Hybrid Bernstein Normalizing Flows for Flexible Multivariate Density Regression with Interpretable Marginals

Marcel Arpogaus[1] **Thomas Kneib**[2] **Thomas Nagler**[3] **David Rügamer**[3,4]

[1]Institute for Applied Research , HTWG - University of Applied Sciences , Konstanz, GERMANY
[2]Chair of Statistics and Campus Institute Data Science (CIDAS) , University of Göttingen , Göttingen, GERMANY
[3]Department of Statistics , LMU Munich, Munich, GERMANY
[4]Munich Center for Machine Learning , LMU Munich, Munich, GERMANY

Abstract

Density regression models allow a comprehensive understanding of data by modeling the complete conditional probability distribution. While flexible estimation approaches such as normalizing flows (NFs) work particularly well in multiple dimensions, interpreting the input-output relationship of such models is often difficult, due to the black-box character of deep learning models. In contrast, existing statistical methods for multivariate outcomes such as multivariate conditional transformation models (MCTMs) are restricted in flexibility and are often not expressive enough to represent complex multivariate probability distributions. In this paper, we combine MCTMs with state-of-the-art and autoregressive NFs to leverage the transparency of MCTMs for modeling interpretable feature effects on the marginal distributions in the first step and the flexibility of neural-network-based NF techniques to account for complex and non-linear relationships in the joint data distribution. We demonstrate our method's versatility in various numerical experiments and compare it with MCTMs and other NF models on both simulated and real-world data.

1 INTRODUCTION

Many real-world regression problems involve modeling of complex conditional distributions of an outcome Y given an input vector $\mathbf{x} = [x_1, \ldots, x_D]^\top$. In such instances, a mere prediction of the mean would be insufficient to capture the aleatoric uncertainty inherent in the data-generating process and a comprehensive understanding requires estimating the conditional distribution $Y|\mathbf{x}$ [Kneib et al., 2021]. This becomes more challenging for multivariate outcomes $\mathbf{Y} \in \mathbb{R}^D$, both in terms of estimation and model interpretation. While normalizing flow (NF) [Papamakarios et al., 2017] models can represent such complex conditional multivariate distributions, their deep neural network architectures often hinder interpretability. Statistical methods for flexible density estimation in high dimensions have largely focused on graphical models, often involving copulas [e.g., Liu et al., 2011, Bauer and Czado, 2016, Nagler and Czado, 2016]. A notable exception is the MCTM, a transformation model approach [Hothorn et al., 2018] that shares similarities with NFs and models the joint relationship similar to copulas. While less flexible than NFs, MCTMs preserve the interpretability of the feature-outcome relationship.

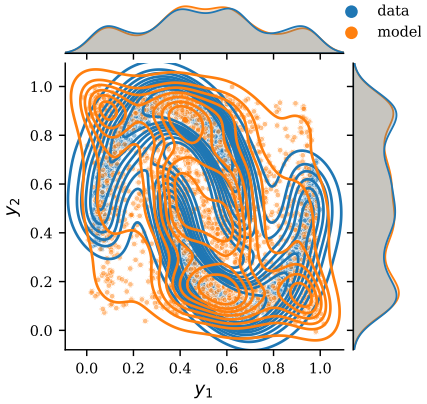


Figure 1: An MCTM (orange) applied to a complex data distribution (blue). The model captures marginals well but fails to capture the dependence structure.

Our Contribution To bridge the gap between flexible black-box and rigid but interpretable statistical models, we propose a hybrid approach combining the transparency of MCTMs with the flexibility of NFs. We thereby provide a method that can be applied if the understanding of feature effects on each response variable is indispensable. At the same time, our method allows complex modeling of the dependency structure of the outcome dimensions using the

idea of masking from autoregressive flows. We evaluate its effectiveness against MCTMs and other density estimation models on simulated and real-world datasets.

Notation We refer to random variables with a capital Latin letter, e.g., Y, and their realizations (observed or sampled) with a lowercase Latin letter, e.g. y. The corresponding cumulative distribution function (CDF) is referred to as F_Y and the probability density function (PDF) as f_Y. For sets and vectors, we use bold-faced variables, e.g. $\mathbf{Y} = \{Y_1, \ldots, Y_J\}$. For individual parameters, optimized during the training procedure we use lowercase Greek letters, e.g. ϑ, and bold-faced Greek letters for sets or vectors of parameters, e.g. $\boldsymbol{\vartheta} = (\vartheta_1, \ldots, \vartheta_M)^\top$. Bold capital Greek letters indicate tuples, e.g. $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \alpha, \beta)$.

2 BACKGROUND AND RELATED WORK

Several methods have emerged for modeling complex conditional distributions. Among these methods, transformation-based models have proven to be highly flexible and have been extensively used since their introduction by Box and Cox [1964].

2.1 PROBABILITY TRANSFORMATION

Transformation models are based on the *probability-transformation theorem*, using a bijective and continuously differentiable *transformation function* h to map a complex distribution F_Y to a simpler one F_Z (often standard normal) [Kneib et al., 2021] via $F_Y = F_Z(h(y))$. This allows density estimation without restrictive shape assumptions [Hothorn et al., 2018]. The density f_Y is calculated from the base density f_Z by

$$f_Y(y) = f_Z\left(h(y)\right) \left|\det \nabla h(y)\right|, \tag{1}$$

where the absolute Jacobian determinant $|\det \nabla h(y)|$ reflects the change in density induced by the transformation. To sample from F_Y, we can draw samples z from F_Z and apply the inverse transformation h^{-1} to obtain y.

2.2 NORMALIZING FLOWS AND TRANSFORMATION MODELS

NFs apply the probability-transformation theorem using a series of K simple bijective transformation functions h_k to form more expressive transformations $h(z) = h_K \circ h_{K-1} \circ \cdots \circ h_1(z)$ [see, e.g. Papamakarios et al., 2021, for a comprehensive review]. Other methods use a single flexible transformation like sum-of-squares polynomials [Jaini et al., 2019] or splines [Durkan et al., 2019]. Conditional transformation models (CTMs) also focus on single transformation function, which, however, can be easily interpreted [see, e.g., Hothorn et al., 2018]. This idea has also recently been combined with neural-networks [Baumann et al., 2021, Sick et al., 2021, Arpogaus et al., 2023, Kook et al., 2024].

A fundamental difference between NFs and CTMs is the definition of the transformation direction [Kook et al., 2024]: NFs transform a simple base density $f_Z(z)$ into a more complex target density $f_Y(y)$ through a series of transformations. CTMs do it vice-versa, using only a single transformation function, often utilizing flexible Bernstein polynomials (see Appendix A.1 for details).

The previous concepts can be further extended to multivariate objects $\mathbf{y} = (y_1, \ldots, y_J)^\top$ and can also be applied conditional on U features $\mathbf{x} = (x_1, \ldots, x_U)^\top$ as described in the following.

2.3 MULTIVARIATE AND CONDITIONAL MODELS

To model a potentially multivariate conditional distribution $F_{\mathbf{Y}|\mathbf{X}=\mathbf{x}}$, we define $z_j = h(y_j, \boldsymbol{\theta}_j)$ with $\boldsymbol{\theta}_j = c_j(\mathbf{y}, \mathbf{x})$ where h is a bijective *transformation function* for the element y_j with parameters $\boldsymbol{\theta}_j$ obtained from the *conditioner* c_j [Papamakarios et al., 2021]. The latter can be an arbitrarily complex function that depends on the features $\mathbf{x}$ and, in the multivariate case, on some variables of the response $\mathbf{y}$. The limiting factor here is the computational tractability of the Jacobian in Equation (1).

Multivariate Conditional Transformation Models To make computations tractable and provide a model that is better to interpret, MCTMs define a triangular transformation $H(\boldsymbol{y}, \boldsymbol{\Theta}(\boldsymbol{x})) = \left(h_1(y_1|\boldsymbol{\theta}_1), \ldots, h_J(y_1, \ldots, y_J|\boldsymbol{\theta}_J)\right)^\top$. This Tranformation is characterized by a set of linearly combined marginal basis transformations $\tilde{h}_j(y_j|\boldsymbol{\theta}_j)$, scaled with a $(J \times J)$ triangular coefficient matrix $\boldsymbol{\Lambda}$ to encode structural information: $h_j(y_j, \ldots, y_j|\boldsymbol{\theta}_j) = \lambda_{j,1}\tilde{h}_1(y_1|\boldsymbol{\theta}_1) + \ldots + \lambda_{j,j-1}\tilde{h}_{j-1}(y_{j-1}|\boldsymbol{\theta}_{j-1}) + \tilde{h}_j(y_j|\boldsymbol{\theta}_j)$. The function $\boldsymbol{\Theta} : \boldsymbol{x} \mapsto (\Lambda, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J)$ accounts for feature effects on the parameters [Klein et al., 2019]. Hence, MCTMs can only model linear dependencies between the response variables y_j, making them an inadequate choice for complex joint distributions as shown in Figure 1. This limitation motivates the development of more flexible approaches, such as the hybrid model proposed in this paper.

Normalizing Flows In the NF literature, various approaches are used to deal with computations involving the aforementioned Jacobian. Among these, autoregressive models are one of the most prevalent ones. Due to the autoregressive structure, these NFs yield a triangular Jacobian and thus the determinant simplifies to the product of the Jacobian's diagonal entries [Papamakarios et al., 2021, Kobyzev et al., 2021]. These models factorize a multivariate distribution based on the chain rule of probability. They

are typically implemented by conditioning the transformation h_j of the jth element of $\mathbf{y}$ on all previous elements $\mathbf{y}_{<j} = (y_1, \ldots, y_{j-1})$ or a subset of those. This can be seen as a non-linear generalization of the triangular coefficient matrix Λ used in MCTMs Kobyzev et al. [2021]. There are two widely adopted neural-network architectures for implementing an autoregressive conditioner c_j: masked autoregressive flows (MAFs) [Papamakarios et al., 2017] and coupling flows (CFs) [Dinh et al., 2017].

MAFs generalize this idea by implementing the conditioner with neural networks that incorporate autoregressive constraints through parameter masking, inspired by the masked autoencoder for distribution estimation (MADE) architecture [Germain et al., 2015]. The main advantage is that the parameters can be obtained in one neural network pass $(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J)^\top = (c_1(), c_2(y_1), \ldots, c_J(\mathbf{y}_{<j}))^\top$. Further, if h and the conditioning network are expressive enough, they remain universal approximators to transform between any two distributions [Papamakarios et al., 2021]. In practice, however, the results highly depend on the ordering of the input variables. Because some orderings can be extremely difficult to learn, multiple MAFs layers are often stacked with permutations between.

CFs can provide fast sampling and density evaluation by splitting the response vector $\mathbf{y}$ into two subsets

$$\mathbf{y} = (\underbrace{y_1, \ldots, y_j}_{\mathbf{y}_{\leq j}}, \underbrace{y_{j+1}, \ldots, y_J}_{\mathbf{y}_{>j}})^\top$$

and then applying the transformation h only to one subset conditioned on the other. Typically, these subsets are chosen to contain half of the variables and the result of the transformation on the first subset is then permuted and another transformation is applied to the remaining subset: $\mathbf{z} = (h(\mathbf{y}_{\leq d}, c_1(\mathbf{y}_{>d}, \mathbf{x}), h(\mathbf{y}_{>d}, c_2(\mathbf{y}_{\leq d}, \mathbf{x}))^\top$.

Further background is given in Appendix A.3 and A.4.

3 HYBRID CONDITIONAL MASKED AUTOREGRESSIVE BERNSTEIN FLOWS

To bridge the gap between the limited expressiveness of MCTMs and the black-box nature of general NFs, we propose a combined approach that leverages the strengths of both methodologies. This hybrid approach allows us to capture complex dependencies in the joint distribution while retaining interpretability for the marginal distributions.

3.1 MODEL SPECIFICATION

Let $\mathbf{y} = (y_1, \ldots, y_J)^\top \in \mathbb{R}^J$ be a J-dimensional response vector and $\mathbf{x} = (x_1, \ldots, x_U)^\top \in \mathbb{R}^U$ be a vector of U features. Our goal is to learn the conditional joint density $f_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$.

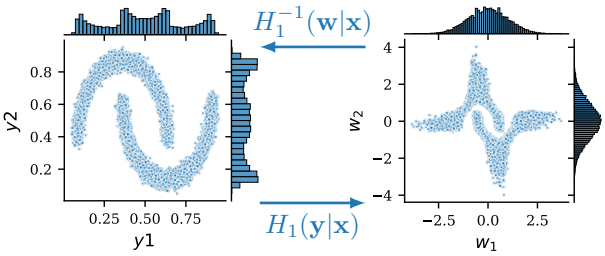


Figure 2: The first transformation H_1 maps the marginals to the base distribution F_Z.

Step 1: Modeling Marginal Distributions First, we model the marginal distributions of each response variable Y_j given $\mathbf{X}$ using a transformation model:

$$H_1(\mathbf{y}, \boldsymbol{\Theta}(\mathbf{x})) = (h_1(y_1, \boldsymbol{\theta}_{1,\mathbf{x}}), \ldots, h_1(y_J, \boldsymbol{\theta}_{J,\mathbf{x}}))^\top = \\ = \mathbf{w} = (w_1, \ldots, w_J)^\top,$$

where each element of $\mathbf{W} = (W_1, \ldots, W_J)^\top$ follows the base distribution F_Z (Figure 2).

Specifically, for each $j = 1, \ldots, J$, we assume:

$$\mathbb{P}(Y_j \leq y_j | \mathbf{X} = \mathbf{x}) = F_{Y_j|\mathbf{X}}(y_j|\mathbf{x}) = F_Z(h_1(y_j, \boldsymbol{\theta}_{j,\mathbf{x}})),$$

where F_Z is a known base distribution and $h_1(\cdot)\colon \mathbb{R} \to \mathbb{R}$ is a strictly monotonic marginal transformation function, with potentially feature-dependent parameters $\boldsymbol{\theta}_{j,\mathbf{x}}$ obtained form the conditioning function $\boldsymbol{\Theta}(\mathbf{x})$.

For interpretability, a (shifted) Bernstein polynomial can be used:

$$h_1(y_j, \boldsymbol{\theta}_{j,\mathbf{x}}) = \boldsymbol{\alpha}_j(y_j)^\top \boldsymbol{\vartheta}_j + \beta_{j,\mathbf{x}},$$
$$\boldsymbol{\alpha}_j(y_j)^\top \boldsymbol{\vartheta}_j = \frac{1}{M+1} \sum_{i=0}^{M} \text{Be}_i(\tilde{y}_j) \vartheta_{ji},$$

where $\text{Be}_m(\tilde{y}_j)$ is the density of a Beta distribution with parameters $i+1$ and $M-i+1$ evaluated at the normalized response $\tilde{y}_j = (y_j - l_j)/(u_j - l_j) \in [0,1]$, with $u_j > l_j$ defining the support of Y_j[1]. The vector $\boldsymbol{\vartheta}_j = (\vartheta_{j0}, \ldots, \vartheta_{jM})^\top$ contains the Bernstein coefficients, which are constrained to be increasing for monotonicity (see Appendix A.1 for details). The shift term $\beta_{j,\mathbf{x}}$ allows the marginal distribution of Y_j to vary with the features. Instead of a feature-dependent shift term, we can also let $\boldsymbol{\vartheta}_j$ depend on $\mathbf{x}$ to change the shape of the transformation h_1 (or combine feature-dependent shift with feature-dependent $\boldsymbol{\vartheta}_j$s).

[1]The transformation is generally unbounded, as we apply linear extrapolation outside the bound of the polynomial, as all our experiments show. But other transformations may be considered if expressiveness on the outer tails is a requirement

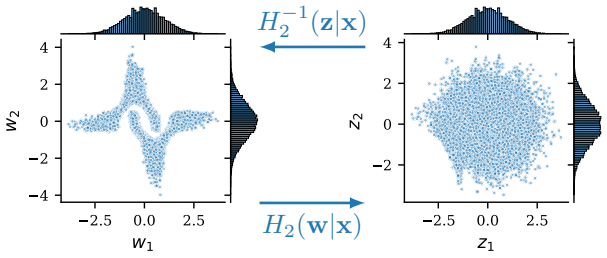


Figure 3: The second transformation H_2 removes the dependency structure.

Step 2: Modeling Dependency Structures We model the dependencies between elements of $\mathbf{W}$ using an autoregressive flow $H_2(\mathbf{w}, \mathbf{\Psi}(\mathbf{w}, \mathbf{x})) : \mathbf{W} \to \mathbf{Z}, \mathbf{w} \mapsto \mathbf{z}$:

$$z_1 = h_2(w_1|\boldsymbol{\psi}_{1,\mathbf{x}}),$$
$$z_j = h_2(w_j|\boldsymbol{\psi}_{j,\mathbf{w}_{<j},\mathbf{x}}), \quad j = 2, \ldots, J,$$

where $h_2(\cdot)$ is an increasing transformation function, with parameters $\boldsymbol{\psi}_{j,\mathbf{w}_{<j},\mathbf{x}}$ depending on previous elements of $\mathbf{w}$ and features $\mathbf{x}$. For h_2, flexible transformation functions like Bernstein polynomials or splines can be used.

We combine ideas from CFs and MAFs to construct the conditioner. Similar to CFs, we do not apply any transformation to w_1 but pass it as a conditional input to the masked neural network that estimates the parameters of the subsequent transformations $\mathbf{\Psi}(\mathbf{w}, \mathbf{x}) = (\boldsymbol{\psi}_{\mathbf{x}}, \boldsymbol{\psi}_{\mathbf{w}_{<1},\mathbf{x}}, \ldots, \boldsymbol{\psi}_{\mathbf{w}_{<J},\mathbf{x}})^\top$. Binary masking, similar to MADE, ensures that the jth parameters do not depend on $\mathbf{z}_{1,\geq j}$. Autoregressive transformations can be stacked if a single transformation is not expressive enough. Through this autoregressive transformation, we aim to "de-correlate" the elements of $\mathbf{W}$ such that $\mathbf{Z}$ follows a multivariate standard normal distribution $\mathcal{N}_J(\mathbf{0}, \mathbf{I})$.

The joint density of the original response vector $\mathbf{Y}$ is then:[2]

$$\begin{aligned} f_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) &= f_Z\left(H\left(\mathbf{y}|\mathbf{x}\right)\right) \left|\nabla H\left(\mathbf{y}|\mathbf{x}\right)\right| \\ &= f_Z\left(H_2\left(H_1\left(\mathbf{y}|\mathbf{x}\right)|\mathbf{x}\right)\right) \\ &\quad \cdot \left|\nabla H_2\left(H_1\left(\mathbf{y}|\mathbf{x}\right)|\mathbf{x}\right) \nabla H_1\left(\mathbf{y}|\mathbf{x}\right)\right|. \end{aligned}$$

Model Training and Inference We train the hybrid model by minimizing the conditional negative log-likelihood of the data $\mathcal{D}$,

$$\begin{aligned} \text{NLL}(\boldsymbol{\omega}) &= -\sum_{(\mathbf{y},\mathbf{x})\in\mathcal{D}} \log f_{\mathbf{Y}|\mathbf{X}}\left(\mathbf{y}|\mathbf{x}\right) \\ &= -\sum_{(\mathbf{y},\mathbf{x})\in\mathcal{D}} \log f_Z\left(H\left(\mathbf{y}|\mathbf{x},\boldsymbol{\omega}\right)\right) \left|\nabla H\left(\mathbf{y}|\mathbf{x},\boldsymbol{\omega}\right)\right|, \end{aligned}$$

[2]We use the short notation $H_1\left(\mathbf{y}, \mathbf{\Theta}\left(\mathbf{x}\right)\right) = H_1(\mathbf{y}|\mathbf{x})$ and $H_2\left(\mathbf{w}, \mathbf{\Psi}\left(\mathbf{w}, \mathbf{x}\right)\right) = H_2(\mathbf{w}|\mathbf{x})$.

to optimize the weights $\boldsymbol{\omega}$ of the conditioning functions $\mathbf{\Theta}_x$ and $\mathbf{\Psi}_x$ parameterizing the normalizing flow $H\left(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega}\right) = H_2 \circ H_1(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega})$.

This involves optimizing the parameters of the marginal transformation functions H_1 and the parameters of the autoregressive flow H_2, which are controlled by masked neural networks. Optimization can be done using any suitable gradient-based optimization algorithm, such as Adam [Kingma and Ba, 2017]. Once the model is trained, we can sample from the joint distribution of $\mathbf{Y}|\mathbf{X}$ by:

1. Sample $\mathbf{z}$ from the base distribution F_Z.
2. Apply the inverse autoregressive flow to obtain $\mathbf{w} = H_2^{-1}(\mathbf{z}|\mathbf{x})$.
3. Apply the inverse marginal transformation to obtain $\mathbf{y} = H_1^{-1}(\mathbf{w}|\mathbf{x})$.

In short: $\mathbf{y} = H_1^{-1}(H_2^{-1}(\mathbf{z}|\mathbf{x})|\mathbf{x})$ with $\mathbf{z} \sim F_Z$.

3.2 INTERPRETABILITY

Our hybrid approach combines CTM's interpretability with the flexibility of autoregressive NFs: The marginal transformation step (H_1) may use shifted Bernstein polynomials, which allows quantifying feature effects on the marginal distribution, similar to coefficients in linear models or generalized additive models (GAMs) [Wood, 2017]. For instance, with a logistic base distribution, linear effect coefficients in the structured additive predictor (SAP) can be interpreted as linear changes on the level of log-odds ratios (see Appendices A.5 and B.3 for more details). Our model prioritizes marginal interpretability while accepting a trade-off for the dependence structure to gain flexibility via the masked autoregressive flow in step two (H_2). It is suitable for scenarios requiring understanding individual feature effects while modeling complex relationships between response variables. Understanding how features affect the dependence structure in this step is challenging and presents another open research question.

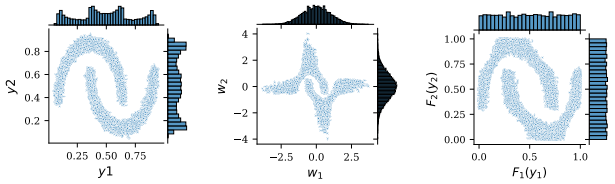
3.3 RELATION TO COPULA METHODS

Copulas model multivariate distributions by separating marginal distributions from the dependence structure. Sklar's Theorem states that a copula function can express any CDF. Let $F_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$ be the joint CDF of the response vector $\mathbf{Y} = (Y_1, \ldots, Y_J)^\top$ given features $\mathbf{X}$. Sklar's theorem implies a copula function $C(u_1, \ldots, u_J|\mathbf{x})$ such that:

$$\begin{aligned} &F_{\mathbf{Y}|\mathbf{X}}(y_1, \ldots, y_J|\mathbf{x}) \\ &= C(F_{Y_1|\mathbf{X}}(y_1|\mathbf{x}), \ldots, F_{Y_J|\mathbf{X}}(y_J|\mathbf{x})|\mathbf{x}), \end{aligned}$$

where $u_j = F_{Y_j|\mathbf{X}}(y_j|\mathbf{x})$ are uniform marginal CDFs. The copula C is a multivariate CDF on $[0, 1]^J$ with uniform

marginals. Our hybrid approach directly relates to this. The first step (H_1) models marginals $F_{Y_j|\mathbf{X}}$ and transforms them to the base distribution F_Z. Applying the probability integral transform (PIT), $u_j = F_Z(z_{1j})$, yields uniform marginals.



(a) Original Data (b) Normalized Marginals (c) Uniform Marginals

Figure 4: Illustration of the hybrid approach on the Moons dataset. (a) The original data exhibits a non-linear dependency structure. (b) After applying H_1, the marginal distributions follow a normal distribution. (c) The autoregressive flow H_2 further transforms the data to obtain approximately independent uniform marginals, implicitly modeling the copula function.

The copula density c of $\boldsymbol{u} = (u_1, \ldots, u_J)^\top$ is the ratio of the joint density and the product of the marginal densities:

$$c(\boldsymbol{u}|\mathbf{x}) = \frac{f_{\mathbf{Y}|\mathbf{X}}(F^{-1}_{Y_1|\mathbf{X}}(u_1|\mathbf{x}), \ldots, F^{-1}_{Y_J|\mathbf{X}}(u_J|\mathbf{x})|\mathbf{x})}{\prod_{j=1}^{J} f_{Y_j|\mathbf{X}}(F^{-1}_{Y_j|\mathbf{X}}(u_j|\mathbf{x})|\mathbf{x})}.$$

4 NUMERICAL EXPERIMENTS

Next, we evaluate our method using simulated and real-world datasets. All models are implemented using `TensorFlow` (v2.15.1) and `TensorFlow Probability` (v0.23.0) and trained using Adam [Martín Abadi et al., 2015, Dillon et al., 2017, Kingma et al., 2017]. In our experiments, hyperparameters were chosen based on our previous experience and settings suggested in related works. However, we did some automated hyperparameter tuning (using Optuna) on some models applied to the simulated 2D data with performance monitored on a held-out validation set. This was mainly done to understand the influence of certain parameters, but it was not necessary to obtain a good model for each data set. We found that high-order Bernstein polynomials are needed for some datasets, suggesting the fixed knot placement (**(TODO: Kann man das so bezeichnen?)**) of Bernstein polynomials could limit their flexibility[see **?**, for an ablation to the sesitivity of the Bernstein order].

Splines, on the other hand, might be more suitable for representing highly nonlinear or discontinuous functions, leading to a better fit on certain datasets. All hyperparameters used to generate the results presented are documented in the supplementary material and in the GitHub repository[3]. This includes the code to perform the hyperparameter tuning along with the defined search spaces.

[3] `https://github.com/MArpogaus/hybrid_flows`

4.1 BENCHMARK DATASETS

We evaluate our method on five common benchmark datasets POWER, GAS, HEPMASS, MINIBOONE, and BSDS300 (see Appendix Appendix B.5). We follow the preprocessing steps as in Papamakarios et al. [2017] and compare a MAF and our proposed hybrid MAF (HMAF). Both models utilize masked autoregressive transformations. Each MAF layer uses a masked affine transformation, parameterized by a masked neural network followed by an invertible linear transformation (1x1 convolution) initialized with a random (non-trainable) permutation. For the HMAF, we use a marginal transformation step with Bernstein polynomials before the MAFs layers.

For both MAFs and HMAF, we employ Rational Quadratic Splines (RQS) as the transformation functions within the MAFs layers of H_2. Hyperparameters, including the number of MAFs layers, the number of hidden units in the MADE networks, and the number of bins for the RQS transformations, were chosen based on the values reported in Durkan et al. [2019] and are detailed in the supplementary material and the GitHub repository. All models were trained using the Adam optimizer with early stopping after 50 epochs without improvements and a cosine learning rate schedule.

Table 1 presents the test log-likelihoods achieved by our models. Overall, HMAF demonstrates competitive performance compared to MAFs. Further investigation into more flexible marginal transformations could lead to even better performance and broader applicability of hybrid flow-based models. See Appendix B.5 for a visualization of individual model runs.

4.2 SIMULATED DATA

We start with the classical bivariate (i.e. $J = 2$) NFs datasets, *moons* and *circles*, exhibiting non-linear dependencies. Each dataset has $16,384$ data points (we reserved 25% for validation) generated using `scikit-learn` [Pedregosa et al., 2011]. A binary feature x based on spatial location is introduced. For *moons*, $x = 1$ indicates the lower-right moon, and $x = 0$ the upper-left. For *circles*, $x = 1$ denotes the inner circle, and $x = 0$ the outer. This assesses the models' ability to capture $f(\mathbf{y}|x)$. We compare the following models:

- **multivariate normal (MVN)** Assumes a conditional multivariate normal distribution parameterized by x using a connected network (two layers, 16 units, ReLU activation).
- **multivariate conditional transformation model (MCTM):** Employs Bernstein polynomials of order

Table 1: Test negative log-likelihood comparison against the state-of-the-art on real-world datasets (lower is better). Values are averaged over 20 trials, and their spread is reported as two standard deviations.

model	split	bsds300	gas	hepmass	miniboone	power
HMAF	test loss	-153.660 $\pm$ 0.032	-11.750 $\pm$ 0.089	18.075 $\pm$ 0.026	12.145 $\pm$ 0.045	-0.528 $\pm$ 0.003
	train loss	-165.285 $\pm$ 0.086	-11.924 $\pm$ 0.102	17.820 $\pm$ 0.026	8.815 $\pm$ 0.275	-0.574 $\pm$ 0.002
	validation loss	-168.697 $\pm$ 0.029	-11.745 $\pm$ 0.088	18.094 $\pm$ 0.026	11.374 $\pm$ 0.039	-0.538 $\pm$ 0.003
MAFs	test loss	-155.132 $\pm$ 0.046	-11.769 $\pm$ 0.030	18.143 $\pm$ 0.093	12.141 $\pm$ 0.046	-0.537 $\pm$ 0.003
	train loss	-167.913 $\pm$ 0.183	-11.869 $\pm$ 0.030	17.923 $\pm$ 0.100	9.617 $\pm$ 0.044	-0.603 $\pm$ 0.004
	validation loss	-170.199 $\pm$ 0.054	-11.765 $\pm$ 0.031	18.157 $\pm$ 0.092	11.415 $\pm$ 0.046	-0.548 $\pm$ 0.003

$M = 300$ for marginal transformations and a triangular matrix Λ for the dependency structure. In the conditional case, the marginal shift parameter β_x and Λ_x are obtained using Bernstein polynomials of order 6, as in Klein et al. [2019]. While this approach offers interpretability through SAP, the linear dependence structure constrains its capacity.

- **coupling flow (CF) and masked autoregressive flow (MAF):** These consist of two stacked coupling or masked autoregressive layers, parameterized by fully connected or masked neural networks, respectively (tree layers, 128 units, ReLU activation). Conditional models incorporate x as input to an FCN whose output is added to the first layer's parameters. We use Bernstein polynomials of order $M = 300$ (CF (B), MAF (B)) and quadratic splines with 32 bins (CF (S), MAF (S)) as transformation functions.
- **hybrid CF (HCF):** Combines elementwise Bernstein polynomials of order $M = 300$ for marginals (similar to MCTMs) with a single coupling layer to model dependencies. Again Bernstein polynomials (B) and quadratic splines (S) are compared as transformation functions in the coupling layer. As the data exhibits a very complex distribution, simple shift terms are not enough to model the feature effect in the marginals. Instead, class-specific coefficients are used to model the feature effect on the Bernstein coefficients.

All models use the Adam optimizer [Kingma and Ba, 2017] with early stopping and cosine learning rate decay [Loshchilov and Hutter, 2017]. All chosen hyperparameters are reported in the supplementary material.

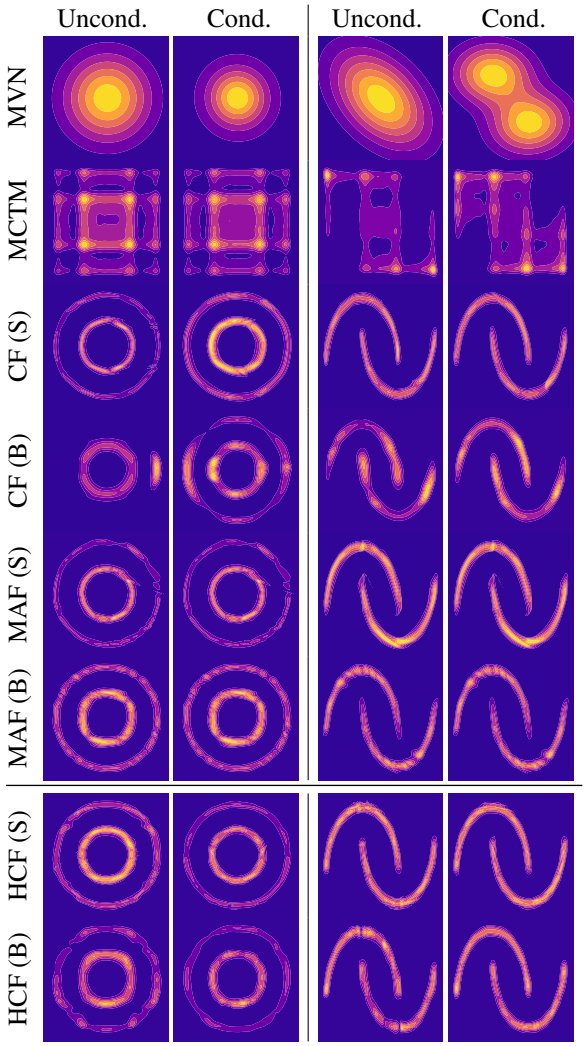


Figure 5: Estimated densities from different models fitted to simulated 2D data. Left two columns: Circles dataset, Right two columns: Moons dataset. Each cell shows the estimated density function, conditioned on x (left) and not conditioned on x (right). The conditional variable x is 1 for the inner circle and the lower right moon, respectively.

Results Figure 5 visualizes the estimated densities for different models on both the *circles* and *moons* datasets. Each cell displays contour plots of the estimated densities, both conditional (left) and unconditional (right). As expected, the multivariate normal (MVN) and the MCTM struggle with the non-linear dependencies. The MVN is restricted to elliptical distributions, while the MCTM, despite capturing marginals well, fails to represent the complex joint distribution due to its linear dependence structure. In contrast, CFs, MAFs, and hybrid CF (HCF) successfully capture the datasets' non-linear shapes, demonstrating their ability to model complex distributions. Interestingly, models using

Bernstein polynomials (CF (B), MAF (B), HCF (B)) tend to produce slightly blurrier results, indicating a bias towards smoother distributions compared to the spline-based models. Among the flexible models, quadratic spline transformations generally show a slight performance advantage over Bernstein polynomials for the CFs and MAFs architectures in terms of capturing fine-grained details of the data distribution, although the HCF models seem to perform equally well with both transformations.

Table 3 (in Appendix B.1) provides a quantitative comparison, reporting the average test negative logarithmic likelihood (NLL) across 20 trails. The results corroborate the visual observations from Figure 5. MVN and MCTM have significantly higher NLL values, indicating their inadequacy for these complex distributions. CFs and MAFs, especially with spline transformations, consistently achieve lower NLL. This difference underscores the importance of flexible dependency modeling. The HCF models perform well and demonstrate the effectiveness of the hybrid approach. The differences of Bernstein polynomial models in Figure 5 are directly reflected in lower NLL values, suggesting that these models lack certain flexibility to represent the data distribution. Consistent with the visualizations, including feature information improves model performance across all models, leading to consistently lower NLL values in the conditional setting. This improvement highlights the models' ability to leverage the feature to better capture the conditional structure of the data.

4.3 MALNUTRITION DATA

Finally, we evaluate our approach using a real-world dataset on childhood malnutrition in India, with three anthropometric indices (`stunting`, `wasting`, and `underweight`) as response variables. The goal is to model the joint distribution conditional on the child's age (`cage`). We follow the data preprocessing steps from Klein et al. [2019].

We fit three models, all using Bernstein polynomials of order $M = 6$ for the marginal transformations with a linear feature shift.

- MCTM: Identical to the model specification in Klein et al. [2019], this model combines marginal transformations with a triangular matrix Λ. We capture the influence of `cage` on both the marginals (using linear shifts) and on the elements of Λ using Bernstein polynomials of order $M = 6$, i.e. the marginal shift is modeled as Bernstein polynomial $\boldsymbol{\beta}_x = \alpha(\text{cage})^\top \boldsymbol{\vartheta}$, where α represents the Bernstein basis, and $\boldsymbol{\vartheta}$ are the coefficients.
- hybrid MAF (HMAF): For our approach, the MAFs for H_2 uses either Bernstein polynomials (B) or quadratic splines (S). We apply H_2 to $\mathbf{y}_{j>1}$, as y_1 is already normalized by H_1. Each MAFs layer is parameterized by a masked neural network conditional on `cage`. To capture dependencies between y_1 and $\mathbf{y}_{j>1}$, an additional fully connected neural network is used to condition the output of the MADE networks on y_1.

Results Table 2 presents average test NLL. HMAF variants, especially HMAF (S), outperform MCTM, indicating that the non-linear dependence modeling of the MAFs is crucial. Additionally, this is confirmed by Figure 10 in Appendix B.2 via scatter, pairwise density, and marginal density plots of the dataset as well as samples drawn from the three models. The plots indicate that the MCTM captures marginals well, but fails to model dependencies. The HMAF model shows a greater resemblance to the observed data in the pairwise density and scatter plots.

Table 2: Average test NLL on the malnutrition data. Lower values indicate better performance. Values are averaged over 20 trials, and their spread is reported as two standard deviations.

model	test loss
MCTM	3.470 ± 0.026
HMAF (S)	0.687 ± 0.206
HMAF (B)	2.062 ± 0.038

Marginal distributions To assess whether the model captures the marginal distribution, Hothorn et al. [2014] suggest using quantile-quantile (Q-Q) plots to verify $\mathbf{W}$ follows the base distribution $F_W = \mathcal{N}(0, 1)$. Figure 6 shows the empirical quantile function of marginally normalized samples $\mathbf{W}$ from the validation set plotted against the quantile function of a standard normal distribution evaluated at 200 equidistant probabilities. The results suggest a good fit of the transformation H_1, with slight underestimations of the marginals.

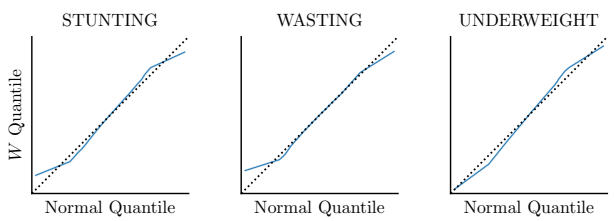


Figure 6: Q-Q plots of transformed samples against a standard normal distribution. Deviations from the diagonal indicate non-normality. The solid line represents the mean, while the shaded area indicates the 95% probability intervals obtained from 20 trials of randomly initialized models.

To assess the influence of H_2 on the marginals, we plotted the empirical quantiles of the dataset against an equal number of samples from all three models in Figure 7. All three

models leave the normalized samples for `stunting` unchanged while introducing more deviations for `wasting` and `underweight`. This deviation is especially large for the MCTM models. The HMAF, particularly the spline variants, generally perform well but struggle with upper tails.

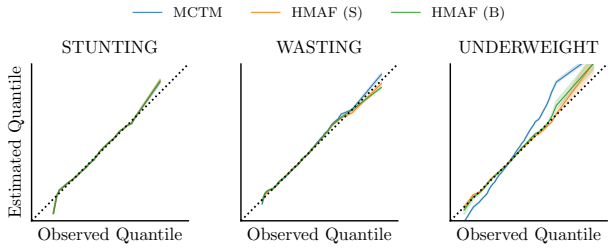


Figure 7: Q-Q plot comparing empirical quantiles of the dataset with those generated by the three models. The solid lines represent the mean, while the shaded areas indicate the 95% probability intervals obtained from 20 trials of randomly initialized models.

4.4 INTERPRETABILITY

A crucial aspect of our proposed hybrid approach is its ability to model marginal distributions in an interpretable manner while simultaneously capturing complex dependencies. Using the Malnutrition data, we demonstrate the model's interpretability in understanding learned relationships.

Feature-induced Changes in Marginal Distribution Figure 8 shows the estimated marginal CDFs $F(y_j|\text{cage})$ and PDFs $f(y_j|\text{cage})$. The plots indicate non-linear shifts towards lower values as the age of the child increases indicating a deteriorating nutrition status according to all three malnutrition indicators. This overall trend can be explained by the fact that most children are born with a close-to-normal nutritional status but the effects of resource-scarce environments become increasingly relevant as children age.

Inverse Marginal Shift Terms A more nuanced interpretation can be achieved by depicting the inverse marginal shift terms $\boldsymbol{\beta}_x$ as in Figure 9. The plot shows the complex non-linear change in the nutritional status with increasing age, but also highlights that the change is more pronounced for stunting and underweight. This coincides with the intuition that acute malnutrition (as measured by the stunting indicator) materializes more quickly than chronic malnutrition (as measured by the wasting indicator). Underweight represents a mixture of both acute and chronic malnutrition, which again fits with the estimated shift term.

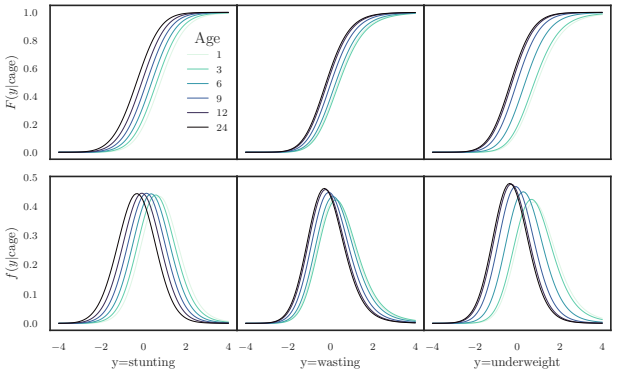


Figure 8: Comparison of marginal effect on the PDFs and CDFs of `stunting`, `wasting`, and `underweight` with respect to `cage`.

5 DISCUSSION

We introduce a hybrid approach for density regression that combines the strengths of MCTMs and autoregressive NFs. In the first step of our HMAF approach, we model the marginal distributions using a transformation model with interpretable structured additive predictor. This enables transparent modeling of feature effects on the marginal distribution. In the second step, we use an autoregressive NF to effectively capture complex, non-linear dependencies between the response variables while preserving the marginals. Our results on simulated and real-world datasets showed that our method is competitive with state-of-the-art methods.

The hybrid approach offers 1) interpretability, by transparently modeling of feature effects using structured additive predictors, 2) flexibility, by capturing complex non-linear dependencies using an autoregressive flow, and 3) efficiency, by offering fast computation of log-likelihoods and gradients due to the MADE architecture.

Acknowledgements

During the finalization of this paper, large language models (LLMs) were used to optimize language, and grammar, and restructure some parts of the document.

This research was funded by the Carl-Zeiss-Stiftung in the project "DeepCarbPlanner" (grant no. P2021-08-007).

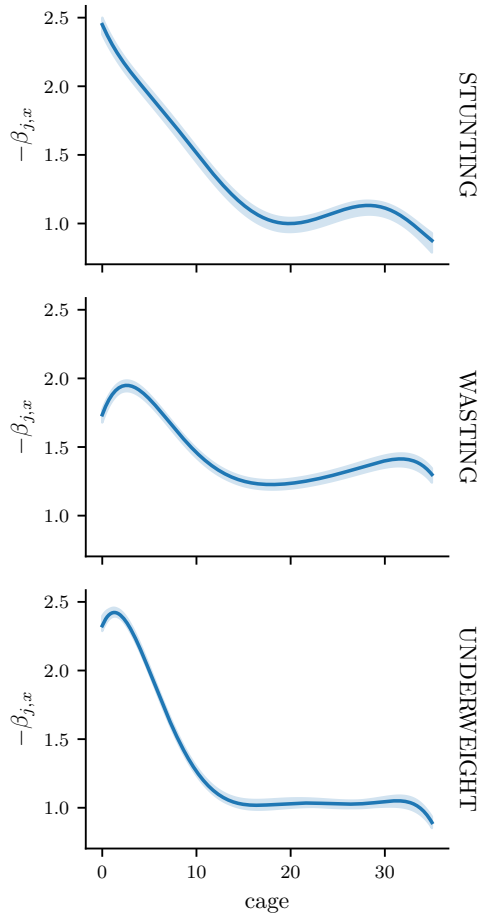


Figure 9: Inverse marginal shift $-\boldsymbol{\beta}_x$. The plot demonstrates the relationship between marginal shifts and `cage`. The solid line represents the mean and the shaded area the 95% probability intervals over 20 trials of randomly initialized models.

References

Marcel Arpogaus, Marcus Voss, Beate Sick, Mark Nigge-Uricher, and Oliver Dürr. Short-Term Density Forecasting of Low-Voltage Load Using Bernstein-Polynomial Normalizing Flows. *IEEE Transactions on Smart Grid*, 14 (6):4902–4911, November 2023. doi: 10.1109/tsg.2023.3254890.

Alexander Bauer and Claudia Czado. Pair-copula bayesian networks. *Journal of Computational and Graphical Statistics*, 25(4):1248–1271, 2016.

Philipp F. M. Baumann, Torsten Hothorn, and David Rügamer. Deep Conditional Transformation Models. In *Machine Learning and Knowledge Discovery in Databases. Research Track*, pages 3–18. Springer International Publishing, 2021. doi: 10.1007/978-3-030-86523-8_1.

G. E. P. Box and D. R. Cox. An Analysis of Transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, 26(2):211–243, 1964. ISSN 2517-6161. doi: 10.1111/j.2517-6161.1964.tb00553.x.

Tirupathi R. Chandrupatla. A New Hybrid Quadratic/Bisection Algorithm for Finding the Zero of a Nonlinear Function without Using Derivatives. *Advances in Engineering Software*, 28(3):145–149, April 1997. ISSN 0965-9978. doi: 10/djqsx4.

Joshua V. Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A. Saurous. TensorFlow Distributions, November 2017.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP, February 2017.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Cubic-Spline Flows, June 2019.

Ludwig Fahrmeir, Thomas Kneib, and Stefan Lang. *Regression*. Springer, Berlin, Heidelberg, 2009. ISBN 978-3-642-01836-7 978-3-642-01837-4. doi: 10.1007/978-3-642-01837-4.

Ludwig Fahrmeir, Thomas Kneib, Stefan Lang, and Brian Marx. *Regression: Models, Methods and Applications*. Springer, Berlin, Heidelberg, 2013. ISBN 978-3-642-34332-2 978-3-642-34333-9. doi: 10.1007/978-3-642-34333-9.

R. T. Farouki and V. T. Rajan. Algorithms for polynomials in Bernstein form. *Computer Aided Geometric Design*, 5 (1):1–26, June 1988. ISSN 0167-8396. doi: 10/b5s3bp.

Rida T. Farouki. The Bernstein Polynomial Basis: A Centennial Retrospective. *Comput. Aided Geom. Des.*, 29(6):379–419, August 2012. ISSN 0167-8396. doi: 10.1016/j.cagd.2012.03.001.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: Masked Autoencoder for Distribution Estimation. In *International Conference on Machine Learning*, pages 881–889. PMLR, 2015.

Torsten Hothorn, Thomas Kneib, and Peter Bühlmann. Conditional Transformation Models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 76(1):3–27, January 2014. ISSN 1369-7412. doi: 10.1111/rssb.12017.

Torsten Hothorn, Lisa Möst, and Peter Bühlmann. Most Likely Transformations. *Scandinavian Journal of Statistics*, 45(1):110–134, 2018. ISSN 1467-9469. doi: 10.1111/sjos.12291.

Priyank Jaini, Kira A. Selby, and Yaoliang Yu. Sum-of-Squares Polynomial Flow. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3009–3018. PMLR, May 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, January 2017.

Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving Variational Inference with Inverse Autoregressive Flow, January 2017.

Nadja Klein, Torsten Hothorn, and Thomas Kneib. Multivariate Conditional Transformation Models. *arXiv:1906.03151 [stat]*, September 2019.

Thomas Kneib, Alexander Silbersdorff, and Benjamin Säfken. Rage Against the Mean – A Review of Distributional Regression Approaches. *Econometrics and Statistics*, August 2021. ISSN 2452-3062. doi: 10.1016/j.ecosta.2021.07.006.

Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. Normalizing Flows: An Introduction and Review of Current Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979, November 2021. ISSN 1939-3539. doi: 10.1109/TPAMI.2020.2992934.

Lucas Kook, Chris Kolb, Philipp Schiele, Daniel Dold, Marcel Arpogaus, Cornelius Fritz, Philipp F. M. Baumann, Philipp Kopper, Tobias Pielok, Emilio Dorigatti, and David Rügamer. How inverse conditional flows can serve as a substitute for distributional regression. In *The 40th Conference on Uncertainty in Artificial Intelligence*, 2024.

Han Liu, Min Xu, Haijie Gu, Anupam Gupta, John Lafferty, and Larry Wasserman. Forest density estimation. *The Journal of Machine Learning Research*, 12:907–951, 2011.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts, May 2017.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

Thomas Nagler and Claudia Czado. Evading the curse of dimensionality in nonparametric density estimation with simplified vine copulas. *Journal of Multivariate Analysis*, 151:69–89, 2016.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked Autoregressive Flow for Density Estimation. *Advances in Neural Information Processing Systems*, 30, 2017.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021. ISSN 1533-7928.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python, 2011.

Beate Sick, Torsten Hothorn, and Oliver Dürr. Deep Transformation Models: Tackling Complex Regression Problems with Neural Network Based Transformation Models. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 2476–2481. IEEE, IEEE, January 2021. doi: 10/gpd2v4.

Simon N Wood. *Generalized additive models: an introduction with R*. chapman and hall/CRC, 2017.

Title in Title Case
(Supplementary Material)

Marcel Arpogaus[1] **Thomas Kneib**[2] **Thomas Nagler**[3] **David Rügamer**[3,4]

[1]Institute for Applied Research , HTWG - University of Applied Sciences , Konstanz, GERMANY
[2]Chair of Statistics and Campus Institute Data Science (CIDAS) , University of Göttingen , Göttingen, GERMANY
[3]Department of Statistics , LMU Munich, Munich, GERMANY
[4]Munich Center for Machine Learning , LMU Munich, Munich, GERMANY

A EXTENDED BACKGROUND

A.1 BERNSTEIN POLYNOMIALS

Bernstein polynomials of order M are defined as

$$h(z) = \frac{1}{M+1} \sum_{i=0}^{M} \mathrm{Be}_i(z)\vartheta_i, \tag{2}$$

where $\mathrm{Be}_i(z)$ is the density of a Beta distribution with parameters $i+1$ and $M-i+1$, and $\vartheta_0, \ldots, \vartheta_M$ are the Bernstein coefficients Farouki [2012]. As M increases, Bernstein polynomials become a *universal approximator* for smooth functions in $[0,1]$ Farouki and Rajan [1988]. In practice, $M \geq 10$ is often sufficient Hothorn et al. [2018].

Bernstein polynomials are defined for values of z within the range $[0,1]$. Outside this interval, linear extrapolation is performed. To guarantee invertibility, transformation functions must be bijective, achieved by strict monotonicity. When using Bernstein polynomials, monotonicity is enforced by constraining the Bernstein coefficients $\vartheta_0, \ldots, \vartheta_M$ to be increasing. This is achieved by recursively applying a strictly positive function like $\mathrm{softplus}$ to an unconstrained vector $\tilde{\vartheta}_0, \ldots, \tilde{\vartheta}_M$, such that $\vartheta_0 = \tilde{\vartheta}_0$ and $\vartheta_k = \vartheta_{k-1} + \mathrm{softplus}(\tilde{\vartheta}_k)$ for $k = 1, \ldots, M$. Parameter estimation can depend on initialization, as ϑ_0 is directly derived from unconstrained parameters. Monotonicity of h is enforced by constraining the coefficients to be increasing, e.g. by $\vartheta_0 = \tilde{\vartheta}_0$ and $\vartheta_k = \vartheta_{k-1} + \mathrm{softplus}(\tilde{\vartheta}_k)$ for $k = 1, \ldots, M$ Sick et al. [2021]. The inverse transformation can be found using a root-finding algorithm Chandrupatla [1997].

We require transformations to cover at least the range $[-3, 3]$ (e.g., $\pm 3\sigma$ of a standard Gaussian). Since Bernstein polynomials' boundaries are defined by its first and last coefficients ($f(0) = \vartheta_0$ and $f(1) = \vartheta_M$), we determine these from unrestricted parameters $\tilde{\vartheta}_0$ and $\tilde{\vartheta}_{M+1}$ via $\vartheta_0 = -\mathrm{softplus}(\tilde{\vartheta}_0) - 3.0 \leq -3$ and $\vartheta_M = \mathrm{softplus}(\tilde{\vartheta}_{M+1}) + 3.0 \geq 3$. To ensure $\sum_{k=1}^{M} (\vartheta_k - \vartheta_{k-1}) = \vartheta_M - \vartheta_0 =: \Delta$, the remaining coefficients ϑ_k for $k = 1, \ldots, M$ are calculated as:

$$\vartheta_k = \vartheta_{k-1} + \Delta \cdot \mathrm{softmax}\left(\left[\tilde{\vartheta}_1, \tilde{\vartheta}_3, \ldots, \tilde{\vartheta}_M\right]\right)_{k-1} \tag{3}$$

Since Δ and all $\mathrm{softmax}$ components are non-negative, $\vartheta_k - \vartheta_{k-1} \geq 0$, ensuring monotonicity.

A.2 MAXIMUM LIKELIHOOD ESTIMATION FOR PARAMETER INFERENCE

Parameter estimation usually involves minimizing a divergence between the true distribution $p_y(y)$ and the transformation model $p_y(\mathbf{y}|\theta)$ regarding the model's parameters θ [Papamakarios et al., 2021]. Since $p_y(y)$ is unknown, we minimize the negative log-likelihood of the empirical distribution $p_{\mathcal{D}}$ obtained from a finite set $\mathcal{D}$ of N i.i.d. observations $\mathbb{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$:

$$\mathrm{NLL} = -\sum_{y \in \mathcal{D}} \log\left(p_y(y;\theta)\right) \tag{4}$$

which is equivalent to minimizing the KL divergence between $p_{\mathcal{D}}$ and the flow-based model $p_y(y;\theta)$ [Papamakarios et al., 2021].

A.3 MULTIVARIATE CONDITIONAL TRANSFORMATION MODELS

MCTM [Klein et al., 2019] use element-wise transformations $\tilde{h}_j\left(y_j\right), j = 1, \ldots, J$ and a linear triangular $(J \times J)$ matrix Λ to model correlations:

$$h_j(y_1, \ldots, y_j) = \lambda_{j1}\tilde{h}_1(y_1) + \ldots + \lambda_{j,j-1}\tilde{h}_{j-1}(y_{j-1}) + \tilde{h}_j(y_j) \tag{5}$$

For conditional distributions, Λ and transformation parameters θ can depend on covariates $\mathbf{x}$:

$$h_j(\mathbf{y}|\mathbf{x}) = \sum_{i=1}^{j-1} \lambda_{ji}(\mathbf{x})\tilde{h}_i(y_i;\theta_i(\mathbf{x})) + \tilde{h}_j(y_j;\theta_j(\mathbf{x})) \tag{6}$$

A.4 AUTOREGRESSIVE TRANSFORMATION MODELS

Autoregressive flows factorize multivariate distributions based on the chain rule of probability:

$$p_y(\mathbf{y}) = \prod_{i=1}^{D} p_y(y_i|\mathbf{y}_{<i}) \tag{7}$$

Applying the change of variables formula yields:

$$p_y(\mathbf{y}) = p_z\left(h(y)\right)\left|\det \nabla h(y)\right| = \prod_{i=1}^{D} p_z\left(h_i(y_i, \mathbf{y}_{<i})\right)\left|\det \nabla h_i(y_i, \mathbf{y}_{<i})\right| \tag{8}$$

where h_i is a diffeomorphism applied to the i-th element of $\mathbf{y}$, conditioned on preceding elements $\mathbf{y}_{<i}$. The lower triangular Jacobian determinant is the product of its diagonal elements:

$$\det \nabla h(y) = \prod_{i=1}^{D} \frac{\partial F_i}{\partial y_i} \tag{9}$$

This is implemented as $z_i = h_i(y_i, \theta_i)$ with $\theta_i = c(\mathbf{y}_{<i})$, where the conditioner c_i, often a neural network, captures dependencies and enforces the autoregressive property [Papamakarios et al., 2021]. New samples are obtained via the inverse transformation $y_i = T_i^{-1}(z_i, \theta_i)$.

Two common architectures for the conditioner are coupling layers and masked autoregressive networks. Coupling flows [Dinh et al., 2017] split the response $\mathbf{y} \in \mathbb{R}^D$ into subsets $(\mathbf{y}_A, \mathbf{y}_B) \in (\mathbb{R}^d, \mathbb{R}^{D-d})$, with one subset conditioning the transformation of the other:

$$\mathbf{y} = \begin{cases} \mathbf{y}_A = \mathbf{y}_A \\ \mathbf{y}_B = h(\mathbf{y}_B; \Theta(\mathbf{y}_A)) \end{cases} \tag{10}$$

MADE [Germain et al., 2015], as used in MAFs [Papamakarios et al., 2017] and inverse autoregressive flows (IAFs) [Kingma and Ba, 2017], generalizes this using masked neural networks to enforce autoregressive constraints.

In MADE, the output d depends only on inputs $\mathbf{y}_{<d}$. This is achieved by element-wise multiplying weight matrices by binary masks, zeroing connections that violate the autoregressive property. Whether the masked network uses $\mathbf{y}$ or the latent representation $\mathbf{z}$ as input affects only whether inference or sampling is iterative [Papamakarios et al., 2021].

A.5 STRUCTURED ADDITIVE PREDICTORS FOR ENHANCED FLEXIBILITY

Structured Additive Predictors [SAP; Fahrmeir et al., 2013, 2009] allow for (non-)linear effects, interactions, and other structured terms. An exemplary predictor is given by

$$\eta(\mathbf{x}) = \beta_0 + \sum_{u=1}^{U} f_u(x_u) + \sum_{u<v} f_{uv}(x_u, x_v) + \ldots \tag{11}$$

Here, $f_u(x_u)$ is a linear or non-linear function, where the latter is usually specified using regression splines to stay in the context of parametric regression. $f_{uv}(x_u, x_v)$ are linear or smooth interaction effects. The order of interaction determines the degree of interpretability. In our hybrid approach, SAP can be used to parameterize marginal shifts $\beta_j(\mathbf{x})$ in H_1.

B EXTENDED RESULTS

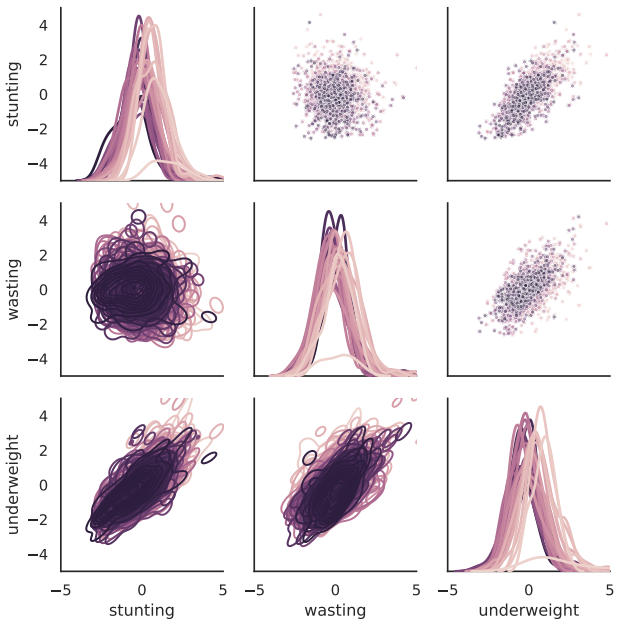
B.1 NEGATIVE LOG-LIKELIHOOD ON 2D SIMLUATION DATASETS

dataset name	circles		moons	
conditional	False	True	False	True
model				
MVN	-0.204 $\pm$ 0.000	-0.423 $\pm$ 0.002	-0.151 $\pm$ 0.002	-0.704 $\pm$ 0.004
MCTM	-0.490 $\pm$ 0.002	-0.489 $\pm$ 0.002	-0.536 $\pm$ 0.000	-1.046 $\pm$ 0.006
MAF (S)	-1.123 $\pm$ 0.022	-1.123 $\pm$ 0.022	-1.611 $\pm$ 0.042	-1.611 $\pm$ 0.042
MAF (B)	-1.179 $\pm$ 0.014	-1.179 $\pm$ 0.014	-1.625 $\pm$ 0.016	-1.625 $\pm$ 0.016
CF (S)	-1.045 $\pm$ 0.132	-1.861 $\pm$ 0.056	-1.587 $\pm$ 0.052	-2.306 $\pm$ 0.050
CF (B)	-0.651 $\pm$ 0.202	-1.657 $\pm$ 0.038	-1.350 $\pm$ 0.106	-2.186 $\pm$ 0.116
HCF (S)	-1.175 $\pm$ 0.012	-1.870 $\pm$ 0.018	-1.628 $\pm$ 0.018	-2.332 $\pm$ 0.014
HCF (B)	-1.071 $\pm$ 0.024	-1.826 $\pm$ 0.078	-1.583 $\pm$ 0.042	-2.332 $\pm$ 0.032

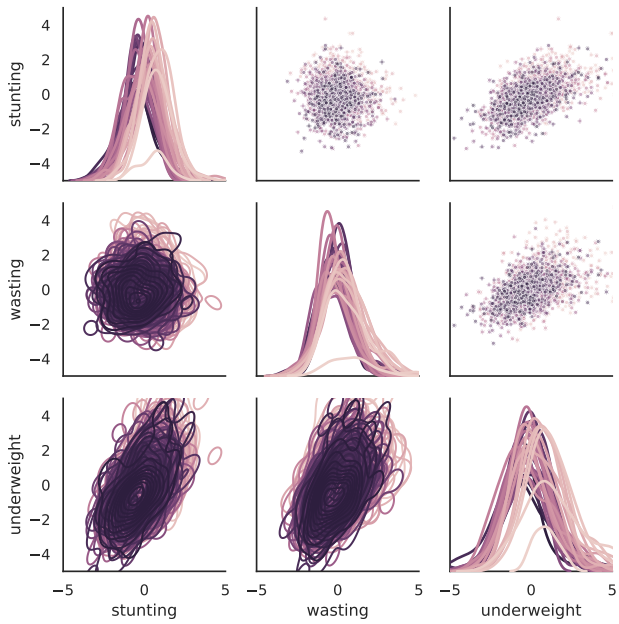
Table 3: Test negative log-likelihood on 2D simluation datasets (lower is better). Log-likelihoods are averaged over 20 trials, and their spread is reported as two standard deviations.

B.2 SCATTER PLOTS OF SAMPLES FROM MALNUTRITION MODELS

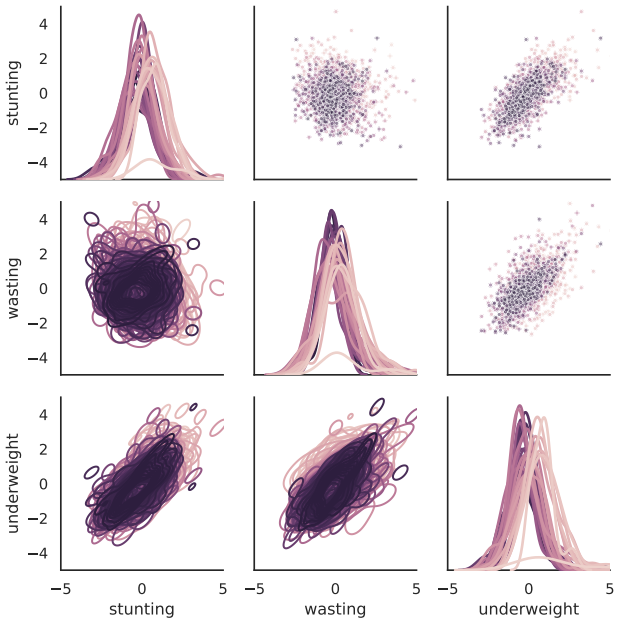
Figure 10 shows that MCTM captures marginals well but fails to model dependencies. The HMAF models, especially with spline transformations (HMAF (S)), show a greater resemblance to the observed data in the pairwise density plots.



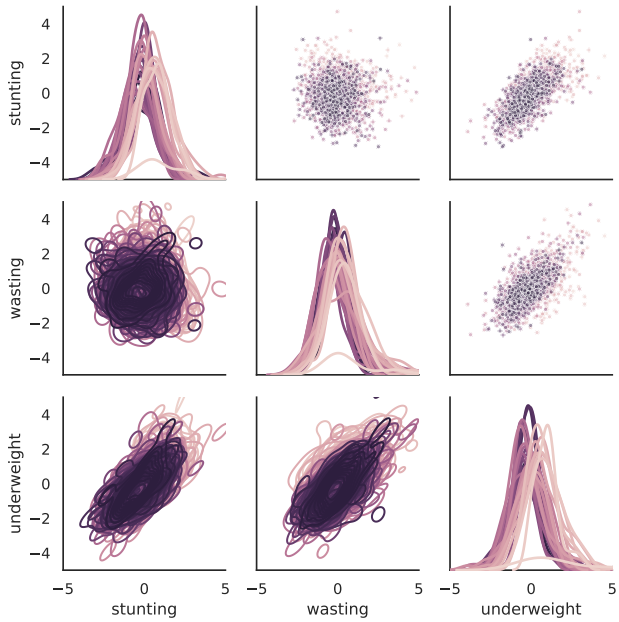
(a) Random samples from the validation dataset.



(b) Random samples from the MCTM model.



(c) Random samples from the hybrid model using quadratic splines as the transformation function.



(d) Random samples from the hybrid model using Bernstein polynomials as the transformation function.

Figure 10: Scatter plots of the three target variables—`stunting`, `wasting`, and `underweight`. In the upper triangular section, random data points are illustrated, either from the validation dataset or sampled from the three models. The diagonal displays the marginal kernel density estimation (KDE) plots, while the lower triangular region contains the two-dimensional KDE plots. The data is categorized by the variable `cage`.

B.3 INTERPRETABILITY OF COVARIATE EFFECTS

Interpretable covariate effects are important in many regression tasks. CTMs offer a balance between interpretability and flexibility [Hothorn et al., 2014]. The base distribution and transformation function parameterization determine the interpretability.

In a univariate CTM with covariate x:

$$\mathbb{P}(Y \leq y | X = x) = F_Z(h(y|\theta_x)), \tag{12}$$

where $h(y|\theta_x) = \alpha(y)^\top \boldsymbol{\vartheta} + \beta x$, with β is the covariate effect on the transformed response, and $\alpha(y)^\top \boldsymbol{\vartheta}$ controlling the distribution's shape.

- **Logistic Distribution:** β is the log-odds ratio, quantifying the change in the odds of $Y \leq y$ associated with a unit increase in x.
- **Minimum Extreme Value Distribution:** β is the log-hazard ratio, representing the influence of x on the instantaneous risk of an event.
- **Gaussian Distribution:** With a linear transformation $h(y|\theta_x)$, β is the change in Y's conditional mean per unit change in x (scaled by the standard deviation). With non-linear h, β's interpretation is less direct, affecting multiple moments.

B.4 INTERPRETABLE COVARIATE EFFECT ON THE DEPENDENCY STRUCTURE IN MCTMS

As described in Klein et al. [2019], the covariate effect on the dependency in the MCTM model can be interpreted as Spearman's rank correlation from the covariance matrix

$$\Sigma = \Lambda^\top \Lambda^{-\top} \tag{13}$$

via the Pearson correlation coefficient,

$$\rho_{ij} = \frac{\Sigma_{ij}}{\sigma_i \sigma_j} = \frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}} \tag{14}$$

where Σ_{ij} represents the covariance between variables y_i and y_j and Σ_{ii} the variance σ_i^2. To convert Pearson correlations to Spearman's rank correlation, the following transformation is applied:

$$\rho_s = \frac{6}{\pi} \arcsin\left(\frac{\rho}{2}\right) \tag{15}$$

The resulting covariate-dependent rank correlations are shown in Figure 11.

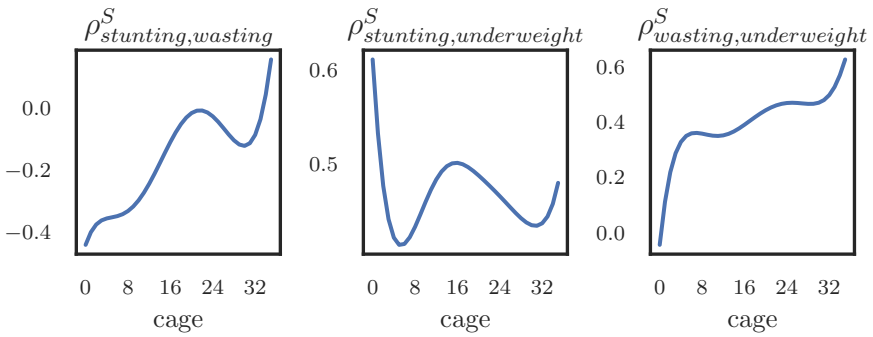


Figure 11: Comparison of Spearman's rank correlation ρ^S estimates between `stunting`, `wasting`, and `underweight` with respect to `cage`.

B.5 BENCHMARK DATASETS

We evaluate our method on five common benchmark datasets: POWER[1], GAS[2]. HEPMASS[3], MINIBOONE[4], BSDS300[5]. We follow the preprocessing steps from Papamakarios et al. [2017][6].

Figure 12 shows the distribution of NLL scores for HMAF and MAF across 20 different random initializations. The figure visualizes performance robustness. Consistent lower NLL values indicate better performance, smaller IQRs suggest less sensitivity to initialization, and a lower spread of outliers suggests more stable training.

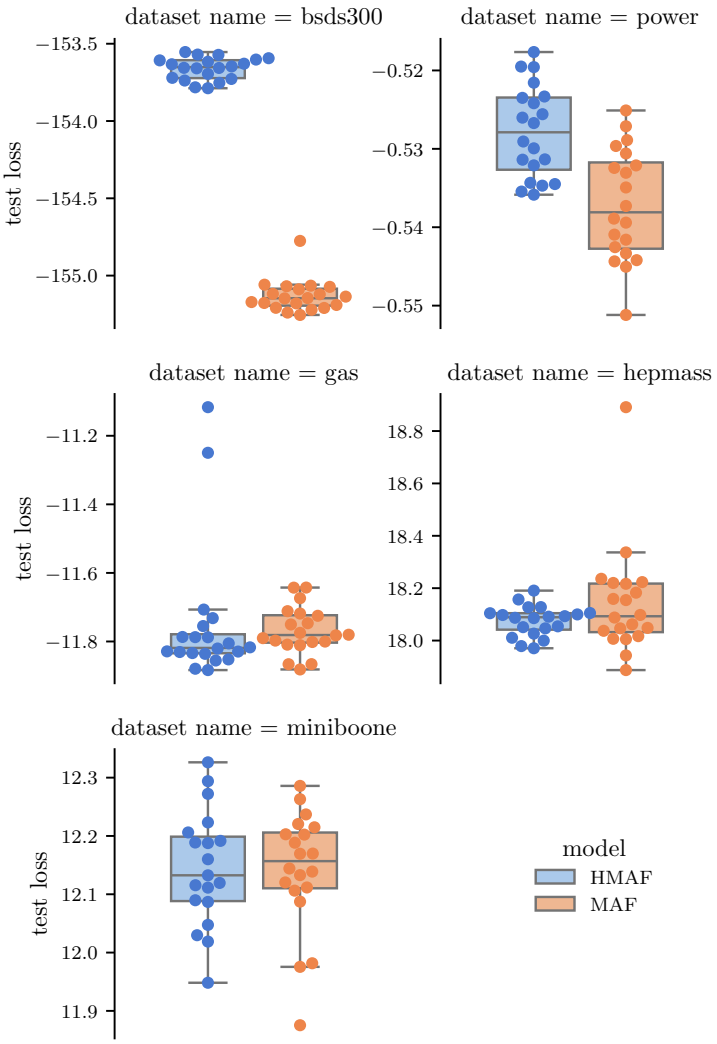


Figure 12: Distribution of negative log-likelihood (NLL) scores for the HMAF and MAF models, resulting from 20 runs with different random weight initializations. The box plots show the median, interquartile range (IQR), and the data range of 1.5 times the IQR, outside the IQR. The swarm plots above the boxes show the individual NLL scores for each run.

[1] https://archive.ics.uci.edu/dataset/235
[2] https://archive.ics.uci.edu/dataset/322
[3] https://archive.ics.uci.edu/dataset/347
[4] https://archive.ics.uci.edu/dataset/199/
[5] https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/
[6] `https://github.com/francois-rozet/uci-datasets`

C COMPLEXITY ESTIMATES (RUNTIME AND NUMBER OF PARAMETERS)

C.1 RUNTIME OF MODEL VARIANTS

As our code is currently not perfectly optimized for run-time efficiency, this can lead to a roughly 4x increase in training time and a roughly 10x increase in inference time.

Table 4: Runtime in Minutes for training and evaluation of models on benchmark data. Variance resulting from 20 runs is reported as standard deviation.

model	dataset name	train	evaluation
HMAF	bsds300	1134.793 ± 322.482	481.958 ± 0.719
	gas	331.343 ± 159.021	14.903 ± 0.041
	hepmass	253.475 ± 153.636	15.069 ± 0.060
	miniboone	78.852 ± 37.704	3.923 ± 0.010
	power	441.250 ± 28.877	11.255 ± 0.031
MAF	bsds300	267.503 ± 56.176	16.676 ± 0.032
	gas	68.998 ± 0.002	1.828 ± 0.002
	hepmass	32.984 ± 10.346	1.521 ± 0.004
	miniboone	17.708 ± 0.277	0.274 ± 0.001
	power	136.793 ± 0.004	4.917 ± 0.008

C.2 NUMBER OF PARAMETERS

Table 5: Mean runtime in seconds for training and evaluation of models on simulated data. Variance resulting deviations from 20 runs reported as standard deviation.

dataset name	conditional	model	train	evaluation
circles	False	CF (B)	35.043 ± 15.008	16.782 ± 0.238
		CF (S)	317.228 ± 233.129	17.428 ± 0.181
		HCF (B)	110.039 ± 11.230	57.759 ± 0.830
		HCF (S)	76.960 ± 37.909	29.489 ± 0.191
		MAF (B)	91.213 ± 0.354	19.562 ± 2.876
		MAF (S)	145.815 ± 36.820	17.369 ± 1.585
		MCTM	42.597 ± 0.724	29.009 ± 0.375
		MVN	44.892 ± 0.099	10.722 ± 0.067
	True	CF (B)	41.655 ± 20.906	20.423 ± 0.223
		CF (S)	329.650 ± 211.377	17.864 ± 0.267
		HCF (B)	66.384 ± 40.931	73.782 ± 1.620
		HCF (S)	171.494 ± 107.205	31.913 ± 0.128
		MAF (B)	91.153 ± 0.337	20.306 ± 2.840
		MAF (S)	145.765 ± 36.843	17.912 ± 1.554
		MCTM	304.875 ± 112.109	31.375 ± 2.577
		MVN	56.360 ± 36.401	11.708 ± 0.055
moons	False	CF (B)	33.991 ± 12.603	16.842 ± 0.145
		CF (S)	443.163 ± 80.343	17.129 ± 0.312
		HCF (B)	184.718 ± 71.683	67.237 ± 2.418
		HCF (S)	76.214 ± 39.760	29.368 ± 0.106
		MAF (B)	91.252 ± 0.538	19.872 ± 3.607
		MAF (S)	134.682 ± 62.895	14.719 ± 3.458
		MCTM	42.544 ± 0.742	28.949 ± 0.141
		MVN	44.823 ± 0.024	11.497 ± 0.046
	True	CF (B)	44.995 ± 20.047	20.477 ± 0.313
		CF (S)	368.668 ± 250.469	17.644 ± 0.415
		HCF (B)	48.243 ± 17.635	40.735 ± 0.516
		HCF (S)	169.016 ± 62.618	31.744 ± 0.146
		MAF (B)	91.215 ± 0.529	20.623 ± 3.563
		MAF (S)	135.603 ± 68.688	15.308 ± 3.487
		MCTM	475.688 ± 1.217	35.501 ± 7.524
		MVN	48.118 ± 18.322	11.030 ± 0.071

Table 6: Mean runtime in seconds for training and evaluation of models on malnutrition data. Variance resulting deviations from 20 runs reported as standard deviation.

model	train	evaluation
HMAF (B)	1074.950 ± 829.696	23.297 ± 4.855
HMAF (S)	2718.254 ± 1020.019	20.631 ± 2.793
MCTM	4300.900 ± 940.799	17.401 ± 1.822