611	0.0143
11	71.6780	71.9073	4.8519	3.1276	0.1247	0.9003
12	72.0937	72.0630	4.4939	3.4614	0.5351	0.0007
13	71.6108	71.8293	4.4362	2.6581	-0.0052	0.2343
14	71.5063	71.3657	4.2569	2.5846	0.2001	0.1142
15	71.6539	71.5459	4.9013	3.6469	0.1230	0.0749
16	71.5550	71.7832	4.7242	3.0683	0.1915	0.6314
17	71.7348	71.5783	4.8394	2.7690	0.0973	0.5168
18	71.7149	71.3114	4.4329	2.9711	0.0177	0.8693
19	71.4578	71.6132	4.2455	2.4722	0.1635	0.0650
20	71.6526	71.5065	4.6349	3.0424	0.1277	0.2733
21	71.7326	72.2212	4.6082	3.0612	-0.0401	0.8511
22	71.5498	71.6016	4.9611	3.2911	0.3268	0.0115
23	71.6915	71.7282	4.5501	3.0412	0.1021	0.4382
24	71.4712	71.3337	4.8462	3.4061	0.1846	0.1460
25	71.7813	72.1153	4.7040	3.2497	-0.0248	0.8850
26	71.5938	71.7936	4.6807	2.9165	0.1540	0.7936
27	71.5279	71.3395	4.5650	2.5916	-0.2339	0.0585
28	71.5668	71.7543	4.6246	3.0035	0.2647	0.0993

The above bootstrap exercise is conducted imposing $B = 100$ and $T = 30$[18], when the underlying data generating process is nonlinear. To control for the validity of the bootstrap exercise, it is necessary to study if the bootstrap predictions of each f_t learner are unbiased and, similarly, if the ensemble of the bootstrap predictions is an unbiased estimate of the out-of-sample target variable. For each of the $T = 30$ learners, the mean, the standard deviation, the kurtosis, the skewness, and the p-value from the Shapiro test are collected. The results reported in Table 2 show how the bootstrapped means are approximately equal to the mean prediction obtained on the test set $\{\mathbf{x}_i\}_{i=1}^n$, and that the null hypothesis of normality is rejected at 0.05

[18]Unreported results, available upon request, confirm the obtained results for $T = 50, 70$.

significance level only five times[19].

The average covariance obtained with the procedure above is 0.0093. Thus, these results show how the bootstrap samples from the extra-neural network predictions are independent and unbiased, with the average prediction over the test sample approximately normal. The average prediction is approximately equal to the average of the observed target variable, and we fail to reject the null hypothesis of normality from the Shapiro-Wilk test at 0.05 significance level. It is then possible to conclude that also the ensemble of the bootstrap predictions is an unbiased estimate of the out-of-sample target variable. To further corroborate the assumption of independence, the average correlation (and absolute correlation) across the T average bootstrapped predictions is computed. The average absolute correlation among the T learners is 0.0498, and the average correlation is 0.0124.

To summarize, the simulation results prove that the predictions across the T learners in the extra-neural network are independent and thus, that Equation 11 is valid. Additionally, the simulated results show that the proposed extra-neural network methodology not only returns correct prediction intervals but it also improves the forecast accuracy for both deep and shallow ensembles. Based on Equation 10, one could also notice that the PICP improves not only by correctly estimating the variance but also by providing more accurate pointwise predictions of the true observations. The following section, by using the experimental settings of Hernández-Lobato and Adams (2015), evaluates the out-of-sample accuracy in terms of root mean square prediction error (RMSPE) of the novel approach for real world datasets.

6 Empirical Analysis

Hernández-Lobato and Adams (2015) after proposing a novel scalable method for learning Bayesian neural networks - called probabilistic backpropagation (PBP) - evaluate the performance of their novel methodology on real world datasets. The experimental settings used in their evaluation are widely adopted by the literature focusing on deep learning (see, for example, Gal and Ghahramani 2016a; and Lakshminarayanan et al., 2017) when evaluating novel algorithms. Using their experimental setup ensures comparability of the results with the variational inference method by Graves (2011), their novel probabilistic backpropagation, the MC dropout in Gal and Ghahramani (2016a), and the deep ensemble approach developed by Lakshminarayanan et al. (2017). Therefore, we implement the extra-neural nets approach proposed herein to the different datasets discussed by these authors and compare their performance in terms of RMSPE.[20] Furthermore, the present empirical application (focused on shallow struc-

[19]Two subnetworks are missing as the random fixed Bernoulli mask returned the null model.

[20]The original experiment of Hernández-Lobato and Adams (2015) evaluates the models not only in terms of RMSPE but also in terms of predictive log-likelihood (the latter being extremely relevant in Bayesian learning). Being the present paper focused on evaluating the accuracy of different procedures in constructing predic-

tures in order to allow for cross-comparability) complements the results reported in Table 1 by analyzing the RMSPE of the extra-neural network in large dimensional settings. The obtained RMSPEs (see also Equation 10), by capturing both bias and variance of the predictions, provide an additional indication regarding the accuracy of the prediction intervals obtained from the extra-neural network algorithm.

The experimental setup is as follows: 10 datasets are analyzed. Each dataset is split into random training (0.90 of the observations) and test (0.10 of the observations) sets $B = 20$ times, and the average test set performance (RMSPE) and relative standard error are reported. As an exception, the Protein and Year Prediction MSD datasets are split only 5 and 1 times into train and test sets. The datasets are normalized to guarantee that the regressors have zero mean and unit standard deviation. The same network architecture is considered: one hidden layer ReLu neural network with $Z_1 = 50$ for the small datasets and $Z_1 = 100$ for the larger Protein and Year Prediction MSD datasets. Each neural network is trained for 40 epochs. Following Gal and Ghahramani (2016a), we use a dropout rate of 0.05, Adam optimizer and a batch size of 32. We decided to use the same dropout rate as in Gal and Ghahramani (2016a) for comparability reasons. When implementing the Adam optimizer, the parameters of interest are: a) the learning rate, b) β_1, capturing the exponential decay for the first moment estimates, c) β_2, representing the exponential decay for the second moment estimates, d) the learning decay over each update, e) whether or not to apply the AMSGrad variant of the algorithm (Reddi et al., 2019), f) whether or not to clip the gradients when their L_2 norm exceeds a pre-specified value, and g) whether or not to clip the gradients when their absolute value exceeds a pre-specified value. When performing the empirical analysis, the learning rate adopted is 0.01, $\beta_1 = 0.9$, and $\beta_2 = 0.999$, the learning decay is set equal to 0, the AMSGrad variant and both options of gradients clipping are not implemented (defaults in the Adam optimizer of Keras for RStudio). For the implementation of the competing methods, we refer the reader to Gal and Ghahramani (2016a), Hernández-Lobato and Adams (2015), and Lakshminarayanan et al. (2017). Lakshminarayanan et al. (2017) use 5 networks in their ensemble and Gal and Ghahramani (2016a) perform 10000 stochastic forward passes[21]. In order to allow for a fair comparison between the deep ensemble of Lakshminarayanan et al. (2017) and the novel algorithm proposed in the present paper, we fit an extra-neural network with 5 subnetworks. Furthermore, in order to compare the predictive performance of Algorithm 1 with the MC dropout of Gal and Ghahramani (2016a), we also consider an extra-neural network with 70 subnetworks.

tion intervals for regression tasks, only the former performance metric is reported.

[21]This is not directly reported by the authors and it is inferred from the code reported in their Github page (Gal and Ghahramani, 2016c).

Table 3: The table reports the average test RMSPE and relative standard error (SE) for the variational inference method (VI) of Graves (2011); the probabilistic backpropagation (PBP) of Hernández-Lobato and Adams (2015); the MC dropout of Gal and Ghahramani (2016a); and the deep ensemble proposed by Lakshminarayanan et al. (2017). Extra-net$_1$ uses $T = 70$, while Extra-net$_2$ uses $T = 5$. The number of observations is reported as $M + n$, and the dimension of the input as d. In bold the lowest average RMSPE is highlighted.

Dataset	**(M+n)**	**d**	**VI**	**PBP**	**MC-Dropout**	**Deep Ens.**	**Extra-net$_1$**	**Extra-net$_2$**
Boston Housing	506	13	4.32±0.29	3.01±0.18	2.97±0.19	3.28±1.00	**2.80±0.15**	3.22±0.21
Concrete Strength	1030	8	7.19±0.12	5.67±0.09	5.23±0.12	6.03±0.58	5.26±0.15	**5.09±0.10**
Energy Efficiency	768	8	2.65±0.08	1.80±0.05	1.66±0.04	2.09±0.29	**0.59±0.01**	0.72±0.02
Kin8nm	8192	8	0.10±0.00	0.10+0.00	0.10±0.00	0.09±0.00	**0.08±0.00**	**0.08±0.00**
Naval Propulsion	11934	16	0.01±0.00	0.01±0.00	0.01±0.00	**0.00±0.00**	0.01±0.00	0.03±0.00
Power Plant	9568	4	4.33±0.04	4.12±0.03	**4.02±0.04**	4.11±0.17	4.12±0.05	4.24±0.04
Protein Structure	45730	9	4.84±0.03	4.73±0.01	4.36±0.01	4.71±0.06	**4.32±0.01**	4.36±0.02
Wine Quality Red	1599	11	0.65±0.01	0.64±0.01	**0.62±0.01**	0.64±0.04	0.63±0.01	0.64±0.01
Yacht Hydrodynamics	308	6	6.89±0.67	1.02±0.05	1.11±0.09	1.58±0.48	**0.72±0.06**	0.97±0.06
Year Protection MSD	515345	90	9.03±NA[22]	8.88±NA	8.85±NA	8.89±NA	**8.84±NA**[23]	8.97±NA

Table 3 reports the average RMSPE and standard errors; in bold are reported the lowest average RMSPEs. From the results reported in Table 3, it is possible to compute the $1 - \alpha$ confidence intervals around the average RMSPE (for those cases for which the simulated standard errors are different from zero and well defined) as $\mu_{RMSPE} \pm z_{1-\alpha/2} * \sigma/\sqrt{B}$, with $\sigma/\sqrt{B}$ the simulated standard errors over B simulated samples. More importantly, it is possible to test the null hypothesis: $\mathcal{H}_0 : \mu^a_{RMSPE} = \mu^b_{RMSPE}$ that assesses statistically the relative magnitudes of the average RMSPEs in pairwise comparisons. The extra-neural network is considered as model b(aseline) and the competing models in Table 3 as model a(lternative). We test this pairwise null hypothesis by constructing the following confidence intervals:

$$(\mu^a_{RMSPE} - \mu^b_{RMSPE}) \pm z_{1-\alpha/2} \frac{\sigma^a + \sigma^b}{\sqrt{B}} \tag{18}$$

The usual interpretation of confidence intervals for two-sided tests applies. If the zero is included in the confidence interval (18), one cannot reject the null hypothesis of equal average RMSPEs across competing models at an α significance level. If the confidence interval is inside the negative real line, model b underperforms model a, and if the confidence interval is inside the positive real line, we conclude that model b outperforms model a, at 0.05 significance level. It is worth noting that the confidence interval (18) is constructed under the assumption that the RMSPE are independent across models. It is also plausible to assume that these statistics are positively correlated. In the latter case we are not able to compute the correct confidence intervals as we do not have specific information on the RMSPEs of each simulation for the competing models. Nevertheless, the positive correlation between the models guarantees a smaller confidence interval than (18) but centred at the same point. Therefore, rejecting the null hypothesis of equal average

[22]For the last dataset, it is not possible to compute the SE as only 1 split is performed.

[23]If the predictions are rounded to the closest digit, or the floor operator is used, the RMSPE is 8.85.

RMSPEs using the above expression guarantees the rejection of the null hypothesis also in case of positive correlation, due to narrower confidence intervals.[24] As an illustrative example we discuss the results for the Energy Efficiency dataset. In this case the confidence intervals for the difference between the RMSPEs of the variational inference method (Graves, 2011), of the probabilistic backpropagation (Hernández-Lobato and Adams, 2015), of the MC Dropout (Gal and Ghahramani, 2016a), of the Deep ensemble (Lakshminarayanan et al., 2017), with the extra-neural network are: $[1.88; 2.24]$, $[1.09; 1.33]$, $[0.97; 1.17]$, and $[0.91; 2.09]$, respectively. Clearly, these intervals provide empirical evidence rejecting the null hypothesis and in favour of the extra-neural network approach. Unreported results, available from the authors upon request, show that at a 95% confidence level, the extra neural network approach outperforms the variational inference method (Graves, 2011) in all cases except for the Wine Quality Red dataset. The method also outperforms the probabilistic backpropagation (Hernández-Lobato and Adams, 2015) and the Monte Carlo dropout (Gal and Ghahramani, 2016a) for the Protein Structure and Yacht Hydrodynamics datasets. It also outperforms the deep ensemble approach of Lakshminarayanan et al. (2017)[25] at a 0.05 significance level for the Protein Structure datasets and, importantly, the reported standard errors are considerably lower than under this approach.

The above null hypothesis to test the equality of RMSPEs can be also extended to multiple comparisons across models. In particular, we are interested in testing the composite null hypothesis $\mathcal{H}_0 : \mu^{a_j}_{RMSPE} = \mu^{b}_{RMSPE}$, for all $j = 1, \ldots, 4$, with a_j denoting the four competing models. The alternative hypothesis is given by the RMSPE of model b being different from at least one of the competitors. In this case it is well known that the critical values need to be adjusted to guarantee an overall significance level equal to 0.05 for the composite null hypothesis. Following the same strategy, we test this hypothesis using confidence intervals as in (18) but adjusting the critical value. More precisely, based on $0.05 = 1 - (1 - \alpha)^4$, computing the individual confidence intervals at 0.013 significance level will ensure reaching conclusions at an overall 0.05 significance level. The corresponding critical value from the Normal distibution is $z_{1-0.013/2} = 2.49$. For the Energy Efficiency dataset discussed above, when adjusted for multiple comparison, the corresponding confidence intervals of the competing models against the extra-neural network model are $[1.84; 2.28]$, $[1.06; 1.36]$, $[0.95; 1.19]$, and $[0.75; 2.25]$, which provide ample evidence of the superiority of the latter method in terms of RMSPE. Unreported results also show that the extra-neural network outperforms the variational inference and the probabilistic backpropagation in the Protein Structure and Yacht Hydrodynamics datasets. It outperforms the Monte Carlo dropout in the Yacht Hydrodynamics datasets, and the deep ensemble approach in the

[24]More specifically, the amplitude of the confidence interval under positive correlation between the observations is $\frac{((\sigma^a)^2+(\sigma^b)^2-2cov(a,b))^{1/2}}{\sqrt{B}}$, that is strictly smaller than $\frac{\sigma^a+\sigma^b}{\sqrt{B}}$ if $cov(a, b) > 0$.

[25]The deep ensemble proposed by Lakshminarayanan et al. (2017) is a novel algorithm that it is shown to consistently outperform classic bootstrap based approaches.

Protein Structure dataset.

As an aside comment, we also note that, with the exception of the Concrete Strength dataset, increasing the number of learners leads to an increase in the out-of-sample accuracy, further validating the independence among the different predictors.

As an additional robustness exercise, the extra-network algorithm is also implemented using deep structures: when the small datasets are considered, deep neural networks with 5 hidden layers and 10 hidden nodes each are trained; when the large datasets are analyzed, the depth of the subnetworks is 5 with equal width of 20 hidden nodes per layer. The results suggest that the outperformance of the extra-neural network approach over state-of-art deep learning algorithm is robust also for deep structures[26].

7 Conclusions

This article proposes a novel model based on an ensemble of deep neural networks. Our novel approach builds upon the work of Geurts et al. (2006) by extending the extremely randomized trees approach to ensembles of neural networks. The introduction of a Bernoulli mask allows for an additional randomization scheme in the prediction of the individual learners that ensures not only the correct construction of the prediction intervals, but also training the neural networks on the entire training set, better generalization performance due to randomized architecture structures, and accuracy gains due to an increase in the diversity among the members of the ensemble. The randomization across individual learners guarantees mutual independence across individual prediction models reducing the variance of the ensemble predictor by $1/T$, with T the number of models comprising the ensemble prediction.

The performance of the proposed algorithms is assessed in a comprehensive Monte Carlo exercise. The simulation results show the excellent performance of the proposed approach in terms of mean square prediction error. Similarly, the empirical PICPs obtained from the three competing ensemble methods assessed in this study (MC dropout, bootstrap approach, and extra-neural network) are close to the nominal significance levels when tested using out-of-sample data. Nevertheless, the extra-neural network introduced in this paper is shown to outperform the competing models in most cases but more significantly at a 10% significance level. Additionally, the simulation results also show the robustness of the extra-neural network approach to the choice of the dropout rate, as opposed to the MC dropout approach. In fact, in order to return correct prediction intervals with MC dropout, it is necessary to fine-tune the dropout rate that minimizes the out-of-sample error. This is not necessary for the extra-net approach.

These methods for prediction using ensembles of neural networks are further evaluated on

[26]The results –available upon request– are not reported in Table 3 as they require changes in the experimental settings (network structure) of the original experiment of Hernández-Lobato and Adams (2015).

real world datasets using the experimental settings of Hernández-Lobato and Adams (2015). The results suggest that the extra-neural network approach outperforms state-of-the-art deep learning algorithms in terms of out-of-sample mean square prediction error.

References

[1] Breiman, L. (2001) ”Random forests” *Machine learning*; 45(1), pp. 5 - 32.

[2] Calvo-Pardo, H. F., Mancini, T. and Olmo, J. (2020) ”Optimal Deep Neural Networks by Maximization of the Approximation Power”. Available at SSRN: `https://ssrn.com/abstract=3578850` or `http://dx.doi.org/10.2139/ssrn.3578850`.

[3] Carney, J.G., Cunningham, P. and Bhagwan, U. (1999) ”Confidence and prediction intervals for neural network ensembles”. In *IJCNN'99 International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*; 2, pp. 1215 - 1218.

[4] Claeskens, G., Croux, C. and Van Kerckhoven, J. (2006) ”Variable selection for logistic regression using a prediction focused information criterion” *Biometrics*; 62(4), pp.972 - 979.

[5] Cortes-Ciriano, I. and Bender, A. (2019) ”Reliable prediction errors for deep neural networks using test-time dropout” *Journal of Chemical Information and Modeling*; 59(7), pp. 3330 - 3339.

[6] Cybenko, G. (1989) ”Approximation by superpositions of a sigmoidal function” *Mathematics of control, signals and systems*; 2(4), pp. 303 - 314.

[7] Denker, J.S. and LeCun, Y. (1991) ”Transforming neural-net output levels to probability distributions”. In *Advances in neural information processing systems*; pp. 853 - 859.

[8] De vieaux, R.D., Schumi, J., Schweinsberg, J. and Ungar, L.H. (1998) ”Prediction intervals for neural networks via nonlinear regression” *Technometrics*; 40(4), pp. 273 - 282.

[9] Dipu Kabir, H.D., Khosravi, A., Hosen, M.A. and Nahavandi, S. (2018) ”Neural network-based uncertainty quantification: A survey of methodologies and applications” *IEEE access*; 6, pp. 36218 - 36234.

[10] Dormann, C.F., Calabrese, J.M., Guillera-Arroita, G., Matechou, E., Bahn, V., Barton, K., Beale, C.M., Ciuti, S., Elith, J., Gerstner, K. and Guelat, J. (2018) ”Model averaging in ecology: A review of Bayesian, information-theoretic, and tactical approaches for predictive inference” *Ecological Monographs*; 88(4), pp.485 - 504.

[11] Efron, B. (1979) ”Bootstrap methods: another look at the jackknife”. In *Annals of Statistics*;7(1); pp. 1 - 26.

[12] El Karoui, N. and Purdom, E. (2018) "Can we trust the bootstrap in high-dimensions? the case of linear models" *The Journal of Machine Learning Research*; 19(1), pp.170 - 235.

[13] Errouissi, R., Cardenas-Barrera, J., Meng, J., Castillo-Guerra, E., Gong, X. and Chang, L. (2015) ”Bootstrap prediction interval estimation for wind speed forecasting”. In *2015 IEEE Energy Conversion Congress and Exposition (ECCE)*; pp. 1919 - 1924.

[14] Farrell, M.H., Liang, T. and Misra, S. (2021) "Deep neural networks for estimation and inference" *Econometrica*; 89(1), pp. 181 - 213.

[15] Fernandez, C., Ley, E. and Steel, M.F. (2001) ”Model uncertainty in cross-country growth regressions” *Journal of applied Econometrics*; 16(5), pp.563 - 576.

[16] Gal, Y. and Ghahramani, Z. (2016a) ”Dropout as a bayesian approximation: Representing model uncertainty in deep learning”. *In international conference on machine learning*; pp. 1050 - 1059.

[17] Gal, Y. and Ghahramani, Z. (2016b) ”Bayesian convolutional neural networks with Bernoulli approximate variational inference” *arXiv preprint arXiv:1506.02158*

[18] Gal, Y. and Ghahramani, Z. (2016c) ”Dropout as a bayesian approximation: Representing model uncertainty in deep learning”. Github repository:`https://github.com/yaringal/DropoutUncertaintyExps`. Accessed [Online]: 02/10/2020.

[19] Géron, A. (2019) *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* O’Reilly Media.

[20] Geurts, P., Ernst, D. and Wehenkel, L. (2006) ”Extremely randomized trees. Machine learning”; 63(1), pp. 3 - 42.

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

[22] Graves, A. (2011) ”Practical variational inference for neural networks”. In *Advances in neural information processing systems*; pp. 2348 - 2356.

[23] Hernández-Lobato, J.M. and Adams, R. (2015) ”Probabilistic backpropagation for scalable learning of bayesian neural networks”. In *International Conference on Machine Learning*; pp. 1861 - 1869.

[24] Heskes, T. (1997) ”Practical confidence and prediction intervals”. In *Advances in neural information processing systems*; pp. 176 - 182.

[25] Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R.R. (2012) ”Improving neural networks by preventing co-adaptation of feature detectors” *arXiv preprint arXiv:1207.0580.*

[26] Hornik, K. (1991) ”Approximation capabilities of multilayer feedforward networks” *Neural networks*; 4(2), pp. 251 - 257.

[27] Hüllermeier, E. and Waegeman, W. (2020) ”Aleatoric and epistemic uncertainty in machine learning: A tutorial introduction” *arXiv preprint arXiv:1910.09457.*

[28] Hwang, J.G. and Ding, A.A. (1997) ”Prediction intervals for artificial neural networks” *Journal of the American Statistical Association*; 92(438), pp. 748 - 757.

[29] Kendall, A. and Gal, Y. (2017) ”What uncertainties do we need in bayesian deep learning for computer vision?”. In *Advances in neural information processing systems*; pp. 5574 - 5584.

[30] Kingma, D.P., Salimans, T. and Welling, M. (2015) ”Variational dropout and the local reparameterization trick”. In *Advances in neural information processing systems*; pp. 2575 - 2583.

[31] Krogh, A. and Vedelsby, J. (1995) ”Neural network ensembles, cross validation, and active learning”. In *Advances in neural information processing systems*; pp. 231 - 238.

[32] Kull, M., and Flach, P. (2014) ”Reliability maps: A tool to enhance probability estimates and improve classification accuracy”. In: *Proc. ECML/PKDD, European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*; pp. 18 - 33.

[33] Lakshminarayanan, B., Pritzel, A. and Blundell, C. (2017) ”Simple and scalable predictive uncertainty estimation using deep ensembles”. In *Advances in neural information processing systems*; pp. 6402 - 6413.

[34] Lambrou, A., Papadopoulos, H. and Gammerman, A. (2011) ”Reliable confidence measures for medical diagnosis with evolutionary algorithms” *IEEE Transactions on Information Technology in Biomedicine* 15(1), pp.93 - 99.

[35] Leamer, E.E. (2016) ”S-values: Conventional context-minimal measures of the sturdiness of regression coefficients” *Journal of Econometrics*; 193(1), pp.147 - 161.

[36] LeCun, Y., Bengio, Y. and Hinton, G. (2015) ”Deep learning”, *nature*; 521(7553), pp. 436 - 444.

[37] Lee, S., Purushwalkam, S., Cogswell, M., Crandall, D. and Batra, D. (2015) ”Why M heads are better than one: Training a diverse ensemble of deep networks” *arXiv preprint arXiv:1511.06314.*

[38] Leshno, M., Lin, V.Y., Pinkus, A. and Schocken, S. (1993) ”Multilayer feedforward networks with a nonpolynomial activation function can approximate any function” *Neural networks*; 6(6), pp. 861 - 867.

[39] Levasseur, L.P., Hezaveh, Y.D. and Wechsler, R.H. (2017) ”Uncertainties in parameters estimated with neural networks: Application to strong gravitational lensing” *The Astrophysical Journal Letters*; 850(1), p.L7.

[40] Li, Y. and Clyde, M.A. (2018). "Mixtures of g-priors in generalized linear models." *Journal of the American Statistical Association*; 113, pp. 1828-1845.

[41] Lu, Z., Pu, H., Wang, F., Hu, Z. and Wang, L. (2017) ”The expressive power of neural networks: A view from the width” In *Advances in neural information processing systems*; pp. 6231 - 6239.

[42] Maeda, S.I. (2014) ”A Bayesian encourages dropout” *arXiv preprint arXiv:1412.7003.*

[43] Mei, S., Montanari, A. and Nguyen, P.M. (2018) ”A mean field view of the landscape of two-layer neural networks” *Proceedings of the National Academy of Sciences*; 115(33), pp. 7665 - 7671.

[44] Min, C.K. and Zellner, A. (1993) ”Bayesian and non-Bayesian methods for combining models and forecasts with applications to forecasting international growth rates” *Journal of Econometrics*; 56(1-2), pp.89 - 118.

[45] Neal, R.M. (2012) *Bayesian learning for neural networks*; Vol. 118. Springer Science & Business Media.

[46] Nix, D.A. and Weigend, A.S. (1994) ”Estimating the mean and variance of the target probability distribution”. In *Proceedings of 1994 ieee international conference on neural networks (ICNN’94)*; 1, pp. 55 - 60.

[47] Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B. and Swami, A. (2017) ”Practical black-box attacks against machine learning”. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*; pp. 506 - 519.

[48] Pearce, T., Brintrup, A., Zaki, M. and Neely, A. (2018) ”High-quality prediction intervals for deep learning: A distribution-free, ensembled approach”. In *International Conference on Machine Learning*; pp. 4075 - 4084.

[49] Raftery, A.E., Madigan, D. and Hoeting, J.A. (1997) ”Bayesian model averaging for linear regression models” *Journal of the American Statistical Association*; 92(437), pp.179 - 191.

[50] Reddi, S.J., Kale, S. and Kumar, S. (2019) "On the convergence of adam and beyond" *arXiv preprint arXiv:1904.09237.*

[51] Schmidhuber, J. (2015) ”Deep learning in neural networks: An overview” *Neural networks*; 61, pp. 85 - 117.

[52] Seber, G.A.F. and C.J. Wild (1989) *Nonlinear regression.* New York, Wiley.

[53] Senge, R., Bösner, S., Dembczynski, K., Haasenritter, J., Hirsch, O., Donner-Banzhohh, N., and Hüllermeier, E. (2014) ”Reliable classification that distinguish aleatoric and epsitemic uncertainty” *Information Sciences*; 255, pp. 16-19.

[54] Serpell, C., Araya, I., Valle, C. and Allende, H. (2019) ”Probabilistic Forecasting Using Monte Carlo Dropout Neural Networks”. In *Iberoamerican Congress on Pattern Recognition*; pp. 387 - 397. Springer, Cham.

[55] Smith, J. and Wallis, K.F. (2009) ”A simple explanation of the forecast combination puzzle” *Oxford Bulletin of Economics and Statistics*; 71(3), pp.331 - 355.

[56] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014) ”Dropout: a simple way to prevent neural networks from overfitting” *The journal of machine learning research*; 15(1), pp. 1929 - 1958.

[57] Steel, M.F. (2020) ”Model averaging and its use in economics” *Journal of Economic Literature*; 58(3), pp.644 - 719.

[58] Stock, J.H. and Watson, M.W. (2004) ”Combination forecasts of output growth in a seven-country data set” *Journal of forecasting*; 23(6), pp.405 - 430.

[59] Tibshirani, R. (1996) ”A comparison of some error estimates for neural network models” *Neural Computation*; 8(1), pp. 152 - 163.

[60] Ungar, L.H., De Veaux, R.D. and Rosengarten, E. (1996) ”Estimating prediction intervals for artificial neural networks”. In *Proc. of the 9th Yale Workshop on Adaptive and Learning Systems.*

[61] Varshney, K.R. and Alemzadeh, H. (2017) ”On the safety of machine learning: Cyber-physical systems, decision sciences, and data products” *Big data*; 5(3), pp.246-255.

[62] Wang, H., Zhang, X. and Zou, G. (2009) ”Frequentist model averaging estimation: a review” *Journal of Systems Science and Complexity*; 22(4), p.732.

[63] Warde-Farley, D., Goodfellow, I.J., Courville, A. and Bengio, Y. (2014) ”An empirical analysis of dropout in piecewise linear networks” *arXiv preprint arXiv:1312.6197.*

[64] Yang, F., Wang, H.Z., Mi, H. and Cai, W.W. (2009) ”Using random forest for reliable classification and cost-sensitive learning for medical diagnosis” *BMC bioinformatics*; 10(S1), p.S22.

[65] Zhang, X., Lu, Z. and Zou, G. (2013) ”Adaptively combined forecasting for discrete response time series” *Journal of Econometrics*; 176(1), pp.80 - 91.

[66] Zhang, X., Yu, D., Zou, G. and Liang, H. (2016) ”Optimal model averaging estimation for generalized linear models and generalized linear mixed-effects models” *Journal of the American Statistical Association*; 111(516), pp.1775 - 1790.

[67] Zhou, Z.H. (2012) *Ensemble methods: foundations and algorithms.* CRC press.

[68] Zhu, L. and Laptev, N. (2017) ”Deep and confident prediction for time series at uber”. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*; pp. 103 - 110.

[69] Zou, D., Cao, Y., Zhou, D. and Gu, Q. (2018) ”Stochastic Gradient Descent Optimizes Over-parameterized Deep ReLU Networks” *arXiv preprint arXiv:1811.08888.*

A Review of models for prediction intervals

The prediction intervals for the output of a ReLu DNN are derived from its predictive distribution. This distribution can be approximated asymptotically using a Normal distribution; by resampling methods using bootstrap procedures; and by simulation methods using Monte Carlo dropout. In this section we review the prediction intervals obtained from these procedures.

A.1 Asymptotic prediction intervals (Delta Method)

In a neural network setting we estimate the predictive variance σ_e^2 using the test sample, of size n, such that $\widehat{\sigma}_e^2 = \widehat{\sigma}_{\widehat{\boldsymbol{\omega}}}^2(\mathbf{x}_i) + \widehat{\sigma}_\epsilon^2$. Under the assumption of homoscedasticity of the error term over the test sample, we can estimate consistently the aleatoric uncertainty such that $\widehat{\sigma}_\epsilon^2 = \frac{1}{n}\sum_{i=1}^n (y_i - f(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}))^2$. However, estimating the variance due to parameter estimation is cumbersome unless the specific form of the function $f(\mathbf{x}_i;\boldsymbol{\omega})$ is known to the modeler. Under this stringent assumption, the only uncertainty in the proposed model specification is in the choice of the model parameters $\boldsymbol{\omega}$ and hyperparameters. In this case the literature proposes the delta method to approximate the estimated function $f(\mathbf{x}_i;\widehat{\boldsymbol{\omega}})$ under a first order Taylor expansion around the true parameter vector $\boldsymbol{\omega}$. More specifically, given a data point $\mathbf{x}_i$, and assuming that the number of observations M is sufficiently large to ensure that $\widehat{\boldsymbol{\omega}}$ is a local approximation of the true parameter vector $\boldsymbol{\omega}$, Ungar et al. (1996) show that it is possible to linearize the neural network around the data point as:

$$f(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}) = f(\mathbf{x}_i;\boldsymbol{\omega}) + \mathbf{f}_{\boldsymbol{\omega}i}^{\intercal}(\widehat{\boldsymbol{\omega}} - \boldsymbol{\omega}) + o_P(|\widehat{\boldsymbol{\omega}} - \boldsymbol{\omega}|), \tag{A.1}$$

with $\mathbf{f}_{\boldsymbol{\omega}i}^{\intercal}$ a vector with entries $\partial f(\mathbf{x}_i;\boldsymbol{\omega})/\partial w_N$, with N the number of parameters in $\boldsymbol{\omega}$, defined as (see also De vieaux et al., 1998):

$$\mathbf{f}_{\boldsymbol{\omega}i}^{\intercal} = \left[\frac{\partial f(\mathbf{x}_i;\boldsymbol{\omega})}{\partial w_1}, \frac{\partial f(\mathbf{x}_i;\boldsymbol{\omega})}{\partial w_2}, \ldots, \frac{\partial f(\mathbf{x}_i;\boldsymbol{\omega})}{\partial w_N}\right] = \nabla_{\boldsymbol{\omega}} f(\mathbf{x}_i;\boldsymbol{\omega}) \tag{A.2}$$

Following Seber and Wild (1989), the literature focusing on the delta method (see Hwang and Ding, 1997; Ungar et al., 1996; De vieaux et al., 1998) proposes the following estimator of the asymptotic variance of $f(\mathbf{x}_i;\widehat{\boldsymbol{\omega}})$ evaluated at the true parameter vector $\boldsymbol{\omega}$:

$$\widehat{\sigma}_{\widehat{\boldsymbol{\omega}}}^2(\mathbf{x}_i) \approx \widehat{\sigma}_\epsilon^2[\mathbf{f}_{\boldsymbol{\omega}i}^{\intercal}(\mathbf{J}_{\boldsymbol{\omega}}^{\intercal}\mathbf{J}_{\boldsymbol{\omega}})^{-1}\mathbf{f}_{\boldsymbol{\omega}i}], \tag{A.3}$$

with $\mathbf{J}_{\boldsymbol{\omega}}$ the Jacobian matrix evaluated at $\boldsymbol{\omega}$, defined as

$$\mathbf{J}_{\boldsymbol{\omega}} = \left[\frac{\partial f(\mathbf{x};\boldsymbol{\omega})}{\partial \boldsymbol{\omega}}\right]_{\boldsymbol{\omega}} \tag{A.4}$$

Therefore, using the delta method, the corresponding asymptotic predictive variance of y_i is estimated as $\widehat{\sigma}_e^2 = \widehat{\sigma}_\epsilon^2(1 + S(\widehat{\boldsymbol{\omega}}))$, with $S(\widehat{\boldsymbol{\omega}}) = \mathbf{f}_{\widehat{\boldsymbol{\omega}}i}^\intercal(\mathbf{J}_{\widehat{\boldsymbol{\omega}}}^\intercal \mathbf{J}_{\widehat{\boldsymbol{\omega}}})^{-1}\mathbf{f}_{\widehat{\boldsymbol{\omega}}i}$ and under the central limit theorem, we obtain the following asymptotic prediction interval for y_i:

$$f(\mathbf{x}_i; \widehat{\boldsymbol{\omega}}) \pm z_{1-\alpha/2}\widehat{\sigma}_\epsilon\sqrt{1 + S(\widehat{\boldsymbol{\omega}})}, \tag{A.5}$$

with $z_{1-\alpha/2}$ the relevant critical value from the standard Normal distribution at an α significance level.

Hwang and Ding (1997) showed that, regardless the not identifiability of the weights in a neural network, the prediction interval in (A.5) is asymptotically valid when the feedforward neural network is trained to convergence.

A.2 Bootstrap predictive distribution

Let $\{\mathbf{x}_i\}_{i=1}^M$ be a sample of M observations of the set of covariates, with $\mathbf{x}_i \in \mathbb{R}^d$ and M the length of the train sample. Let $\{y_i\}_{i=1}^M \in \mathbb{R}$ be the output variable, and define $\mathbf{x}_i^\llcorner = (\mathbf{x}_i, y_i) \in \mathbb{R}^{d+1}$. Applying the naive bootstrap proposed by Efron (1979) to this multivariate dataset, we generate the bootstrapped dataset $\boldsymbol{x}^{\llcorner,\star} = \{\boldsymbol{x}_i^{\llcorner,\star}\}_{i=1}^M = \{\boldsymbol{x}_i^\star, y_i^\star\}_{i=1}^M$ by sampling with replacement from the original dataset $\mathbf{x}^\llcorner$.

We define $P(\mathbf{x}_i^\llcorner \in \{\mathbf{x}_i^\llcorner\}_i^M) = 1 - [(M-1)/M]^M$ as the probability of sampling a unique observation $\mathbf{x}_i^\llcorner$. By defining the probability $P(\mathbf{x}_i^\llcorner \in \{\mathbf{x}_i^\llcorner\}_i^M)$, it is possible to understand the results reported in Lee et al. (2015) and El Karoui and Purdom (2018). In particular, as $M \to \infty$, the probability of sampling a unique observation will tend to a Poisson distribution with $\lambda = 1$, and thus to $P(\mathbf{x}_i^\llcorner \in \{\mathbf{x}_i^\llcorner\}_i^M) = 1 - 1/e \approx 0.63$. This implies that for large M, only 63% of unique observations will be used to train the ReLu DNNs having a negative impact on the predictive accuracy (Lee et al., 2015), and on the correct estimation of the variance (El Karoui and Purdom, 2018) for the construction of confidence intervals in mid-dimensional settings ($M/d \to 0.63$).

By repeating this procedure T times, it is possible to obtain T bootstrapped samples defined as $\{\mathbf{x}^{\llcorner,\star(t)}\}_{t=1}^T$. Each bootstrap sample is fitted to a single neural network to obtain an empirical distribution of bootstrap predictions $f(\mathbf{x}^{\star(t)}; \widehat{\boldsymbol{\omega}}^{\star(t)})$, with $\widehat{\boldsymbol{\omega}}^{\star(t)} = \{\mathbf{W}^{1,\star(t)}, \ldots, \mathbf{W}^{N,\star(t)}, b_1^{\star(t)}, \ldots, b_N^{\star(t)}\}$, for $t = 1, \ldots, T$. In this context, a suitable bootstrap prediction interval for y_i at an α significance level is $\left[\widehat{q}_{\alpha/2}, \widehat{q}_{1-\alpha/2}\right]$, with $\widehat{q}_\alpha$ the empirical $\alpha-$quantile obtained from the bootstrap distribution of $f(\mathbf{x}_i; \widehat{\boldsymbol{\omega}}^{\star(t)})$, for $t = 1, \ldots, T$.

Alternatively, we can refine the empirical predictive interval by using the critical value from the Normal distribution. A suitable prediction interval for $\mathbf{x}_i$ from the test sample, with $i = 1, \ldots, n$, is

$$f(\mathbf{x}_i; \widehat{\boldsymbol{\omega}}) \pm z_{1-\alpha/2}\widehat{\sigma}_e^\star, \tag{A.6}$$

with $f(\mathbf{x}_i;\widehat{\boldsymbol{\omega}})$ the pointwise prediction of model (4) and $\widehat{\sigma}_e^{\star 2} = \widehat{\sigma}_{\widehat{\boldsymbol{\omega}}}^{\star 2}(\mathbf{x}_i) + \widehat{\sigma}_\epsilon^2$. Under homoscedasticity of the error term ϵ_i, the aleatoric uncertainty σ_ϵ^2 is estimated from the test sample as $\widehat{\sigma}_\epsilon^2 = \frac{1}{n}\sum_{i=1}^n (y_i - f(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}))^2$, with $\widehat{\boldsymbol{\omega}}$ the set of parameter estimates obtained from the original sample $\{\mathbf{x}_i^{\text{L}}\}_{i=1}^M$. The epistemic uncertainty is estimated from the bootstrap samples as $\widehat{\sigma}_{\widehat{\boldsymbol{\omega}}}^{\star 2}(\mathbf{x}_i) = \frac{1}{T}\sum_{t=1}^T [f(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}^{\star(t)}) - \bar{f}(\mathbf{x}_i)]^2$, with

$$\bar{f}(\mathbf{x}_i) = \frac{1}{T}\sum_{t=1}^T f(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}^{*(t)}). \tag{A.7}$$

Unlike for the delta method, the use of bootstrap methods allows us to ameliorate the effect of the bias in the prediction of the ReLu DNN model. The bias in model (4) is defined as $\mathbb{E}[f(\mathbf{x}_i;\boldsymbol{\omega})] - f(\mathbf{x}_i)$. Therefore, a suitable estimator of this quantity is $\bar{f}(\mathbf{x}_i) - f(\mathbf{x}_i;\widehat{\boldsymbol{\omega}})$, with $\bar{f}(\mathbf{x}_i)$ defined in (A.7), such that the above prediction interval can be refined as

$$f(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}) - \underbrace{(\bar{f}(\mathbf{x}_i) - f(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}))}_{\text{bias correction}} \pm z_{1-\alpha/2}\widehat{\sigma}_e^\star = (2f(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}) - \bar{f}(\mathbf{x}_i)) \pm z_{1-\alpha/2}\widehat{\sigma}_e^\star. \tag{A.8}$$

This bootstrap prediction interval can be further refined by exploiting the average prediction in (A.7). In this case the variance of the predictor is $\overline{\sigma}_{\widehat{\boldsymbol{\omega}}}^{\star 2}(\mathbf{x}_i) = \frac{1}{T}\widehat{\sigma}_{\widehat{\boldsymbol{\omega}}}^{\star 2}(\mathbf{x}_i)$ and the relevant prediction interval is obtained from substituting $f(\mathbf{x}_i;\widehat{\boldsymbol{\omega}})$ in (A.8) with the average prediction $\bar{f}(\mathbf{x}_i)$, such that

$$\bar{f}(\mathbf{x}_i) \pm z_{1-\alpha/2}\widehat{\sigma}_e^\star, \tag{A.9}$$

with $\widehat{\sigma}_e^{\star 2} = \overline{\sigma}_{\widehat{\boldsymbol{\omega}}}^{\star 2}(\mathbf{x}_i) + \overline{\sigma}_\epsilon^2$, where $\overline{\sigma}_\epsilon^2 = \frac{1}{n}\sum_{i=1}^n \left(y_i - \bar{f}(\mathbf{x}_i)\right)^2$. This expression assumes that the covariance between the predictions from the different bootstrap samples is zero. Interestingly, in this case the bias correction is not necessary unless T is small. This is so because the bias term for the average predictor is negligible and given by $\frac{1}{T}\mu_i$.

As highlighted by Dipu Kabir et al. (2018), the variation in the outputs of the different networks will be driven by the different random initialization of the weights (parameter uncertainty) and the different bootstrap samples (data uncertainty). Being the bootstrap procedure able to capture both the aleatoric and epistemic uncertainties, it provides more accurate prediction intervals than other methods (i.e., delta method) as also shown in an extensive simulation study in Tibshirani (1996).

A.3 Monte Carlo Dropout (Stochastic Forward Passes)

This subsection introduces an alternative to bootstrap methods to construct prediction intervals in a ReLU DNN setting. In this case we introduce randomness into the DNN prediction by applying Monte Carlo dropout.

A natural interpretation of this methodology follows from the seminal contribution of Gal and Ghahramani (2016a). These authors develop a new theoretical framework casting dropout training in DNNs as approximate Bayesian inference for deep Gaussian processes. As a byproduct of this theory, Gal and Ghahramani (2016a) provide the tools to model prediction uncertainty with dropout in DNNs. A growing branch of the literature has been focusing on the Bayesian interpretation of dropout[27] (see among others, Gal and Ghahramani (2016a, 2016b) and Kingma et al. (2015)). Maeda (2014) explains how dropout training can be considered an approximate learning method of the model parameters that optimizes a weighted sum of the likelihoods of all possible models.

Starting from this interpretation, one could consider dropout as a tool for the estimation of the posterior of a Bayesian neural network. More specifically, let $p(\widehat{y} \mid \mathbf{x}, \mathbf{X}, \mathbf{Y})$ denote the distribution of the predictive output $\widehat{y}$ conditional on the set of observations $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and $\mathbf{Y} = \{y_1, \ldots, y_n\}$. The predictive probability distribution of the DNN model is

$$p(\widehat{y} \mid \mathbf{x}, \mathbf{X}, \mathbf{Y}) = \int_{\mathbf{\Omega}} p(\widehat{y} \mid \mathbf{x}, \boldsymbol{\omega}) p(\boldsymbol{\omega} \mid \mathbf{X}, \mathbf{Y}) \mathrm{d}\boldsymbol{\omega}, \tag{A.10}$$

with $p(\widehat{y} \mid \mathbf{x}, \boldsymbol{\omega})$ the likelihood function of the observations, and $\boldsymbol{\omega} \in \mathbf{\Omega}$ where $\mathbf{\Omega}$ denotes the parameter space. The posterior probability distribution $p(\boldsymbol{\omega} \mid \mathbf{X}, \mathbf{Y})$ is intractable.

Gal and Ghahramani (2016a) propose DNN dropout to approximate this distribution. More formally, under model dropout, we consider a distribution function $q(\boldsymbol{\omega})$ that follows a Bernoulli distribution, Ber(p). The above predictive distribution in this Bayesian neural network setting can be approximated by

$$p(\widehat{y} \mid \mathbf{x}, \mathbf{X}, \mathbf{Y}) = \int_{\Omega} p(\widehat{y} \mid \mathbf{x}, \boldsymbol{\omega}) q(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega}. \tag{A.11}$$

In practice this predictive distribution can be approximated using Monte Carlo methods. Thus, by sampling T sets of vectors from the Bernoulli distribution $\{\mathbf{r}^{\star(t)}\}_{t=1}^{T}$, one can approximate the above predictive distribution from the random sample $\widehat{y}(\mathbf{x}_i; \widehat{\boldsymbol{\omega}}^{(t)})$, for $i = 1, \ldots, n$, where $\widehat{\boldsymbol{\omega}}^{(t)} = \{\widehat{\mathbf{W}}^{1(t)}, \ldots, \widehat{\mathbf{W}}^{N(t)}, \widehat{b}_1^{(t)}, \ldots, \widehat{b}_N^{(t)}\}$ denotes the sequence of weights associated to the different nodes and layers of the neural network and the associated bias parameters for a given pass t for $t = 1, \ldots, T$.

Using this Monte Carlo dropout technique, Gal and Ghahramani (2016a) propose the first moment from the MC predicted outputs as the model prediction:

$$\bar{f}_{MC}(\mathbf{x}_i) = \frac{1}{T} \sum_{t=1}^{T} \widehat{y}(\mathbf{x}_i; \widehat{\boldsymbol{\omega}}^{(t)}), \text{ for } i = 1, \ldots, n. \tag{A.12}$$

These authors show that, in practice, this is equivalent to performing T stochastic forward passes through the network and averaging the results. This result has been presented in the literature

[27]Hinton et al. (2012) in their seminal paper associate dropout training to a form of Bayesian learning.

before as model averaging. Srivastava et al. (2014) have reasoned empirically that MC dropout can be approximated by averaging the weights of the network (multiplying each weight $\mathbf{W}^n$ by p at test time, and referred to as standard dropout).

Importantly, the model parameters $\boldsymbol{\omega}$ are fixed across random samples implying that the cross-correlation between the predictions $\widehat{y}(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}^{(t)})$ and $\widehat{y}(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}^{(t')})$ for $t, t' = 1, \ldots, T$ is perfect. Then, the predictive variance is defined as

$$\sigma^2_{MC} = \widehat{\sigma}^2_\epsilon + \frac{1}{T^2}\sum_{t=1}^{T}\sum_{t'=1}^{T} E\left[\left(\widehat{y}(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}^{(t)}) - E[\widehat{y}(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}^{(t)})]\right)\left(\widehat{y}(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}^{(t')}) - E[\widehat{y}(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}^{(t')})]\right)\right], \tag{A.13}$$

The first component on the right hand side expression of (A.13) captures the aleatoric uncertainty whereas the second term captures the epistemic uncertainty associated to parameter estimation. The second term includes the estimation of the variance and covariance terms between the different random samples obtained from using dropout. Thus, under the assumption that the approximation error is negligible, the above predictive variance can be estimated as

$$\widehat{\sigma}^2_{MC} = \widehat{\sigma}^2_\epsilon + \frac{1}{T}\sum_{t=1}^{T}\left(\widehat{y}(\mathbf{x}_i;\widehat{\boldsymbol{\omega}}^{(t)}) - \bar{f}_{MC}(\mathbf{x}_i)\right)^2, \tag{A.14}$$

with $\widehat{\sigma}^2_\epsilon = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \bar{f}_{MC}(\mathbf{x}_i)\right)^2$ a consistent estimator of σ^2_ϵ under homoscedasticity of the error term, see also Gal and Ghahramani (2016a) and Kendall and Gal (2017). A suitable prediction interval for y_i under the assumption that $p(\widehat{y}\mid\mathbf{x},\boldsymbol{\omega})$ is normally distributed is

$$\bar{f}_{MC}(\mathbf{x}_i) \pm z_{1-\alpha/2}\widehat{\sigma}_{MC}. \tag{A.15}$$

B Random weight initialization

Shallow and deep neural network are usually trained via the gradient descent (GD) algorithm that - being an iterative algorithm - requires an initial value for the parameter to be estimated. Goodfellow et al. (2016) explain how - due to the difficulty in training neural networks (in particular DNNs) - training algorithms and their convergence depend heavily on the choice of the initialization: different initial points can determine if the algorithm converges or not, if it converges to a global or local minimum, or the speed of convergence. Consequentially, it follows that different weight initializations will lead to different parameter ($\boldsymbol{\omega}$) estimates. More formally, consider Gaussian initialization and define $\{\mathbf{W}^1_0, \ldots, \mathbf{W}^N_0\}$ as the weights generated at the beginning of the GD algorithm; by considering $e = 1, \ldots, E$ epochs, it is possible to define the GD update rule as:

$$\mathbf{W}^n_e = \mathbf{W}^n_{e-1} - \eta\nabla_{\mathbf{W}^n}L(\mathbf{W}^n_{e-1}), \qquad n = 1, \ldots, N \tag{B.1}$$

with η being the learning rate and $\nabla_{\mathbf{W}^n} L(\mathbf{W}^n_{e-1})$ being the partial gradient of the training loss $L(\mathbf{W}^n_{e-1})$ with repsect to $\mathbf{W}^n$ defined as:

$$L(\mathbf{W}^n_{e-1}) = \frac{1}{M}\sum_{i=1}^{M} L(f(\mathbf{x}_i;\widehat{\boldsymbol{\omega}});y_i), \qquad n = 1,\ldots,N \tag{B.2}$$

with M the number of observation in the train set.

From Equation (B.1) and (B.2), one could notice how the estimated $\{\mathbf{W}^n_E\}_{n=1}^N$ depends on $\{\mathbf{W}^n_0\}_{n=1}^N$, η, and the optimization algorithm implemented. Therefore, following the aforementioned literature and by assuming that both learning rate and optimization algorithm are equal across the different bootstrap realizations, the $\widehat{\sigma}^{\star 2}_{\widehat{\boldsymbol{\omega}}}(\mathbf{x}_i)$ can be captured by allowing random weight initialization[28].

[28]The present analysis does not consider recent advances analyzing the relation between neural networks' dimensions (Z_{tot}) and weight initialization that ensures the preservation of the initialization properties during training. As an example, Zou et al. (2018) provide the condition under which Gaussian random initialization and (stochastic) GD produce a set of iterated estimated weights that centers around $\{\mathbf{W}^n_0\}_{n=1}^N$ with a perturbation small enough to guarantee the global convergence of the algorithm, ultimately impacting on the approximation of the $\widehat{\sigma}^{\star 2}_{\widehat{\boldsymbol{\omega}}}(\mathbf{x}_i)$ via random weight initialization.