RECLAIMING UNCERTAINTIES OF DETERMINISTIC DEEP MODELS WITH VINE COPULAS

Natasa Tagasovska[1*], **Axel Brando**[2], **Firat Ozdemir**[1]
[1]Swiss Data Science Center, EPFL & ETH Zurich,
[2]Barcelona Supercomputing Center (BSC-CNS) & Universitat de Barcelona (UB)
[*]Corresponding author: natasa.tagasovska@epfl.ch

ABSTRACT

Despite the major progress of deep models as learning machines, uncertainty estimation remains a major challenge. Existing solutions rely on modified loss functions or architectural changes. We propose to compensate for the lack of built-in uncertainty estimates by supplementing any network, retrospectively, with a subsequent vine copula model, Vine-Copula Neural Networks (VCNN). Through synthetic and real-data experiments, we show that VCNNs could be task (regression/classification) and architecture (recurrent, fully connected) agnostic, with uncertainty estimates comparable to state-of-the-art solutions.

1 INTRODUCTION

Despite the high accuracy of deep models in the recent years, some industries still struggle to include neural networks (NNs) in production/fully operational levels. Such issues originate in the lack of confidence estimates in deterministic neural networks, inherited by all subsequent architectures (convolutional, recurrent, residual ones to name a few). Accordingly, significant amount of work has been put into making NNs more trustworthy and reliable (Lakshminarayanan et al., 2017; Gal, 2016; Guo et al., 2017; Tagasovska & Lopez-Paz, 2019; Havasi et al., 2020). Given their high predictive performance, one would expect that deep models could also provide reasonably well confidence estimates. On the other hand, the challenge persists how to yield such confidence estimates without excessively disrupting the model, i.e. impacting its accuracy and train/inference time. The most popular approaches for overall uncertainty estimation used in production/industry are MC-dropout (Gal & Ghahramani, 2016; Kendall & Gal, 2017), ensembles (Lakshminarayanan et al., 2017) and Bayesian NNs (Hernández-Lobato & Adams, 2015). Each of those methods comes at a price, weather in terms of accuracy or computation time (see Appendix for overview). This work is an attempt to reclaim uncertainties for any deterministic model retrospectively by supplementing it with a vine copula, and hence not affecting the model's accuracy. We favour them for the non-parametric characteristic as well as the (theoretically justified) scalability.

Problem setup We consider a supervised learning setup where $X \in \mathbb{R}^p$ is a random variable for the features, and $Y \in \mathbb{R}$ is the target variable. We assume a process $y = f(x, \epsilon)$ to be responsible for generating a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ from which we observe realizations. We are interested in learning a prediction model $f_\Theta : \mathcal{X} \to \mathcal{Y}$, in particular, we chose a (deep) neural network (DNN), with its weights considered as parameters we denote Θ, to approximate f. The DNN can learn a model that approximates the conditional mean well, i.e $\hat{f}_\Theta \in \arg\min_f \ell(\widehat{f_\Theta}(x), y)$ by setting the loss ℓ to mean-squared error.

Uncertainty intervals Typically, a NN does not provide uncertainty quantities to state confidence in its outputs. The predictive uncertainty, as the errors of a model, are associated to two sources: 1) an *epistemic* one - the data provided in the training is not complete, i.e. certain input regions have not been covered or the model lacks capacity to approximate the true function (lack of knowledge, hence reducible); And 2) an *aleatoric* one, resulting from the inherent noise in the data (the ϵ in the generative process), hence, an irreducible quantity, but, can be accounted for. To use NNs for applications in any domain, it is essential to provide uncertainty statements related to both of these sources. Within standard statistical models, one could relate the epistemic uncertainty to *confidence intervals* (CIs) which evaluate the $Pr(f(x)|\widehat{f}(x))$, whereas the aleatoric, noise uncertainty is captured

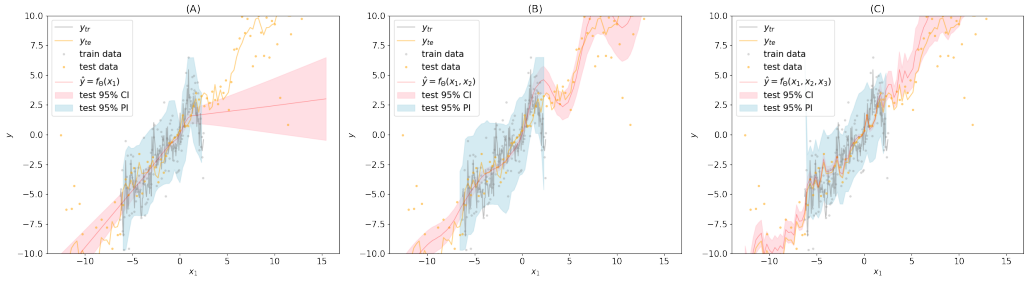

Figure 1: Confidence and Prediction intervals by VCNN in a toy regression problem. Details in text.

by *prediction intervals* (PIs) $Pr(y|\widehat{f}(x))$. Both of these uncertainties contribute to a variability in the predictions which can be expressed through entropy or variance (Depeweg et al., 2018). In this work we will use the variance σ^2, as a quantity of interest unifying the two sources of uncertainty, and represent the predictive uncertainty per data point by its lower bound $\min_{i \in N}(\sigma^2_{model}{}^i, \sigma^2_{data}{}^i)$ and its upper bound $\max_{i \in N}(\sigma^2_{model}{}^i, \sigma^2_{data}{}^i)$. Our goal is to propose a tool that recovers estimates for *both the confidence and the prediction intervals* of a DNN. This natural decomposition will further allow for them to be combined and used in applications as deemed most suitable[1].

In Figure 1 we present our uncertainty estimates. The observations are following a wiggly sequential data where the true generative process is a function $Y = f(X_1, X_2, X_3, \epsilon)$ which we try to predict with a DNN of 2 hidden layers with 50 neurons each. Moving from left to right, plots in Figure 1 show network predictions given: A) only observations from X_1 as input, i.e. $\hat{y} = \widehat{f_\Theta}(x_1)$, then $\hat{y} = \widehat{f_\Theta}(x_1, x_2)$ in B, and $\hat{y} = \widehat{f_\Theta}(x_1, x_2, x_3)$ in C. By including relevant variables (more knowledge; more data), the fitted model improves in each consecutive plot, visible also through its CI and PI. This toy example shows the sensitivity of our estimates to both the sources: As the DNN gets more information, the prediction of the model improves, its CIs improve as well (narrower in train region, wider outside), while the PIs capture the noisiness of the data (when variables are omitted, such as in A and B, this translates into more stochasticity, i.e as aleatoric noise). Moreover, from Figure 1, we also see why considering both sources is important: in the out of distribution region the confidence intervals prevail, while in the nosier regions, the prediction intervals envelope the CI. It is also important to note that although there are many recent developments on the topic of uncertainty of deep models, only a few consider the two sources, and even fewer have a unifying framework for both of them, which is why in domain science and industry that require both, Bayesian NNs prevail. Our contributions are as follows:

- A methodology for recovering uncertainties based on vine copulas (subsection 2.1),
- An implementation of plug-in estimates (section 2),
- Empirical evaluation on real-world datasets (section 3).

2 From Copulas to Model Uncertainties

Next we present how we envision using the natural properties of a vine-copula to account for the uncertainties of a deep model. We assume that a DNN, $\widehat{f_\Theta}$, has already been trained until convergence for a specific task. For efficiency and simplicity of the method, we use the last hidden layer of the network - h, (where $h(x) \in \mathbb{R}^d, d << p$) as a proxy for the overall network parameters.

Copulas and Vines According to Sklar's theorem (Sklar, 1959), the joint density of any bivariate random vector (X_1, X_2), can be expressed as $f(x_1, x_2) = f_1(x_1)f_2(x_2)c(F_1(x_1), F_2(x_2))$, where f_i are the marginal densities, F_i the marginal distributions, and c the copula density. That is, any bivariate density is uniquely described by the product of its marginal densities and a copula density,

[1]Different domains have different ways of adding up or considering the two uncertainties in final applications (Der Kiureghian & Ditlevsen, 2009)

which is interpreted as the dependence structure. As a benefit of such factorisation, by taking the logarithm on both sides, one could straightforwardly estimate the joint density in two steps, first for the marginal distributions, and then for the copula. Hence, copulas provide means to flexibly specify the marginal and joint distribution of variables. For further details on copulas, please refer to Aas et al. (2009); Joe et al. (2010). There exist many parametric representations through different copula families, however, to leverage even more flexibility, in this paper we focus on kernel-based nonparametric copulas of Geenens et al. (2017). To be able to fit densities of more then two variables, we make use of the pair copula constructions, namely vines; hierarchical models, constructed from cascades of bivariate copula blocks (Nagler et al., 2017). According to Joe (1997); Bedford & Cooke (2002), any d dimensional copula density can be decomposed into a product of $\frac{d(d-1)}{2}$ bivariate (conditional) copula densities. Although such factorization may not be unique, it can be organized as in a graphical model, as a sequence of $d-1$ nested trees, called *regular vine*, *R-vine*, or simply *vine*. We denote a tree as $T_m = (V_m, E_m)$ with V_m and E_m the sets of nodes and edges of tree m for $m = 1, \ldots, d-1$. Each edge e is associated to a bivariate copula. A full example of a vine copula and its decomposition is given in the Appendix. In practice, in order to construct a vine, one has to choose two components 1) the structure, the set of trees $T_m = (V_m, E_m)$ for $m = 1, \ldots, d-1$ and 2) the pair-copulas, the models for $c_{j_e,k_e|D_e}$ for $e \in E_m$ and $m = 1, \ldots, d-1$. Corresponding algorithms exist for both of those steps and in the rest of the paper we assume consistency of the vine copula estimators. We rely on the implementation by Nagler & Vatter (2018a).

2.1 VINE-COPULA UNCERTAINTY ESTIMATES

Simulation-based Confidence Intervals Besides being a flexible density estimation method, (vine) copulas additionally have generative properties (Dissmann et al., 2013; Tagasovska et al., 2019). This has inspired us to use vine copulas as an aid to bootstrap the network parameters, yielding confidence intervals for the network predictions via simulations. Simulation-based approaches for confidence intervals have previously been used in the literature for smoothing splines (Ruppert et al., 2003).

Following a similar approach to Ruppert et al. (2003), we consider the true function f over L locations in x, $\mathbf{l} = \{l_1, l_2, \ldots l_L\}$, $f_\mathbf{l}$ denoting the vector of evaluations of f at each of those locations, and the corresponding estimate of the true function by the trained DNN as $\widehat{f_{\Theta\mathbf{l}}}$. The difference between the true function and our unbiased[2] estimator is given by $\widehat{f_{\Theta\mathbf{l}}} - f_\mathbf{l} = H_\mathbf{l}\left[\widehat{\Theta} - \Theta\right] = \mathbf{U}$, where $H_\mathbf{l}$ is the evaluation of the DNN at the locations $\mathbf{l}$, and the matrix represents the variation in the estimated network parameters. The distribution of the variation is unknown, and we aim to approximate it by simulation. A $100(1-\alpha)\%$ simultaneous confidence interval is $\widehat{f}_l \pm q_{1-\alpha}\widehat{\sigma}[(\widehat{f}(\mathbf{l}_j) - f(\mathbf{l}_j)]_{j=1}^L$ where σ denotes a standard deviation and $q_{1-\alpha}$ is the $1-\alpha$ quantile of the random variable $\sup_{x\in X}\left|\frac{\widehat{f}(x)-f(x)}{\widehat{\sigma}(\hat{f}(x))-f(x)}\right| \approx \max_{1\le j\le L}\left|\frac{\mathbf{U}_j}{\widehat{\sigma}\hat{f}(\mathbf{l}_j)-f(\mathbf{l}_j))}\right|$. The sup refers to the supremum or the least upper bound; that is the least value of $\mathcal{X}$ the set of all values of which we observed subset x, which is greater than all other values in the subset. Commonly, this is the maximum value of the subset, as indicated by the right-hand side of the equation; hence we want the maximum (absolute) value of the ratio over all values in $\mathbf{l}$. The fractions in both sides of the equation correspond to the standardized deviation between the true function and the model estimate, and we consider the maximum absolute standardized deviation. While we do not have access to the distribution of the deviations, we only need its quantiles that we can approximate via simulations. Herein, we exploit the generative nature of the copulas whereby we "bootstrap" the trained DNN with the vine copula fitted over the embeddings ξ (the representation of a data point x in the last hidden layer) and target y. For each simulation we find the maximum deviation of the (re)fitted functions from the true function over the grid of $\mathbf{l}$ values we are considering. As a last step we find the critical value to scale the standard errors SE such that they yield the simultaneous interval; we calculate the critical value r for a 95% simultaneous confidence interval/band using the empirical quantile of the ranked standard errors. Finally we recover the upper and lower epistemic bounds obtained as $y^i_{Ue}/y^i_{Le} = \hat{f}_\theta(x^i) \pm r * SE(\mathbf{U}^i_\mathbf{l})$, respectively.

Conditional-quantile Prediction Intervals According to Nagler & Vatter (2018b), conditional expectations can be replaced by unconditional, and such property can be used to leverage copulas for

[2]We consider a network of sufficient capacity such that it can approximate any function (considering NNs as universal approximators (Hornik, 1991)), hence, we exclude the model to true function bias and consider only the variability of the models parameters with respect to the data.

Table 1: PICP and MPIW values for VCNN and baselines for the two real-world datasets.

BiLSTM	Train		Test	
	PICP	MPIW	PICP	MPIW
VCNN	.97	5.71	.85	**5.84**
MC-Dropout+N	.94	6.97	**.86**	6.88

DenseNet	Train		Test	
	PICP	MPIW	PICP	MPIW
MC-Dropout+N	.99 $\pm$.00	468.03 $\pm$49.	.99 $\pm$.00	514.15 $\pm$72.
Ensemble+N	.99 $\pm$.00	433.25 $\pm$27.	.98 $\pm$.00	468.95 $\pm$31.
VCNN	$\mathbf{.97 \pm .01}$	$\mathbf{226.8 \pm 4.4}$	$\mathbf{.95 \pm .00}$	$\mathbf{191.33 \pm 3.0}$

solving equations. This allows for new estimators of various nature. Since in the case of aleatoric uncertainty, we are interested in estimating conditional distribution of the target variable given the inputs $F(Y|X)$, which can be captured by conditional quantiles of interest (Tagasovska & Lopez-Paz, 2019), we rely on solving estimating equations when the identifying function ψ_θ (kept notation from Nagler & Vatter (2018b) for consistency) is suitable for a quantile regression. That is, we let $\theta_{0,\tau} = F^{-1}_{Y|X}(\tau|X = x)$ be the conditional $\tau-$quantile for $\tau \in (0, 1)$. Using this knowledge, and the embeddings ξ, we are now able to compute the upper and lower bound of a required $(1 - \alpha)$ prediction interval, by obtaining $y^i_{Ua} = F^{-1}_{Y|\xi}((1 - \frac{\alpha}{2})|\xi^i)$ and $y^i_{La} = F^{-1}_{Y|\xi}(\frac{\alpha}{2}|\xi^i)$ respectively by solving the conditional expectation using vine copulas. In practice, we use the recent implementation from the `eecop` package (Nagler & Vatter, 2020). To the best of our knowledge, this is the first attempt to use copula-based simulations to obtain confidence estimates for DNNs.

3 Experiments

To empirically evaluate our method we use two real-world datasets: Datalakes - estimating the surface temperature of Lake Geneva based on sensor measurements, hydrological simulations and satellite imagery, and an AirBnB apartment price forecasting dataset for Barcelona. In both cases we wish to estimate the predictive uncertainty, hence, we incorporate both epistemic and aleatoric uncertainty sources. As baselines we use the Ensembles and MC-dropout; both adjusted to capture the two types of uncertainties. We compare by the Prediction Interval Coverage Probability (PICP): how many of the observations are captured inside the estimated interval; higher is better; and the Mean Prediction Interval Width (MPIW) where lower values are better.

Datalakes Datalakes[3] tackles certain data-driven problems such as estimating lake surface temperature given a range of sensory, satellite imagery, and simulation-based datasets. One such dataset is of Lake Geneva where a sparse hourly dataset is collected between years 2018 to 2020. Given the temporal nature of the observations, the model that provided the best prediction was a bidirectional (Bi)(Schuster & Paliwal, 1997) long short-term memory (LSTM)(Hochreiter & Schmidhuber, 1997) network. Originally, this method uses MC-dropout in addition with a negative log-likelihood loss as in Kendall & Gal (2017). We replace those uncertainties with the vine based ones and we present our results in section 3. Comparisons with ensemble methods were omitted due to both computational restrictions and MC-dropouts method being a comparable proxy. From section 3 we see that the VCNN achieves on-par estimates for PICP with a (significantly) narrower MPIW. The measuring unit is Celsius degrees.

Room price forecasting Based on a publicly available data from the Inside Airbnb platform Cox (2019), for Bacelona, we followed a regression problem proposed in (Brando et al., 2019). The aim is to predict the price per night for $36,367$ flats using data from April 2018 to March 2019. The architecture used is a DenseNet. From Table 3 and Figure 4 we notice that the plug-in estimates in VCNN outperform the ensembles and MC-dropout, the intervals are properly calibrated (95%) and they are tighter by at least a half. The measurement unit here is euros.

4 Conclusion

We provide a method for quantifying trustworthiness of deep models, existing and new, that we anticipate can bring a great value for real-world scenarios which require critical decision making. This is particularly attractive given the increasing training costs (financial and environmental) with deeper and wider NNs. There are multiple directions for extensions: We are developing out-of distribution detection method for classifiers (toy example in Appendix) and, we expect VC to benefit uncertainties in other tasks, such as segmentation or reinforcement learning.

[3] `https://www.datalakes-eawag.ch`

REFERENCES

Kjersti Aas, Claudia Czado, Arnoldo Frigessi, and Henrik Bakken. Pair-copula constructions of multiple dependence. *Insurance: Mathematics and economics*, 44(2):182–198, 2009.

Tim Bedford and Roger M. Cooke. Vines – A New Graphical Model for Dependent Random Variables. *The Annals of Statistics*, 30(4):1031–1068, 2002.

Axel Brando, Jose A Rodríguez-Serrano, Mauricio Ciprian, Roberto Maestre, and Jordi Vitrià. Uncertainty modelling in deep networks: Forecasting short and noisy series. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 325–340. Springer, 2018.

Axel Brando, Jose A Rodríguez-Serrano, Jordi Vitria, and Alberto Rubio. Modelling heterogeneous distributions with an uncountable mixture of asymmetric laplacians. *NeurIPS*, 2019.

Axel Brando, Damia Torres, Jose A Rodriguez-Serrano, and Jordi Vitria. Building uncertainty models on top of black-box predictive apis. *IEEE Access*, 8:121344–121356, 2020.

Murray Cox. Inside airbnb: adding data to the debate. *Inside Airbnb [Internet].[cited 16 May 2019]. Available: http://insideairbnb.com*, 2019.

Stefan Depeweg, Jose-Miguel Hernandez-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning. In *ICML*, 2018.

Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural Safety*, 2009.

J. Dissmann, Eike Christian Brechmann, Claudia Czado, Dorota Kurowicka, J Dißmann, Eike Christian Brechmann, Claudia Czado, and Dorota Kurowicka. Selecting and estimating regular vine copulae and application to financial returns. *Computational Statistics & Data Analysis*, 59:52–69, March 2013.

Y Gal and Z Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.

Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

Gery Geenens, Arthur Charpentier, and Davy Paindaveine. Probit transformation for nonparametric kernel estimation of the copula density. *Bernoulli*, 23(3):1848–1873, 2017.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. *ICML*, 2017.

Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew M Dai, and Dustin Tran. Training independent subnetworks for robust prediction. *arXiv preprint arXiv:2010.06610*, 2020.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *ICML*, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9 8: 1735–80, 1997.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(91)90009-T. URL https://www.sciencedirect.com/science/article/pii/089360809190009T.

Harry Joe. *Multivariate Models and Dependence Concepts*. Chapman & Hall/CRC, 1997.

Harry Joe, Haijun Li, and Aristidis K. Nikoloulopoulos. Tail dependence functions and vine copulas. *Journal of Multivariate Analysis*, 101(1):252–270, January 2010.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *NeurIPS*, 2017.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, 2017.

Thomas Nagler and Thibault Vatter. *kde1d: Univariate Kernel Density Estimation*, 2018a. R package version 0.2.1.

Thomas Nagler and Thibault Vatter. Solving estimating equations with copulas. *arXiv preprint arXiv:1801.10576*, 2018b.

Thomas Nagler and Thibault Vatter. *eecop: an R Package to Solve Estimating Equations with Copulas*, 2020.

Thomas Nagler, Christian Schellhase, and Claudia Czado. Nonparametric estimation of simplified vine copula models: comparison of methods. *Dependence Modeling*, 5(1):99–120, 2017.

David Ruppert, Matt P Wand, and Raymond J Carroll. *Semiparametric regression*. Number 12. Cambridge university press, 2003.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

A. Sklar. Fonctions de Répartition à n Dimensions et Leurs Marges. *Publications de L'Institut de Statistique de L'Université de Paris*, 8:229–231, 1959.

Natasa Tagasovska and David Lopez-Paz. Single-model uncertainties for deep learning. *NeurIPS*, 2019.

Natasa Tagasovska, Damien Ackerer, and Thibault Vatter. Copulas as high-dimensional generative models: Vine copula autoencoders. *NeurIPS 2019*, 2019.

A POSITIONING OF VCNN TO RELATED WORK

In Table 2 we compare the most popular approaches for uncertainty quantification which are closest to our setup, that is capturing both aleatoric and epistemic uncertainties on the same scale. This is of course not an exhaustive list but more works will be included for the next iteration of the manuscript. We only use Table 2 to position the unique properties of having the VC estimates. VC allows an elegant way to extend any trained network retrospectively to quantify both epistemic and aleatoric uncertainty with performance on-par or better than existing baselines. Given the undesired training costs of deeper and wider NNs as stated before, VC strikes as a viable solution.

Table 2: Overview of popular solutions that provide model and epistemic uncertainty for deep models within a unified framework.

	No custom loss	**No custom architecture**	**Single model**	**Task agnostic**	**Scalable**	**Retrospective uncertainties**
MC-dropout	✓	✗	✓	✓	✓	✗
Ensembles	✓	✓	✓	✗	✗	✗
Bayesian NNs	✓	✗	✓	✓	✗	✗
Vine-Copula NNs	✓	✓	✓	✓	✓[4]	✓

B ALGORITHMIC LISTING FOR VCNN

Algorithm 1: Vine-copula DNN uncertainty estimates.

Inputs: DNN f_Θ, train samples $\{x_i, y_i\}_{i=1}^N$, test samples $\{x'_j\}_{j=1}^L$;

Step 1. Obtain the embeddings from the last dense layer $\xi = \{h(x_i)\}_{i=1}^N \in \mathbb{R}^d$;
Step 2. Fit a vine copula with the embeddings and target variable, $V(\xi_1, \xi_2, \dots \xi_d, y)$. ;
Epistemic uncertainty ;
Step 3. For S repetitions:

1. Sample random observations from the vine V, $\{\xi^*, y^*\}$.
2. Re-train the last hidden and the output layer of the DNN with $\{\xi^*, y^*\}$.
3. Save the bootstraped S heads[a] of the network.

Step 4. Compute the confidence interval (CI) bounds: $y_{Le}(x'_j)$ and $y_{Ue}(x'_j)$. ;
Aleatoric uncertainty ;
Step 5. Solve for the required quantiles $F^{-1}_{1-\frac{\alpha}{2}}(Y|\xi(x'_j))$ and $F^{-1}_{\frac{\alpha}{2}}(Y|\xi(x'_j))$ using vine V;
Step 6. Compute the prediction interval (PI) bounds: $y_{La}(x'_j)$ and $y_{Ua}(x'_j)$. ;
Outputs S-heads, V vine copula, confidence and prediction intervals for x'_j.

[a] We use the term "heads" here as our bootstrapped layers are similar in architecture to the concept of multi-head networks.

C EXAMPLE OF A VINE COPULA

A sequence is a vine if it satisfies the set of conditions which guarantee that the decomposition represents a *valid joint density*: i) T_1 is a tree with nodes $V_1 = \{1, \dots, d\}$ and edges E_1 ii) For $m \geq 2$, T_m is a tree with nodes $V_m = E_{m-1}$ and edges E_m iii) Whenever two nodes in $T_m + 1$ are joined by an edge, the corresponding edges in T_m must share a common node. The corresponding tree sequence is the *structure* of the vine. Each edge e is associated to a bivariate copula $c_{j_e,k_e|D_e}$, with the set $D_e \in \{1, \cdots, d\}$ and the indices $j_e, k_e \in \{1, \cdots, d\}$ forming respectively its *conditioning set* and the *conditioned set*. Finally, the joint copula density can be written as the product of all pair-copula densities $c(u_1, \cdots, u_d) = \prod_{m=1}^{d-1} \prod_{e \in E_m} c_{j_e,k_e|D_e}(u_{j_e|D_e}, u_{k_e|D_e})$ where $u_{j_e|D_e} = \mathbb{P}\left[U_{j_e} \leq u_{j_e} \mid \boldsymbol{U}_{D_e} = \boldsymbol{u}_{D_e}\right]$ and similarly for $u_{j_e|D_e}$, with $\boldsymbol{U}_{D_e} = \boldsymbol{u}_{D_e}$ understood

as component-wise equality for all components of $(U_1, \ldots, U_d)$ and $(u_1, \ldots, u_d)$ included in the conditioning set D_e.

For self contaned manuscript, we borrow form Tagasovska et al. (2019), a full example of an R-vine for a 5 dimensional density.

The density of a PCC corresponding to the tree sequence in Figure 2 is

$$c = c_{1,2}\, c_{1,3}\, c_{3,4}\, c_{3,5}\, c_{2,3|1}\, c_{1,4|3}\, c_{1,5|3} c_{2,4|1,3}\, c_{4,5|1,3}\, c_{2,5|1,3,4}, \tag{1}$$

where the colors correspond to the edges E_1, E_2, E_3, E_4.

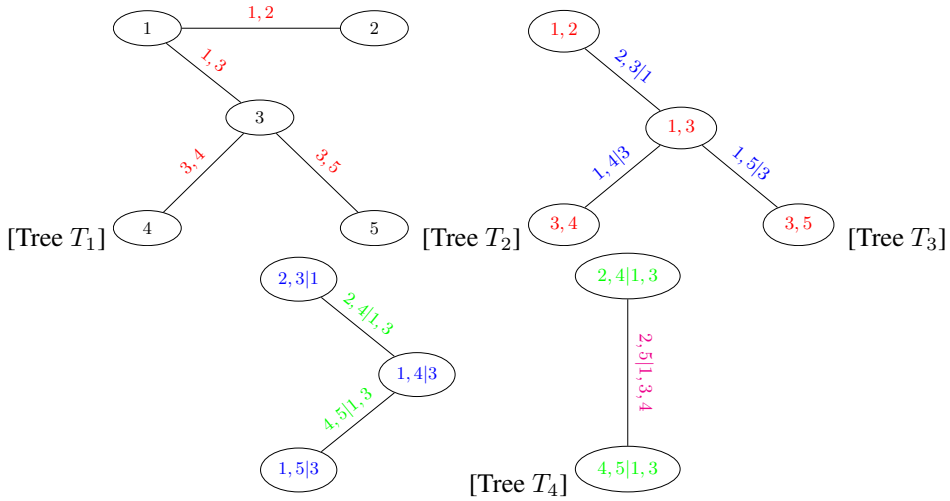

Figure 2: A vine tree sequence: the numbers represent the variables, x, y the bivariate distribution of x and y, and $x, y|z$ the bivariate distribution of x and y conditional on z. Each edge corresponds to a bivariate pair-copula in the PCC.

D Details of the toy experiment in Figure 1

The toy example in Figure 1 is generated as: $Y_{train} = X_1 + X_2 + X_3 + \epsilon$, with $X1 = \mathcal{U}[-2\pi, 2\pi]$, $X_2 = sin(2X_1)$, $X_3 \sim sin(X_1^2)$, with $\epsilon \sim \mathcal{N}(0, 0.2)$. For the test data $Y_{train} = X_1 + X_2 + X_3 + x_1\epsilon$, with $X1 \sim \mathcal{U}[-4\pi, 5\pi]$, $X_2 = sin(X_1)$, $X_3 \sim sin(X_1^2)$, with $\epsilon \sim \mathcal{N}(0, 0.5)$. Furthermore, the train data consists of 280 samples, and the test data of 100. The network was trained for 100 epochs for the three presented cases, using Adam optimizer with default PyTorch parameters. We use S=30 to get the confidence intervals and $\tau_{low} = 0.025$ and $\tau_{high} = 0.975$ for the prediciton intervals.

E More toy experiments

In Figure 3 we provide further toy examples of VCNN. In the first row a one dimensional input bi-modal regression task (Tagasovska & Lopez-Paz, 2019), while the second row is a classification task for the two dimensional input, using moons dataset. We show the results from a trained DNN (predictions and class probabilities respectively) in the first column, the epistemic uncertainty estimate in the second, and the aleatoric uncertainty estimate in the third. For the classification task, since the input is two dimensional, we present the uncertainty scores through a color range, depicting the distance between the upper and lower bounds of the corresponding CI or PI.

We consider the classification results encouraging, since both the epistemic uncertainty grows further away from the train data, and the aleatoric uncertainty is high only in the region where there is an overlap between the two classes, as one would expect.

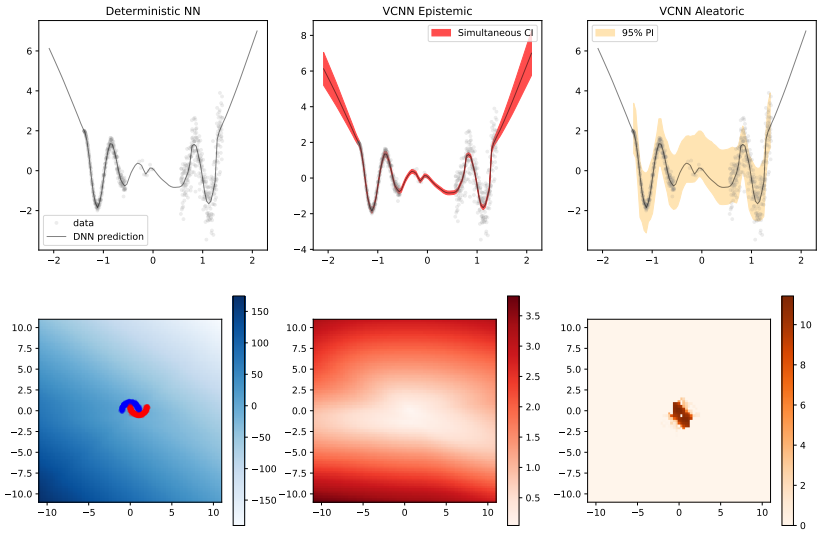

Figure 3: VCNN for toy regression and classification scenarios.

F PRACTICAL CONSIDERATIONS FOR THE VINE COPULAS

Although the vine structure and the copula family could be considered as hyperparameters, empirically we observed that the best results are obtained when we use nonparametric family with a low multiplier for the kernel width, i.e. 0.1, for both the confidence and prediction intervals. For computational purposes, we also chose to truncate the vine, that is to fit only the first 2 - 5 trees in the vine and consider all the subsequent ones as independent copulas. These factors can indeed appear as more important for other datasets and we are conducting multiple ablation studies to explore this further.

Complexity The complexity for fitting the vine copulas as currently implemented is $\approx O(n \times dim \times trunc\_lvl)$ for estimation/sampling algorithms, both involving a double loop over dimension/truncation level with an internal step scaling linearly with the sample size. Due to this linear scaling in the number of samples and dimension, we found it useful to randomly subsample the train data and use trancated vines. Further optimizations were outside of the scope of the current work.

G DETAILS BILSTM IMPLEMENTATION - DATALAKES

Pre-RNN fully connected layers: 1 layer $\times$ 32 units, LeakyReLU ($\alpha = 0.2$)
RNN layers: 3 LSTM layers $\times$ 32 units each and bidirectional ($\times$ 2)
Dimensionality of last hidden layer: 64
Train and test data dimensions: 214,689 and 186,129 samples of 18 dimensions
Dropout level: 0.3
Rough estimate of train time: 6 days
VC fit time: 2634.29 sec;
VC inference time per data point: 2.1 sec
number of VC bootstraps: 15
Framework: Tensorflow 2.4.1

Due to the longer training times of the BiLSTM model and limited time and resources, we were not able to obtain standard variations of the results. Same reasoning goes for not recommending and including ensembles.

H DETAILS DENSENET IMPLEMENTATION - AIRBNB

The architecture used for the main DenseNet is the same that the proposed in Brando et al. (2019). Additionally, the different models proposed in this article satisfies the following parameters and results:

Number of layers: 6 dense layers
Number of neurons per layer: $120, 120, 60, 60, 10$ and 1
Training time: 30.4 secs
Activation types: ReLU activation for hidden layers
Dimensionality of last hidden layer used by the VC: 10
Training , validation and test data dimensions: 29078, 3634 and 3633 respectively.

VC estimates train time: 1861.09 ± 115.91 secs.
Time of the VC predicting each point: $1.53 \pm 3 \cdot 10^{-3}$ secs
Number of VC bootstraps: 10
Framework: Tensorflow 2.4.1

I ADDITIONAL RESULTS ON THE AIRBNB DATASET

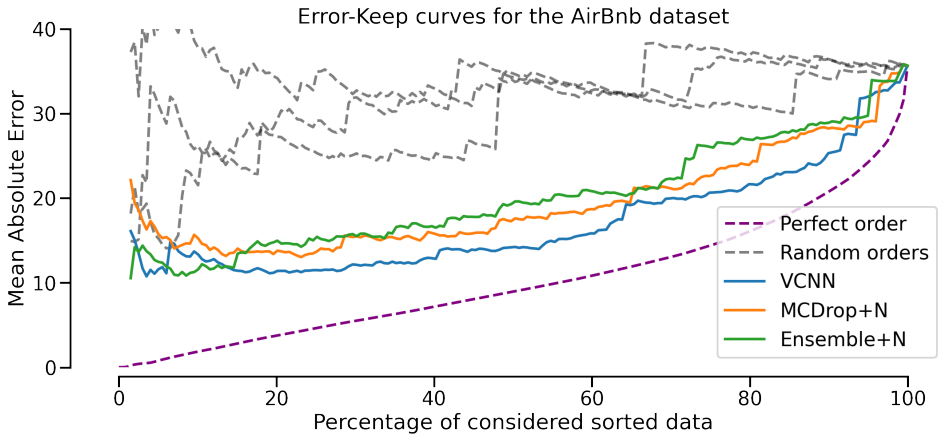

Figure 4: An error-keep curve - the mean cumulative error for the most confident points for the uncertainty scores obtained by baselines and VCNN (see Brando et al. (2018; 2020)).

Given a trained DenseNet, we can extract two conclusions from Figure 4: if we sort our prediction Mean Absolute Errors by the VCNN uncertainty scores, for the first 80% of the data with higher confidence values, our cumulative error would be 18 euros, otherwise, for the 90% we would have approximately 30 euros margin of error. As it is shown in that figure, VCNN obtains an error-keep curve almost always closer to the perfect curve which is not the case for MC Dropout and Ensembles.