# DIFFUSION ADDITIVE MODELS: INTERPRETABLE GENERATIVE PREDICTION THROUGH FEATURE-WISE CONTRIBUTIONS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Generalized additive models (GAMs) for location, scale and shape yield non-linear, interpretable feature-wise effects. However, they require parametric assumptions about the conditional outcome distribution and represent feature effects through deterministic functions. We introduce Diffusion Additive Models (DAMs), a generative framework for interpretable probabilistic regression and classification. Rather than specifying the shape of the conditional outcome distribution directly, DAMs treat individual feature contributions as latent variables and model their full distributions. Learning these contributions is challenging because only their sum is observed. To make this underdetermined composition learnable, we construct an encoder from a closed-form approximate posterior and calibrate its feature-wise distributions using aggregated posterior moments and homoskedastic variance routing. We then use SDE Matching to distill these distributions into feature-specific latent stochastic differential equations. DAMs capture flexible, potentially multimodal feature contributions and propagate their uncertainty to the outcome, while retaining the additive interpretability of GAMs. Across diverse regression and classification benchmarks, DAM consistently matches or outperforms existing additive models in predictive accuracy and calibration.

## 1 INTRODUCTION

Supervised learning traditionally forms the backbone of many machine learning applications. In a probabilistic sense, the goal is to learn the conditional distribution $p(y|\mathbf{x})$. While advances in deep learning and generative modeling have led to impressive results in various domains, their inner workings typically a black-box. This is detrimental for high-stakes applications, e.g., in medicine or finance [23]. One solution is to instead train a generalized additive model [GAM; 10, 11], which models the conditional mean of $p(y|\mathbf{x})$ based on the sum of interpretable, potentially non-linear feature contributions. GAMs for location, scale and shape [GAMLSS; 19] model additional distributional parameters in the same way. But this requires a parametric assumption on $p(y|\mathbf{x})$, which is a difficult empirical choice, given the vast number of available distributions [20] and the potential skewness, kurtosis, multimodality, or heteroskedasticity of $p(y|\mathbf{x})$. Despite recent integrations of neural networks [1] or gradient-boosting [16] into the GAM and GAMLSS frameworks, the interpretability of these models comes at the cost of lower predictive performance. Also, since these models provide only point estimates of the feature contributions, they ignore aleatoric uncertainty of the feature contributions. On the other end of the spectrum lie conditional diffusion models, such as CARD [9], which model the full distribution $p(y|\mathbf{x})$ non-parametrically. While achieving remarkable results, their black-box generative process eliminates any sense of interpretability [23].

Combining the interpretability of additive models with the expressiveness of diffusion models therefore seems promising, albeit challenging: Feature contributions are inherently latent constructs, requiring a diffusion model to be defined in the same latent space. Additionally, the high flexibility of the generative process works against us. A GAM is underdetermined such that a diffusion model is prone to collapse by assigning all signal to a single feature. This asks for a careful design of an encoder that transforms the limited available signal into additive, latent feature contributions.



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

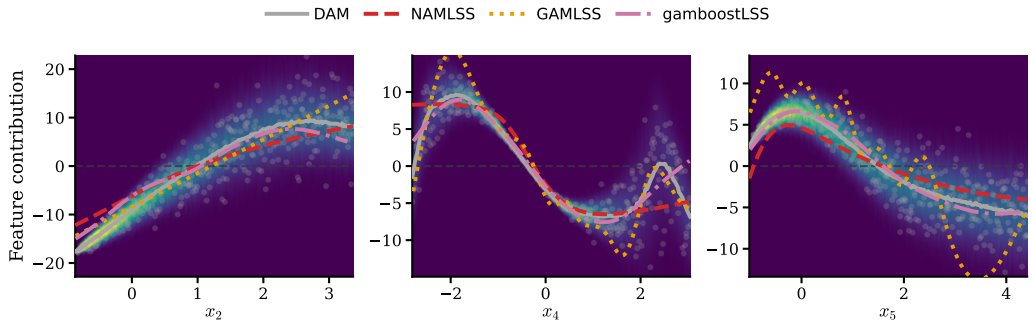



Figure 1: DAM learns the full distributions of the feature contributions using a diffusion-based generative process and propagates their shape and uncertainty into the conditional outcome distribution. We plot DAM samples (grey points), together with their mean (grey line) and their density (blue low to yellow high) in the background. Non-DAM models only provide deterministic mean functions, which we plot in color, and ignore data uncertainty in the latent feature contributions.

In this paper, we present Diffusion Additive Models (DAM), a novel model class that is able to leverage the powerful generative process of diffusion models while remaining fully interpretable. The generative process of DAM is defined as a set of feature-specific prior SDEs in the latent space of the feature contributions, resembling the generative process of a diffusion model. By carefully designing the posterior SDE via a conjugate Gaussian posterior and meaningful variance routing, DAM, unlike existing additive models which only learn point estimates of the feature contributions, learns their full distributions and propagates their shape and uncertainty to the conditional outcome distribution. By making a diffusion-based model interpretable, we tilt the interpretability-accuracy trade-off in our favor, leading to a more effective and interpretable guidance of decision-making processes in high-stakes domains.

We summarize our contributions as follows:

1. We introduce Diffusion Additive Models (DAM), which learn the full, mutually independent distributions of the feature contributions.
2. We rethink interpretable, additive models as latent variable models and develop an encoder that transforms observations into latent feature contributions.
3. We demonstrate improvements on complex regression tasks and performance parity on the simpler binary classification tasks.

## 2 Background

### 2.1 Generalized Additive Models

The goal of generalized additive models [GAMs; 10, 11] is to learn the conditional outcome distribution $p(y|\mathbf{x})$, where $y$ is the outcome and $\mathbf{x} = (x^{(1)}, \ldots, x^{(K)})$ the $K$-dimensional input. By imposing an additive structure on the predictors of the distributional parameters, additive models represent the gold standard for interpretability. A GAM models the parameters $\theta$ of a distribution as

$$\gamma(\theta) = \alpha_j + \mathbf{1}^\mathsf{T} f^{(i)}(x^{(i)}), \tag{1}$$

where $\gamma(\cdot)$ is a link function, $\alpha$ denotes an intercept and $f^{(i)}$ the latent contribution of $x^{(i)}$ that is parameterized by a possibly non-linear function. While a GAM only learns the conditional mean of exponential family distributions, treating other parameters as fixed, GAMLSS [19] generalizes this idea by parameterizing additional distributional parameters with separate additive models. This accommodates a vast number of possible distributions $p(y|\mathbf{x})$ [20] and allows the model to also capture heterogeneity in the higher order moments. Since the effect of $x^{(i)}$ on $\gamma(\theta)$ is fully captured by $f^{(i)}$, the impact of $x^{(i)}$ on the distributional parameters is fully interpretable and visualized. While traditionally $f^{(i)}$ has been parameterized using splines [10, 11, 29], more recent work utilizes neural networks instead [1, 13, 27].

### 2.2 Latent (Conditional) Diffusion Models

Diffusion models [24, 12, 17] are iterative generative models often defined in latent space [21, 5]. In a first stage, an encoder $q_\varphi(\mathbf{z}|\mathbf{y})$ and decoder $q_\varphi(\mathbf{y}|\mathbf{z})$ are trained to transform $\mathbf{y}$, e.g., an image, in and out of its latent representation $\mathbf{z}$, respectively. In the second stage, we construct a forward Stochastic Differential Equation (SDE) which describes the dynamics of the latents $\mathbf{z}_t$ at time $t$. This transports $\mathbf{z}_t$ from the data distribution at time $t = 1$ to a pure noise distribution $p(\mathbf{z}_0) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ at time $t = 0$, possibly conditioned on additional information $\mathbf{x}$, such as a text embedding:

$$\mathrm{d}\mathbf{z}_t = f(\mathbf{z}_t, \mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}, \tag{2}$$

where $f : \mathbb{R}^{d_z} \times \mathbb{R}^{d_x} \times \mathbb{R} \to \mathbb{R}^{d_z}$ is the drift, $g : \mathbb{R} \to \mathbb{R}_+$ the diffusion coefficient, and $\mathbf{w}$ describes the standard Wiener process. Typically, the drift is pre-specified and fixed, such that the forward process contains no trainable parameters. Most importantly, this SDE can be reversed [26]:

$$\mathrm{d}\mathbf{z}_t = [f(\mathbf{z}_t, \mathbf{x}, t) - g^2(t)\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t|\mathbf{x})]\mathrm{d}t + g(t)\mathrm{d}\tilde{\mathbf{w}}, \tag{3}$$

where $\tilde{\mathbf{w}}$ denotes the Wiener process running backwards in time and $\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t|\mathbf{x}) : \mathbb{R}^{d_z} \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_z}$ is the score. Only the score $\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t|\mathbf{x})$ is unknown and needs to be parameterized by a score model $s_{\boldsymbol{\theta}}(\mathbf{z}_t, \mathbf{x}, t)$ and learned via score matching [28, 26]. Then, integrating Equation (3) from $t = 1$ to $t = 0$ using the predicted score allows us to generate samples $\mathbf{z}_1$ from noise $\mathbf{z}_0$. Setting $g(t) = 0$ in Equation (2) leads to an ODE formulation, which can be trained via flow matching [2, 14, 15]. The highly flexible generative process comes at the cost of being a blackbox. This hinders the application of diffusion-based generation to many tabular data problems, in particular in high-stakes environments or decision-making processes.

## 3 Diffusion Additive Models

### 3.1 Rethinking Additive Models

While the classic view of GAMs is simple and convenient, it only considers the distribution $p(y|\mathbf{x})$ and ignores the distributions $p(f^{(i)}|x^{(i)})$ of the feature contributions themselves. Treating $f^{(i)}$ as a deterministic function of $x^{(i)}$ ignores the data uncertainty when fitting $f^{(i)}$. With GAMLSS we can add a separate additive model to parameterize the variance of $p(y|\mathbf{x})$. But this does not provide $p(z^{(i)}|x^{(i)})$ either and the learned uncertainty is limited by the assumed parametric shape of $p(y|\mathbf{x})$. In this paper, we treat the feature contributions $f^{(i)}$ as latent variables and learn their full distributions $p(f^{(i)}|x^{(i)})$. This Diffusion Additive Model (DAM) allows us to propagate attributes such as the shape and uncertainty of $p(f^{(i)}|x^{(i)})$ into $p(y|\mathbf{x})$. With a flexible generative model learning $p(f^{(i)}|x^{(i)})$, the induced $p(y|\mathbf{x})$ distribution can be highly flexible and is not limited by parametric assumptions. Let $\mathbf{f} = (f^{(1)}; \ldots; f^{(K)})$, we decompose $p(y|\mathbf{x})$ as

$$p(y|\mathbf{x}) = \int p(y|\mathbf{f})p(\mathbf{f}|\mathbf{x})\,\mathrm{d}\mathbf{f} = \int \cdots \int p(y|\mathbf{f}) \prod_{i=1}^{K} p(f^{(i)}|x^{(i)})\,\mathrm{d}f^{(i)} \ldots\, \mathrm{d}f^{(K)}. \tag{4}$$

The consequences are threefold: (1) We only make assumptions on $p(y|\mathbf{f})$ and not $p(y|\mathbf{x})$ directly. We avoid the need to make the possibly difficult empirical choice of a parametric $p(y|\mathbf{x})$ among a vast number of distributions for GAMLSS [20]. (2) By modeling $p(\mathbf{f}|\mathbf{x})$ with latent SDEs, we leverage the idea of iterative refinement, popularized by diffusion models, to flexibly capture highly complex distributions which may not follow any known parametric shape. Marginalizing out $\mathbf{f}$ shows that a simple $p(y|\mathbf{f})$ allows for a complex $p(y|\mathbf{x})$. (3) The learned feature-wise distributions $p(f^{(i)}|x^{(i)})$ are easier to interpret than the additive deterministic contributions to different distributional parameters in e.g. GAMLSS.

For clarity, we will first focus on the regression task and only later introduce the adjustments necessary to accommodate classification. For regression, we assume a Gaussian likelihood

$$p(y|\mathbf{f}) = \mathcal{N}(\mu(\mathbf{f}), \lambda^{-1}) \text{ with } \mu(\mathbf{f}) = \alpha + \mathbf{1}^\mathsf{T}\mathbf{f}, \tag{5}$$

where $\lambda$ is the precision and $\alpha$ is an intercept. We set $\lambda^{-1} = 0.002$ such that the model is forced to capture most of the variation via the feature contributions.

Below, we first define the generative process for $p(\mathbf{f}|\mathbf{x})$ via feature-wise latent prior SDEs that ensure full interpretability akin to a GAM. Then, we carefully design the posterior process that encodes the predictive signal into the latent space of the feature contributions.

### 3.2 Generative Process

We assume that the $i$:th feature contribution $f^{(i)}$ is generated by a feature-specific latent prior SDE. To simplify notation, we let $z_1^{(i)} := f^{(i)}$ such that $z_t^{(i)}$ represents the feature contribution at time $t \in [0, 1]$. We define the prior SDE as

$$\mathrm{d}z_t^{(i)} = h_{\boldsymbol{\theta}}^{(i)}(z_t^{(i)}, x^{(i)}, t)\mathrm{d}t + g_{\boldsymbol{\theta}}^{(i)}(t)\mathrm{d}w^{(i)}, \tag{6}$$

where $h_{\boldsymbol{\theta}}^{(i)} : \mathbb{R} \times \mathbb{R} \times [0, 1] \mapsto \mathbb{R}$ is the drift term, $g_{\boldsymbol{\theta}}^{(i)} : [0, 1] \mapsto \mathbb{R}_+$ the volatility, and $w^{(i)}$ the standard Wiener process. Drift and volatility are parameterized by feature-specific neural networks. The implementation details can be found in Appendix A.6. The volatility $g_{\boldsymbol{\theta}}^{(i)}$ captures feature-specific stochasticity in the diffusion trajectory. Together with the prior distribution $p_{\boldsymbol{\theta}}(z_0^{(i)}|x^{(i)}) = \mathcal{N}(0, 1)$, Equation (6) defines a sequence of probability distributions $p_{\boldsymbol{\theta}}(z_t^{(i)}|x^{(i)})$. Integrating the SDE from $t = 0$ to $t = 1$ then produces samples $z_1^{(i)} \sim p(z_1^{(i)}|x^{(i)})$ from pure noise. We introduce a noise sink $z_1^{(0)}$ that is not conditioned on any informative signal, i.e., $x^{(0)} :=$ const., with the purpose of capturing the homoskedastic uncertainty. By construction, the generative process enforces $z_1^{(i)} \perp z_1^{(j)}|\mathbf{x}$ for all $i \neq j$. Since additionally we have $p_{\boldsymbol{\theta}}(z_1^{(i)}|\mathbf{x}) = p_{\boldsymbol{\theta}}(z_1^{(i)}|x^{(i)})$, the conditional variance of the outcome neatly decomposes into $\mathrm{Var}(y|\mathbf{x}) = \sum_i \mathrm{Var}(z_1^{(i)}|x^{(i)}) + \lambda^{-1}$. GAMLSS parameterizes $\mathrm{Var}(y|\mathbf{x})$ directly, whereas DAM propagates shape and uncertainty learned by $p_{\boldsymbol{\theta}}(z_1^{(i)}|x^{(i)})$ to $p(y|\mathbf{x})$.

### 3.3 Posterior Process

**Modeling challenges.** The prior SDEs induce a posterior distribution over latent trajectories $\mathbf{z}_{0:1}^{(i)} = (z_t^{(i)})_{t\in[0,1]}$. Stacking the feature-wise prior SDEs from Equation (6) into a vector-valued SDE, with $\mathbf{g}_{\boldsymbol{\theta}}(t) = (g_{\boldsymbol{\theta}}^{(i)}(t))_{i=0}^K$, we can use Doob's $h$-transform [22] to condition on the observed $y$ at $t = 1$. The true posterior for $\mathbf{z}_t = (z_t^{(i)})_{i=0}^K$ is thus

$$\mathrm{d}\mathbf{z}_t = [\mathbf{h}_{\boldsymbol{\theta}}(\mathbf{z}_t, \mathbf{x}, t) + \mathbf{g}_{\boldsymbol{\theta}}^2(t)\nabla_{\mathbf{z}_t} \log p(y|\mathbf{z}_t)]\mathrm{d}t + \mathbf{g}_{\boldsymbol{\theta}}(t)\mathrm{d}\mathbf{w}. \tag{7}$$

Notably, $\nabla_{\mathbf{z}_t} \log p(y|\mathbf{z}_t)$ depends on all dimensions of $\mathbf{z}_t$ and therefore induces cooperation between their dynamics. Conditioning on $y$ fixes the sum in Equation (5) and couples the observations in the posterior. A mean-field assumption is therefore inappropriate. Also, since $p(y|\mathbf{z}_t)$ is generally unknown, the posterior is intractable. We therefore follow Bartosh et al. [3] and design an approximate posterior process below.

A further complication in additive models is that we only have a sparse signal for learning, since only the sum of additive components is observed. While we could parameterize $q_\psi(\mathbf{z}_t|\mathbf{x}, y)$ directly using a neural network, we found that this leads to degenerate solutions in practice: With access to $y$, the posterior probability mass collapses to a single $z^{(i)}$, setting all other contributions to zero. Lastly, additive homoskedastic noise is unidentifiable and there exists no signal that allows learning the true attribution. This limited signal calls for a careful design of a more restricted encoder and for an appropriate convention to distribute uncertainty among the feature contributions.

**Stage 1: Calibrate the sum and route homoskedastic variance.** Since only the sum of the feature contributions is observed, we first calibrate the additive means and variances in a first stage. Let $m_i(x^{(i)})$ and $v_i(x^{(i)})$ represent the conditional mean and variance of the unknown, arbitrarily complex distribution $p(z^{(i)}|x^{(i)})$. We parameterize them using feature-wise neural networks with parameters $\psi$, which we train to maximize the Gaussian quasi-likelihood of the sum:

$$\mathcal{L}_{\text{cal}} = \mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\frac{1}{2}\log\sigma_\psi^2(\mathbf{x}) + \frac{(y - \mu_\psi(\mathbf{x}))^2}{2\sigma_\psi^2(\mathbf{x})}\right],\ \mu_\psi(\mathbf{x}) = \alpha + \sum_{i=0}^K m_i,\ \sigma_\psi^2(\mathbf{x}) = \lambda^{-1} + \sum_{i=0}^K v_i, \tag{8}$$

derived by using Equation (5), integrating out $\mathbf{z}_1$ and suppressing the dependence of $m_i$ and $v_i$ on $x^{(i)}$ for brevity. This first stage resembles a NAM [1] with a Gaussian likelihood assumption and the following posterior construction naturally extends the model. After training, we freeze the parameters $\psi$ and therewith fix $m_i$ and $v_i$. However, both $m_i$ and $v_i$ lack identification. The total variance is

$\sigma^2\psi(\mathbf{x})$ is only identified up to constants, since $v_0 + v_1 = (v_0 - c) + (v_1 + c)$ for a constant $c$. A similar argument holds for the mean. We first deal with $v_i$ and defer the identification of the mean to a later section. Let $v_i$ additively decompose into homoskedastic and heteroskedastic components $v_i = v_i^{\text{homo}} + v_i^{\text{het}}(x^{(i)})$.

**Definition 1.** *Noise sink normalization. Let $v_0 \geq 0$ and let $v_i : \mathcal{X}_i \mapsto \mathbb{R}_{\geq 0}$ for $i = 1, \ldots, K$ and define $\sigma^2(\mathbf{x}) = \lambda^{-1} + v_0 + \sum_{i=1}^K v_i$, with $\lambda^{-1}$ fixed. We call this variance decomposition* noise sink normalized *if*

$$\inf_{x^{(i)} \in \mathcal{X}_i} v_i(x^{(i)}) = 0, \qquad i = 1, \ldots, K.$$

We assign $v_i^{\text{homo}} = \inf_{x^{(i)} \in \mathcal{X}_i} v_i(x^{(i)})$ and let $v_0^{\text{routed}} = v_0 + \sum_{i=1}^K v_i^{\text{homo}}$ and $v_i^{\text{routed}} = v_i^{\text{het}}$ for $i = 1, \ldots, K$. This decomposition is noise sink normalized and preserves $\sigma^2(\mathbf{x})$. It is beneficial for interpretation as the remaining feature-wise uncertainty constitutes only the heteroskedastic part.

**Stage 2: Derive the closed-form posterior.** Based on the frozen and calibrated $m_i$ and $v_i^{\text{routed}}$, we derive the posterior that updates $\mathbf{z}_1 = (z^{(i)})_{k=0}^K$ after having observed $y$ under Gaussian assumptions. We drop the superscript in $v_i^{\text{routed}}$ for brevity but all following $v_i$ should be understood as their routed variants.

**Proposition 1.** *If $p(y|\mathbf{z}_1) = \mathcal{N}(\alpha + \mathbf{1}^\mathsf{T}\mathbf{z}_1, \lambda^{-1})$ and $p(\mathbf{z}_1|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{V})$, with $\boldsymbol{\mu} = (m_i(x^{(i)}))_{i=0}^K$ and $\mathbf{V} = \operatorname{diag}(\mathbf{v})$ with $\mathbf{v} = (v_i(x^{(i)}))_{i=0}^K$, then the posterior is given by $q(\mathbf{z}_1|\mathbf{x}, y) = \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)$*

$$\begin{aligned} &\textit{with } \boldsymbol{\mu}^* = \boldsymbol{\mu} + \mathbf{w} r(\mathbf{x}, y) \;\; \textit{and} \;\; \boldsymbol{\Sigma}^* = \mathbf{V} - \frac{\mathbf{v}\mathbf{v}^\mathsf{T}}{\lambda^{-1} + \mathbf{1}^\mathsf{T}\mathbf{v}}, \\ &\textit{where } \mathbf{w} = \frac{\mathbf{v}}{\lambda^{-1} + \mathbf{1}^\mathsf{T}\mathbf{v}} \;\; \textit{and residual} \;\; r(\mathbf{x}, y) = y - \alpha - \mathbf{1}^\mathsf{T}\boldsymbol{\mu}. \end{aligned} \tag{9}$$

*Proof.* See Appendix A.1. □

Accordingly, the residual $r(\mathbf{x}, y)$ gets allocated to the feature contributions $\mathbf{z}_1$ in proportion to their (routed) conditional variances $v_i$. The deterministic conditional mean functions $\boldsymbol{\mu}$ provide an obvious connection to the existing NAM literature. The posterior mean $\boldsymbol{\mu}^*$ improves upon $\boldsymbol{\mu}$ if $\mathbb{E}[r(\mathbf{x}, y)|\mathbf{x}] \neq 0$. This may happen when the additive mean specification does not hold in practice, e.g., due to omitted features, imperfect optimization, or an incorrectly assumed likelihood.

**Stage 3: Simplify feature-wise posterior distributions.** To ensure additive interpretability, the prior SDEs in Equation (6) must be mutually independent. This implies that the actual distribution to be learned by the $i$:th prior SDE is not $q(\mathbf{z}_1|\mathbf{x}, y)$ but the aggregate posterior, which marginalizes out $y$ and the other features $\mathbf{x}^{(-i)} = (x^{(j)})_{j \neq i}$:

$$\tilde{q}^{(i)}(z_1^{(i)}|x^{(i)}) = \int\int\int q(\mathbf{z}_1|\mathbf{x}, y) p_{\text{data}}(y, \mathbf{x}^{(-i)}|x^{(i)}) \mathrm{d}\mathbf{z}^{(-i)} \mathrm{d}y \mathrm{d}\mathbf{x}^{(-i)}. \tag{10}$$

This is a mixture of Gaussians and can therefore be skewed, heavy-tailed and otherwise non-Gaussian. Let $\mathbb{E}_{\tilde{q}}[z_1^{(i)}|x^{(i)}]$ and $\operatorname{Var}_{\tilde{q}}(z_1^{(i)}|x^{(i)})$, be the aggregate posterior mean and variance, respectively. The noise sink normalization leads to extremely small $\operatorname{Var}_{\tilde{q}}(z_1^{(i)}|x^{(i)})$ and increased variation over the support of $x^{(i)}$. This complicates the learning problem for the prior SDEs. In response, we propose to standardize each dimension of $\mathbf{z}_1 \sim q(\mathbf{z}_1|\mathbf{x}, y)$ using the aggregate posterior moments. Afterwards, the generative processes can focus solely on learning the higher-order distributional structure.

We learn small feature-specific neural networks $\tilde{m}_i$ and $\tilde{v}_i$ conditioned only on $x^{(i)}$ to match the posterior moments. Again, using a Gaussian quasi-likelihood, the loss is

$$\mathcal{L}_{\text{moments}} = \mathbb{E}_{p_{\text{data}}(\mathbf{x}, y)} \left[ \frac{1}{2} \sum_{i=0}^K \left[ \log \tilde{v}_i(x^{(i)}) + \frac{\operatorname{Var}(z^{(i)}|\mathbf{x}, y) + (\mathbb{E}[z^{(i)}|\mathbf{x}, y] - \tilde{m}_i(x^{(i)}))^2}{\tilde{v}_i(x^{(i)})} \right] \right], \tag{11}$$

where $\mathbb{E}[z^{(i)}|\mathbf{x}, y]$ and $\operatorname{Var}(z^{(i)}|\mathbf{x}, y)$ are available in closed-form from Equation (9). In principle, since the actual target distributions are $\tilde{q}^{(i)}(z_1^{(i)}|x^{(i)})$ and $\operatorname{Var}_{\tilde{q}}(z_1^{(i)}|x^{(i)}) \neq v_i$ in general, the noise

sink normalization should actually be based on $\mathrm{Var}_{\tilde{q}}(z_1^{(i)}|x^{(i)})$. However, this would complicate the model significantly and we have not found empirical evidence of it leading to better predictive performance. Hence, we opt for the simpler model with $v_i$-based noise sink normalization.

With the learned $\tilde{m}_i$ and $\tilde{v}_i$ we standardize $\mathbf{z}_1$ feature-wise. In what follows, we treat $\mathbf{z}_t$ as implicitly being standardized to keep the derivations uncluttered. For instance, the prior SDEs in Section 3.2 should now define the dynamics of the standardized samples, despite referring to $\mathbf{z}_t$. At inference we reverse the standardization before computing predictions.

**Stage 4: Derive feature-wise posterior processes.** With the (standardized) posterior being defined, we follow Bartosh et al. [3] and derive the posterior process. Let $\mathbf{L}(\mathbf{x})$ represent the Cholesky factor such that $\mathbf{L}(\mathbf{x})\mathbf{L}(\mathbf{x})^\mathsf{T} = \boldsymbol{\Sigma}^*$. We assume a linear probability path [14] between $q(\mathbf{z}_1|\mathbf{x}, y)$ and the source distribution $p(\mathbf{z}_0|\mathbf{x}, y) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. The conditional marginal distribution $q(\mathbf{z}_t|\mathbf{x}, y)$ at time $t$ is then implicitly defined as

$$\mathbf{z}_t = t\boldsymbol{\mu}^* + \mathbf{L}_t\boldsymbol{\varepsilon}, \text{with } \mathbf{L}_t = t\mathbf{L}(\mathbf{x}) + (1-t)\mathbf{I}. \tag{12}$$

This marginal distribution of latents implies the conditional SDE (posterior process)

$$\mathrm{d}\mathbf{z}_t = \mathbf{f}(\mathbf{z}_t, \mathbf{x}, t, y)\mathrm{d}t + \mathbf{g}_{\boldsymbol{\theta}}(t)\mathrm{d}\mathbf{w}, \tag{13}$$

which shares the diffusion terms $\mathbf{g}_{\boldsymbol{\theta}}(t)$ with the prior SDEs in Equation (6) and where the drift is defined as

$$\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}_t, \mathbf{x}, t, y) = \frac{\mathrm{d}\mathbf{z}_t}{\mathrm{d}t} + \frac{1}{2}\mathbf{g}_{\boldsymbol{\theta}}^2(t)\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t|\mathbf{x}, y). \tag{14}$$

Due to our specification of $q(\mathbf{z}_t|\mathbf{x}, y)$, both the time-derivative and the score can be derived in closed-form. To train the generative process, we match the drifts of the prior and posterior processes via the diffusion loss

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{t\sim\mathcal{U}_{[0,1]},\mathbf{z}_t\sim q(\cdot|\mathbf{x},y)}\left[\frac{1}{2}\sum_{i=0}^{K}\left[g_{\boldsymbol{\theta}}^{(i)}(t)\right]^{-2}\left[h_{\boldsymbol{\theta}}^{(i)}(z_t^{(i)}, x^{(i)}, t) - f_{\boldsymbol{\theta}}^{(i)}(z_t^{(i)}, \mathbf{x}, y, t)\right]^2\right]. \tag{15}$$

See Appendix A.2 for a detailed derivation. If $\mathbf{g}_{\boldsymbol{\theta}}(t) = \mathbf{0}$, the generative process in Section 3.2 reduces to an ODE. We denote the ODE-based variant $\text{DAM}_{\text{ODE}}$, with the details and the corresponding flow matching loss provided in Appendix A.3.

### 3.4 Additional Considerations

**Efficient training.** Training the prior SDEs via Equation (15) requires that we are able to sample from $q(\mathbf{z}_t|\mathbf{x}, y)$ and compute the score $\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t|\mathbf{x}, y) = -\mathbf{L}_t^{-1}\boldsymbol{\varepsilon}$. Using the known structure of the posterior variance $\mathbf{V}$ in Equation (9), we can reduce the associated complexity significantly. See Appendix A.4 for the details.

**Post-hoc identification of the mean.** To identify the feature-wise means, in addition to the variances, we follow common practice [1] and center the learned distributions $p_{\boldsymbol{\theta}}(z^{(i)}|x^{(i)})$ post-hoc. To avoid costly Monte Carlo estimates, we assume that the means are matched accurately. We can then rely on the known posterior mean $\boldsymbol{\mu}^*$ from Equation (9) for centering. Specifically, $\boldsymbol{\mu}^* = \mathbb{E}[\mathbf{z}_1|\mathbf{x}, y] = \boldsymbol{\mu} + \mathbf{w}r(\mathbf{x}, y)$, where $\mathbf{w}$ depends on $\mathbf{v}^{\text{routed}}$, which we can directly estimate from the training data using the empirical average $\bar{\mathbf{z}}_1$. Then, similar to the variance routing, we center feature-wise and aggregate $\bar{\mathbf{z}}_1$ into the intercept $\alpha$. This centering is applied to the generated samples, which does not change the predictions but centers each $p(\mathbf{z}^{(i)}|x^{(i)})$ around zero.

**Adjustments for binary classification.** To accommodate binary classification where $y \in \{0, 1\}$, we define $\mathbf{z}_1$ in the latent space of a probit model. Let $p(\mathbf{z}_1|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{V})$ as in Equation (9). The probit model is defined as

$$y^* = \mathcal{N}(\alpha + \mathbf{1}^\mathsf{T}\mathbf{z}_1, 1), \qquad \text{with } y = \mathbb{I}(y^* > 0), \tag{16}$$

where $y^*$ represents some latent utility. Since $\mathbf{z}_1$ is Gaussian, we get

$$p(y^*|\mathbf{x}) = \mathcal{N}(\alpha + \mathbf{1}^\mathsf{T}\mathbf{z}_1, 1 + \mathbf{1}^\mathsf{T}\mathbf{v}). \tag{17}$$

This specification allows for a convenient likelihood for stage 1 and gives access to the exact posterior moments in closed-form, which we use to construct an approximate Gaussian posterior using moment matching as detailed in Appendix A.5. All other parts of DAM remain unchanged.

## 4 Related Work

Through its first stage calibration, DAM directly builds upon NAM [1]. However, even if the conditional mean and variance of $p(y|\mathbf{x})$ can be equally-well captured by both models, only DAM is able to learn higher-order structure by accounting for the distribution of the residual $r(\mathbf{x}, y)$. DAM reduces to a NAMLSS [27] model if every learned distribution $p(z^{(i)}|x^{(i)})$ is independently Gaussian, i.e., $p(y|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{V})$. In this case, there remains no non-Gaussian shape to learn and the likelihood is correctly specified. This outcome is likely for binary classification, as the Bernoulli distribution is entirely determined by its mean. We therefore do not expect DAM to outperform the baselines on classification tasks. For regression, DAM generalizes the existing models and removes the necessity to make an accurate parametric assumption. The flexibility of DAM stems from the diffusion-based generative process that learns the potentially complex, non-Gaussian aggregated posterior $\tilde{q}^{(i)}(z_1^{(i)}|x^{(i)})$. Unlike any existing model, DAM models the full distributions of the feature contributions, which are more interpretable than, e.g., modeling the mean and the log scale of a Gaussian additively, where the uncertainty does not decompose feature-wise. Other models like GAMLSS [19] and gamboostLSS [16] suffer from the same shortcomings as their neural-based counterparts but parameterize $f^{(i)}(x^{(i)})$ using splines or gradient-boosting, respectively. LA-NAM [6] uses Laplace approximations to capture the epistemic uncertainty involved in fitting NAMs, whereas DAM captures aleatoric uncertainty.

DAM explicitly treats the feature contributions as latent variables. This is conceptually similar to a Conditional VAE [CVAE; 25]. However, CVAE limits the learned $p(z^{(i)}|x^{(i)})$ to be Gaussian, does not involve an interpretable generative process, and is designed for images. Similarly, DAM bears resemblance to latent diffusion models [e.g., 21], which involve a similar multi-stage training process, as discussed in Section 2.2. The encoder used by DAM, however, is not a VAE but is instead derived from a closed-form posterior update acting on a calibrated additive model. Lastly, Classification and Regression Diffusion [CARD; 9] also adopt diffusion-based modeling to tabular prediction problems. Unlike DAM, CARD directly models $p(y|\mathbf{x})$ via a blackbox generative model.

## 5 Experiments

**Models.** We benchmark DAM and $\text{DAM}_{\text{ODE}}$ against other additive models for probabilistic prediction. This includes GAMLSS [20] and NAMLSS [27] as well as gamboostLSS [16]. The LSS models reduce to simpler GAM-like models for classification, as the assumed Bernoulli distribution is parameterized by a single parameter. If parameteric assumptions are needed, we impose a Gaussian likelihood for regression. We also compare DAM to CARD [9] and $\text{DAM}_{\text{moments}}$, the model obtained after the first stage as discussed in Section 3.3. See Appendix A.8 for details on the baseline models and Appendix A.6 for the DAM implementation details.

**Datasets and experimental design.** We compare the models on 22 diverse datasets (12 classification and 9 regression tasks) taken from previous work on additive models [6] as well as the BeyondArena benchmark [18]. See Appendix A.7 for a detailed overview. Following standard practice [1, 6, 27], we evaluate performance using 5-fold CV. If needed by the model, we use a random 12.5% subset of the training fold (10% of the total observations) as validation set.

**Metrics.** We evaluate predictive accuracy using CRPS, AUROC, NLL and the Brier score. For models explicitly parameterizing a Gaussian likelihood (all LSS models and $\text{DAM}_{\text{moments}}$), we compute CRPS and NLL in closed-form, avoiding any sampling uncertainty. For sample-based methods, we use the fair empirical CRPS estimator with 1000 samples per observation. We compute NLL only for classification tasks, for which we can estimate the NLL of CARD and DAM using an MC estimate. To evaluate the coverage, we report the Prediction Interval Coverage Probability (PICP), which measures the proportion of real samples that fall into the $95\%$ interval as computed from samples $\hat{y}$ from the predictive distribution. We also report the Quantile Interval Coverage Error [QICE; 9], which extends PICP to a finer granularity. It divides generated samples into deciles and measures the mean absolute error between the empirical proportion of samples in each bin and the optimal proportion (10% with 10 bins). We also report the expected calibration error (ECE) for classification tasks. Similar to QICE, ECE divides predicted probabilities into deciles and evaluates

the bin-wise calibration. While CRPS, PICP and QICE evaluate regression tasks, AUROC, Brier score and ECE apply only to classification.

### 5.1 REAL DATA

| Model | AUROC | Brier | NLL | ECE | Avg. Rank |
|---|---|---|---|---|---|
| $\text{DAM}_{\text{moments}}$ | $\mathbf{3.31 \pm 2.06}$ | $\mathbf{2.77 \pm 1.92}$ | $\mathbf{2.69 \pm 1.97}$ | $\mathbf{2.69 \pm 1.49}$ | **2.87** |
| DAM | $3.38 \pm 0.96$ | $2.92 \pm 1.26$ | $3.08 \pm 1.32$ | $3.15 \pm 1.41$ | 3.13 |
| $\text{DAM}_{\text{ODE}}$ | $3.38 \pm 1.39$ | $3.15 \pm 1.28$ | $3.31 \pm 1.32$ | $3.00 \pm 1.15$ | 3.21 |
| GAMLSS | $\mathbf{3.31 \pm 2.06}$ | $3.46 \pm 1.85$ | $3.54 \pm 1.94$ | $3.23 \pm 1.64$ | 3.38 |
| gamboostLSS | $4.15 \pm 1.86$ | $3.92 \pm 1.75$ | $3.69 \pm 1.80$ | $4.38 \pm 1.94$ | 4.04 |
| NAMLSS | $5.46 \pm 1.51$ | $5.23 \pm 1.24$ | $5.15 \pm 1.07$ | $4.54 \pm 1.94$ | 5.10 |
| CARD | $5.00 \pm 2.83$ | $6.54 \pm 1.66$ | $6.54 \pm 1.66$ | $7.00 \pm 0.00$ | 6.27 |

Table 1: Average ranks (lower is better) and standard deviations per metric across **classification** datasets. The best column-wise average rank is indicated in bold.

| Model | CRPS (↓) | PICP (↓) | QICE (↓) | Avg. Rank (↓) |
|---|---|---|---|---|
| $\text{DAM}_{\text{ODE}}$ | $2.56 \pm 1.33$ | $\mathbf{2.33 \pm 1.32}$ | $3.00 \pm 1.00$ | **2.63** |
| DAM | $3.33 \pm 1.12$ | $2.44 \pm 1.51$ | $\mathbf{2.33 \pm 1.22}$ | 2.70 |
| CARD | $\mathbf{1.44 \pm 1.33}$ | $4.44 \pm 2.30$ | $3.78 \pm 2.77$ | 3.22 |
| $\text{DAM}_{\text{moments}}$ | $5.00 \pm 1.22$ | $3.67 \pm 1.41$ | $3.78 \pm 1.09$ | 4.15 |
| NAMLSS | $4.22 \pm 1.30$ | $4.11 \pm 1.69$ | $4.11 \pm 2.26$ | 4.15 |
| gamboostLSS | $5.67 \pm 1.87$ | $4.44 \pm 1.59$ | $4.89 \pm 1.45$ | 5.00 |
| GAMLSS | $5.89 \pm 1.45$ | $6.67 \pm 1.00$ | $6.22 \pm 1.64$ | 6.26 |

Table 2: Average ranks (lower is better) and standard deviations per metric across **regression** datasets. GAMLSS fails to converge on at least one fold for four datasets, gamboostLSS for one. In these cases, we assign the maximum rank. The PICP ranking reflects the absolute deviation from the optimum of 0.95. The best column-wise average rank is indicated in bold.

**Binary classification.** Table 1 summarizes the results for the classification datasets. DAM and $\text{DAM}_{\text{ODE}}$ perform similarly to $\text{DAM}_{\text{moments}}$ and GAMLSS, considering also the detailed results in Appendix A.10.1 that show that the differences are marginal. Hence, the generative process does not improve upon the first stage model. This is expected, as for classification the Bernoulli distribution is exact and completely determined by its mean. If $\text{DAM}_{\text{moments}}$ accurately learns $p(y|\mathbf{x})$, then there is no higher-order distributional shape left to capture by DAM. Interestingly, CARD consistently performs the worst, despite being the least restricted model. We trace this back to CARD having originally been designed for regression, with only a very crude adaptation to classification problems, without defining a proper likelihood-based objective [9]. Lastly, $\text{DAM}_{\text{moments}}$ outperforming NAMLSS indicates that parameterizing both $m_i$ and $v_i$ (instead of only $m_i$) and relying on the probit model benefits predictive performance and calibration.

**Regression.** Table 2 summarizes the results for the regression datasets. In this more complex problem, DAM outperforms all other additive models on average. The difference to $\text{DAM}_{\text{moments}}$ illustrates the benefit of the flexible generative process. In terms of calibration, it even beats the more flexible blackbox CARD model. While $\text{DAM}_{\text{ODE}}$ outranks DAM on CRPS, the trade-off is slightly worse calibration performance in terms of QICE coverage. The spline-based GAMLSS fails to converge on at least one fold for four out of nine datasets but can perform well when it does, as seen in Appendix A.10.1. We emphasize that the DAM models achieve this performance without having been tuned, unlike gamboostLSS and GAMLSS. Also, on regression datasets DAM trains faster than gamboostLSS, as shown in Appendix A.10.1.

**Interpretability.** There are significant differences in the degree of interpretability. CARD is non-interpretable and a black box. GAMLSS, gamboostLSS and NAMLSS assume a Gaussian conditional outcome distribution and parameterize its mean as well as the log scale or variance. This implies that

for GAMLSS and gamboostLSS, the variation in $y$ does not decompose into feature-wise uncertainty. In Appendix A.10.2 and A.10.3, we therefore compare the learned feature-wise distributions learned by DAM only to the feature-wise mean functions learned by the additive baselines.

### 5.2 Synthetic Data

We investigate the origin of the performance improvements of DAM on regression datasets using synthetic data in Appendix A.9. We find that if ground-truth uncertainty in the feature contributions is moderate, DAM fails to identify the ground-truth non-Gaussian structure, like heavy tails, skewness and bi-modality, accurately. We trace this back to the fundamental design of the closed-form posterior, leading to a conservative assignment of residual signal proportional to the estimated variance. The main benefit of DAM is therefore the principled approach of refining first stage predictions.

### 5.3 Ablations

To investigate the added value of each proposed component, we conduct an ablation study on the regression datasets. We compare DAM to $\text{DAM}_{\text{moments}}$, to model (A), which does not standardize $\mathbf{z}_1$ before training the generative process, to model (B), which does not apply noise sink normalization, to model (C), which adds collaboration between the prior SDEs, and to $\text{DAM}_{\text{ODE}}$, which is trained using flow matching. Model (C) augments each prior SDE with the detached sum $z_t^{\text{sum}} = \mathbf{1}^\top \mathbf{z}_t$, i.e., $h_{\boldsymbol{\theta}}^{(i)}(z_t^{(i)}, x^{(i)}, t, z_t^{\text{sum}})$ in Equation (6). Thus, each prior SDE can adapt to the aggregate at time $t$, also during generation. This collaboration between prior SDEs destroys the interpretability.

| Model | CRPS (↓) | PICP (↓) | QICE (↓) | Avg. Rank (↓) |
|---|---|---|---|---|
| (C): collab. | $\mathbf{2.00 \pm 1.50}$ | $3.00 \pm 2.40$ | $\mathbf{1.56 \pm 1.67}$ | $\mathbf{2.19}$ |
| $\text{DAM}_{\text{ODE}}$ | $\mathbf{2.00 \pm 1.32}$ | $\mathbf{2.89 \pm 1.45}$ | $3.56 \pm 1.51$ | 2.81 |
| DAM | $3.56 \pm 0.88$ | $3.11 \pm 1.62$ | $3.22 \pm 1.39$ | 3.30 |
| (A): no standard. | $3.22 \pm 1.09$ | $3.22 \pm 0.67$ | $3.78 \pm 1.39$ | 3.41 |
| (B): no routing | $4.22 \pm 0.97$ | $4.44 \pm 1.33$ | $3.89 \pm 1.27$ | 4.19 |
| $\text{DAM}_{\text{moments}}$ | $6.00 \pm 0.00$ | $4.33 \pm 2.06$ | $5.00 \pm 1.41$ | 5.11 |

Table 3: Average ranks (lower is better) and standard deviations per metric across **regression** datasets. The PICP ranking reflects the absolute deviation from the optimum of 0.95. The best column-wise average rank is indicated in bold.

Table 3 shows the summary of our ablation study. As expected, on average allowing feature-wise prior SDEs to collaborate improves performance at the cost of losing interpretability. Most importantly, all variants improve consistently upon the first-stage $\text{DAM}_{\text{moments}}$ model, illustrating the benefits the flexible generative process. Switching from an SDE to an ODE-based generative model can improve performance at the cost of some calibration, as also shown in Table 2. In all other cases, we emphasize the the absolute differences in the performance metrics between methods is marginal, as illustrated in Appendix A.10.4. Comparing DAM to model (A) thus makes the benefit of standardization questionable, even though it might reduce effective training time due to faster model convergence.

## 6 Conclusion, Limitations and Future Work

We introduced DAM, a diffusion-based additive model that treat feature contributions as latent variables and learns their full distributions. It achieves performance parity on simple distributions, as in the case of classification, and outperforms the baselines on the more complex regression task. This performance gain comes at cost of no exact likelihood evaluation being available and the requirement for iterative sampling, which can become costly on large test sets. These costs are only worthwhile when modeling complex, continuous distributions.

There are many promising directions for future work. DAM can be extended to multiclass classification, vector outcomes, e.g., to model time correlation, or other modalities that can would benefit from the diffusion-based generative process. Another logical next step is to incorporate epistemic uncertainty into DAM, which would quantify how uncertain the model is about assigning predictive signal to one feature versus another.

## AI USE STATEMENT

In this work, we used generative AI tools for suggesting directions on how to adapt DAM to classification and how to efficiently compute the Cholesky factor, for integrating the R-packages of the GAMLSS and gamboostLSS models into our python codebase via rpy2, and for verifying our own mathematical derivations. We have not used generative AI tools for writing (text or maths), generating or processing data, designing the research methodology or experiments, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for coding assistance and brainstorming as well as searching for information. We have reviewed all AI-assisted work. We verified that generated code works as intended and that any search results can be traced back to legitimate sources of information. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES


[1] Rishabh Agarwal and Levi Melnick. Neural Additive Models: Interpretable Machine Learning with Neural Nets. In *Advances in Neural Information Processing Systems*, volume 34. Curran Associates, Inc., 2021.

[2] Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic Interpolants: A Unifying Framework for Flows and Diffusions, November 2023.

[3] Grigory Bartosh, Dmitry Vetrov, and Christian A. Naesseth. SDE Matching: Scalable and Simulation-Free Training of Latent Stochastic Differential Equations. In *International Conference on Machine Learning*, 2025.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, NY, USA, 2006. ISBN 978-0-387-31073-2.

[5] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your Latents: High-Resolution Video Synthesis with Latent Diffusion Models. April 2023.

[6] Kouroche Bouchiat, Alexander Immer, Hugo Yèche, Gunnar Rätsch, and Vincent Fortuin. Improving Neural Additive Models with Bayesian Principles. In *International Conference on Machine Learning*, 2024. doi: 10.48550/arXiv.2305.16905.

[7] Daniele Durante. Conjugate Bayes for probit regression via unified skew-normal distributions. *Biometrika*, 106(4):765–779, 2019. doi: 10.1093/biomet/asz034.

[8] Yury Gorishniy, Ivan Rubachev, and Artem Babenko. On Embeddings for Numerical Features in Tabular Deep Learning. *arXiv preprint arXiv:2203.05556*, 2022.

[9] Xizewen Han, Huangjie Zheng, and Mingyuan Zhou. CARD: Classification and Regression Diffusion Models, December 2022.

[10] Trevor Hastie and Robert Tibshirani. Generalized Additive Models. *Statistical Science*, 1(3): 297–310, 1986.

[11] Trevor Hastie and Robert Tibshirani. *Generalized Additive Models*. Chapman and Hall, London, 1990.

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

[13] Young Kyung Kim, J Matıas Di Martino, and Guillermo Sapiro. Generalizing Neural Additive Models via Statistical Multimodal Analysis. In *Workshop on Interpretable ML in Healthcare at International Conference on Machine Learning*, 2023.

[14] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations*, 2023.

[15] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. In *International Conference on Learning Representations*, 2023.

[16] Andreas Mayr, Nora Fenske, Benjamin Hofner, Thomas Kneib, and Matthias Schmid. Generalized Additive Models for Location, Scale and Shape for High Dimensional Data—A Flexible Approach Based on Boosting. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 61(3):403–427, 2012. ISSN 0035-9254, 1467-9876. doi: 10.1111/j.1467-9876.2011.01033.x.

[17] Alex Nichol and Prafulla Dhariwal. Improved Denoising Diffusion Probabilistic Models. In *International Conference on Machine Learning*, pp. 8162–8171, 2021.

[18] Lennart Purucker, Andrej Tschalzev, Nick Erickson, Gioia Blayer, David Holzmüller, Alan Arazi, Alexander Pfefferle, Mustafa Tajjar, Gaël Varoquaux, and Frank Hutter. Beyond IID: How General Are Tabular Foundation Models, Really? *arXiv preprint arXiv:2606.30410*, 2026. doi: 10.48550/arXiv.2606.30410.

[19] Robert A. Rigby and Mikis D. Stasinopoulos. Generalized Additive Models for Location, Scale and Shape. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 54(3):507–554, 2005. ISSN 0035-9254, 1467-9876. doi: 10.1111/j.1467-9876.2005.00510.x.

[20] Robert A. Rigby, Mikis D. Stasinopoulos, Gillian Z. Heller, and Fernanda De Bastiani. *Distributions for Modeling Location, Scale, and Shape: Using GAMLSS in R*. Chapman and Hall/CRC, 1st edition, 2021. doi: 10.1201/9780429298547.

[21] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10674–10685, 2022.

[22] Simo Särkkä and Arno Solin. *Applied Stochastic Differential Equations*. Cambridge University Press, 2019.

[23] Ruxue Shi, Yili Wang, Mengnan Du, Xu Shen, Yi Chang, and Xin Wang. A Comprehensive Survey of Synthetic Tabular Data Generation. *arXiv preprint arXiv:2504.16506*, 2025. doi: 10.48550/arXiv.2504.16506.

[24] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265, 2015.

[25] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28, 2015.

[26] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*, 2021.

[27] Anton Thielmann, René-Marcel Kruse, Thomas Kneib, and Benjamin Säfken. Neural Additive Models for Location Scale and Shape: A Framework for Interpretable Neural Regression Beyond the Mean. In *International Conference on Artificial Intelligence and Statistics*, 2024. doi: 10.48550/arXiv.2301.11862.

[28] Pascal Vincent. A Connection Between Score Matching and Denoising Autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

[29] Simon N Wood. Stable and Efficient Multiple Smoothing Parameter Estimation for Generalized Additive Models. *Journal of the American Statistical Association*, 99(467):673–686, 2004. ISSN 0162-1459. doi: 10.1198/016214504000000980.

# A Appendix

## A.1 Derivation of Posterior

We assume a conditional prior distribution for the feature contribution $z_1^{(i)}$ such that $z_1^{(i)} \sim \mathcal{N}(\mu_i(x^{(i)}), v_i(x^{(i)}))$. We stack all feature contributions in $\mathbf{z}_1 = (z_1^i)_{i=0}^K, \dots z_1^{(K)})^\mathsf{T} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{V})$ where $\mathbf{V} = \mathrm{diag}(\mathbf{v})$ with $\mathbf{v} = (v_i(x^{(i)}))_{i=0}^K$. For the observed outcome $y \in \mathbb{R}$, we assume a Gaussian likelihood:

$$p(y|\mathbf{z}_1) = \mathcal{N}(\alpha + \mathbf{1}^\mathsf{T}\mathbf{z}, \lambda^{-1}), \tag{18}$$

where $\lambda$ is the precision parameter and $\alpha$ an intercept. Our goal is to derive the posterior $p(\mathbf{z}|\mathbf{x}, y)$.

As both prior and likelihood are Gaussian, we can make use of established results for linear Gaussian models. Specifically, using Equation 2.116 from Bishop [4, p.93], we get

$$\begin{aligned} p(\mathbf{z}|\mathbf{x}, y) &= \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*) \\ \boldsymbol{\mu}^* &= \boldsymbol{\Sigma}^*[\mathbf{1}\lambda(y - \alpha) + \mathbf{V}^{-1}\boldsymbol{\mu}] \\ \boldsymbol{\Sigma}^* &= [\mathbf{V}^{-1} + \mathbf{1}\lambda\mathbf{1}^\mathsf{T}]^{-1}. \end{aligned} \tag{19}$$

To proceed, we first derive a closed-form expression for the posterior variance $\boldsymbol{\Sigma}^*$, which we then use to significantly simplify the posterior mean. We note that $\boldsymbol{\Sigma}^*$ constitutes a rank-1 downdate of $\mathbf{V}^{-1}$. By applying the Sherman-Morrison formula, the posterior variance becomes

$$\begin{aligned} \boldsymbol{\Sigma}^* = [\mathbf{V}^{-1} + \mathbf{1}\lambda\mathbf{1}^\mathsf{T}]^{-1} &= \mathbf{V} - \mathbf{V}\mathbf{1}(\lambda^{-1} + \mathbf{1}^\mathsf{T}\mathbf{V}\mathbf{1})^{-1}\mathbf{1}^\mathsf{T}\mathbf{V} \\ &= \mathbf{V} - \frac{\mathbf{V}\mathbf{1}\mathbf{1}^\mathsf{T}\mathbf{V}}{\lambda^{-1} + \mathbf{1}^\mathsf{T}\mathbf{V}\mathbf{1}} \\ &= \mathbf{V} - \frac{\mathbf{v}\mathbf{v}^\mathsf{T}}{\lambda^{-1} + \mathbf{1}^\mathsf{T}\mathbf{v}}, \end{aligned} \tag{20}$$

where we used that $(\lambda^{-1} + \mathbf{1}^\mathsf{T}\mathbf{V}\mathbf{1})$ is a scalar and therefore trivially invertible and $\mathbf{V}\mathbf{1} = \mathbf{v}$. Notably, $\boldsymbol{\Sigma}^*$ induces the anticorrelation between the components of $\mathbf{z}$ that is necessary for the sum $y = \alpha + \mathbf{1}^\mathsf{T}\mathbf{z}_1$ to hold once $y$ has been observed, accounting for variance $\lambda^{-1}$.

What is left is to simplify the expression for the posterior mean $\boldsymbol{\mu}^*$:

$$\begin{aligned} \boldsymbol{\mu}^* &= \boldsymbol{\Sigma}^*[\mathbf{1}\lambda(y - \alpha) + \mathbf{V}^{-1}\boldsymbol{\mu}] \\ &= \boldsymbol{\Sigma}^*\mathbf{1}\lambda(y - \alpha) + \boldsymbol{\Sigma}^*\mathbf{V}^{-1}\boldsymbol{\mu}. \end{aligned} \tag{21}$$

Plugging in for $\boldsymbol{\Sigma}^*$ using Equation (20), the first term simplifies as

$$\begin{aligned} \boldsymbol{\Sigma}^*\mathbf{1}\lambda(y - \alpha) &= \left[\mathbf{V} - \frac{\mathbf{V}\mathbf{1}\mathbf{1}^\mathsf{T}\mathbf{V}}{\lambda^{-1} + \mathbf{1}^\mathsf{T}\mathbf{V}\mathbf{1}}\right]\mathbf{1}\lambda(y - \alpha) \\ &= \left[\mathbf{V}\mathbf{1} - \frac{\mathbf{V}\mathbf{1}\mathbf{1}^\mathsf{T}\mathbf{V}\mathbf{1}}{\lambda^{-1} + \mathbf{1}^\mathsf{T}\mathbf{V}\mathbf{1}}\right]\lambda(y - \alpha) \\ &= \mathbf{V}\mathbf{1}\left[1 - \frac{\mathbf{1}^\mathsf{T}\mathbf{V}\mathbf{1}}{\lambda^{-1} + \mathbf{1}^\mathsf{T}\mathbf{V}\mathbf{1}}\right]\lambda(y - \alpha) \\ &= \mathbf{V}\mathbf{1}\left[\frac{\lambda^{-1}}{\lambda^{-1} + \mathbf{1}^\mathsf{T}\mathbf{V}\mathbf{1}}\right]\lambda(y - \alpha) \\ &= \frac{\mathbf{V}\mathbf{1}}{\lambda^{-1} + \mathbf{1}^\mathsf{T}\mathbf{V}\mathbf{1}}(y - \alpha) \\ &= \frac{\mathbf{v}}{\lambda^{-1} + \mathbf{1}^\mathsf{T}\mathbf{v}}(y - \alpha). \end{aligned} \tag{22}$$

For the second term, we get

$$\begin{aligned} \boldsymbol{\Sigma}^*\mathbf{V}^{-1}\boldsymbol{\mu} &= \left[\mathbf{V} - \frac{\mathbf{V}\mathbf{1}\mathbf{1}^\mathsf{T}\mathbf{V}}{\lambda^{-1} + \mathbf{1}^\mathsf{T}\mathbf{V}\mathbf{1}}\right]\mathbf{V}^{-1}\boldsymbol{\mu} \\ &= \boldsymbol{\mu} - \frac{\mathbf{v}}{\lambda^{-1} + \mathbf{1}^\mathsf{T}\mathbf{v}}\mathbf{1}^\mathsf{T}\boldsymbol{\mu}. \end{aligned} \tag{23}$$

Adding the above expressions for the first and second terms gives us the posterior mean

$$\begin{aligned}\boldsymbol{\mu}^* &= \frac{\mathbf{v}}{\lambda^{-1} + \mathbf{1}^\mathsf{T}\mathbf{v}}(y - \alpha) + \boldsymbol{\mu} - \frac{\mathbf{v}}{\lambda^{-1} + \mathbf{1}^\mathsf{T}\mathbf{v}}\mathbf{1}^\mathsf{T}\boldsymbol{\mu} \\ &= \boldsymbol{\mu} + \frac{\mathbf{v}}{\lambda^{-1} + \mathbf{1}^\mathsf{T}\mathbf{v}}(y - \alpha - \mathbf{1}^\mathsf{T}\boldsymbol{\mu}) \\ &= \boldsymbol{\mu} + \mathbf{w}r(\mathbf{x}, y).\end{aligned} \tag{24}$$

### A.2 Derivation of diffusion loss

Let $\mathbf{L}(\mathbf{x})$ represent the Cholesky factor such that $\mathbf{L}(\mathbf{x})\mathbf{L}(\mathbf{x})^\mathsf{T} = \boldsymbol{\Sigma}^*$. At $t = 1$, we observe the posterior distribution $q(\mathbf{z}_1|\mathbf{x}, y)$ as defined in Equation (9). As we chose we choose set $p_{\boldsymbol{\theta}}(z_0^{(i)}|x^{(i)})$ as the source distribution for the prior process in Section 3.2, we let $q(\mathbf{z}_0|\mathbf{x}, y) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. For simplicity, we construct a linear probability path [14] that interpolates between $q(\mathbf{z}_1|\mathbf{x}, y)$ at $t = 1$ and $q(\mathbf{z}_0|\mathbf{x}, y)$ at $t = 0$. The conditional marginal distribution $q(\mathbf{z}_t|\mathbf{x}, y)$ at time $t$ ca be implicitly defined as

$$\mathbf{z}_t = t\boldsymbol{\mu}^* + \mathbf{L}_t\boldsymbol{\varepsilon}, \text{with } \mathbf{L}_t = t\mathbf{L}(\mathbf{x}) + (1 - t)\mathbf{I}. \tag{25}$$

Following Bartosh et al. [3], the $q(\mathbf{z}_t|\mathbf{x}, y)$ implies a posterior process described by the conditional SDE

$$\mathrm{d}\mathbf{z}_t = \mathbf{f}(\mathbf{z}_t, \mathbf{x}, t, y)\mathrm{d}t + \mathbf{g}_{\boldsymbol{\theta}}(t)\mathrm{d}\mathbf{w}, \tag{26}$$

with $g_{\boldsymbol{\theta}}(t)$ being shared with the prior SDEs in Equation (6) and the drift being defined as

$$\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}_t, \mathbf{x}, t, y) = \frac{\mathrm{d}\mathbf{z}_t}{\mathrm{d}t} + 0.5\mathbf{g}^2_{\boldsymbol{\theta}}(t)\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t|\mathbf{x}, y). \tag{27}$$

Conveniently, our specification of $q(\mathbf{z}_t|\mathbf{x}, y)$ in Equation (25) and the closed-form posterior $q(\mathbf{z}_1|\mathbf{x}, y)$ allow for closed-form computations of the time-derivative $\frac{\mathrm{d}\mathbf{z}_t}{\mathrm{d}t}$ and the score $\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t|\mathbf{x}, y)$.

Using Equation (25), the time-derivative is given by

$$\frac{\mathrm{d}\mathbf{z}_t}{\mathrm{d}t} = \boldsymbol{\mu}^* + (\mathbf{L}(\mathbf{x}) - \mathbf{I})\boldsymbol{\varepsilon}, \tag{28}$$

and the score by

$$\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t|\mathbf{x}, y) = -[\mathbf{L}_t\mathbf{L}_t^\mathsf{T}]^{-1}(\mathbf{z}_t - t\boldsymbol{\mu}^*) = -[\mathbf{L}_t\mathbf{L}_t^\mathsf{T}]^{-1}\mathbf{L}_t\boldsymbol{\varepsilon} = -\mathbf{L}_t^{-1}\boldsymbol{\varepsilon}, \tag{29}$$

using that $q(\mathbf{z}_t|\mathbf{x}, y)$ is Gaussian. Hence, computing the score requires $\mathbf{L}_t$, which in turn depends on the Cholesky factor $\mathbf{L}(\mathbf{x})$ of $\boldsymbol{\Sigma}$, and its inversion. In Appendix A.4 we show how the computational complexity can be reduced by exploiting the known structure of $\boldsymbol{\Sigma}^*$. Compared to SDE matching [3] we do not need to include an reconstruction or prior losses. Since we construct $q(\mathbf{z}_t|\mathbf{x}, y)$ such t that it matches the generative source distribution at $t = 0$, the prior loss is always zero. Secondly, the posterior $q(\mathbf{z}_1|\mathbf{x}, y)$ has no trainable parameters but is fixed. Hence, the reconstruction loss $\mathcal{L}_{\text{recon}} = \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x},y)}[-\log p(y|\mathbf{z}_1)]$ is constant and can therefore be ignored.

### A.3 Flow matching setup

If we let $\mathbf{g}_{\boldsymbol{\theta}}(t) = 0$ in Equation (6), the prior and posterior SDEs reduce to ODEs. Also, the diffusion loss in Equation (15) becomes undefined. Instead, we rely on flow matching [2, 14, 15] to train the prior SDEs by minimizing

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t\sim\mathcal{U}_{[0,1]},\mathbf{z}_t\sim q(\cdot|\mathbf{x},y)}\left[\frac{1}{2}\sum_{i=0}^{K}\left[h^{(i)}_{\boldsymbol{\theta}}(z_t^{(i)}, x^{(i)}, t) - f^{(i)}_{\boldsymbol{\theta}}(z_t^{(i)}, \mathbf{x}, y, t)\right]^2\right]. \tag{30}$$

For sampling, we do no longer require the Euler–Maruyama sampler and instead use the popular Runge-Kutta 4th-Order (RK4) ODE-solver with 50 steps.

### A.4 Efficient computation

The computation of the posterior SDE drift in Equation (14) requires access to $\mathbf{L}(\mathbf{x})$ and the computation of the score $\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t|\mathbf{x}, y) = -\mathbf{L}_t^{-1}\boldsymbol{\varepsilon}$. The general algorithm for computing $\mathbf{L}(\mathbf{x})$ from the posterior covariance $\boldsymbol{\Sigma}^*$ is $O((K + 1)^3)$ where $K + 1$ is the dimension of $\mathbf{z}_t$ (accounting for $K$ features and the noise sink). We can do better by exploiting the known covariance structure.

**Matrix square root of $\boldsymbol{\Sigma}^*$.** To find a way to derive $\mathbf{L}$ such that $\mathbf{L}\mathbf{L}^\mathsf{T} = \boldsymbol{\Sigma}^*$ gives the posterior variance, we start by factoring out the diagonal $\mathbf{V}^{1/2} = \operatorname{diag}(\sqrt{v_1}, \dots, \sqrt{v_K})$. We let $s = \sum_i v_i$ and $c = \frac{1}{\lambda^{-1}+s}$, then $\boldsymbol{\Sigma}^* = \mathbf{V} - c\mathbf{v}\mathbf{v}^\mathsf{T}$.

Factoring out $\mathbf{V}^{1/2}$ yields $\mathbf{L} = \mathbf{V}^{1/2}\mathbf{M}$ for some matrix $\mathbf{M}$. We know the following must hold:

$$\mathbf{L}\mathbf{L}^\mathsf{T} = \mathbf{V}^{1/2}\mathbf{M}\mathbf{M}^\mathsf{T}\mathbf{V}^{1/2} = \boldsymbol{\Sigma}^* = \mathbf{V} - c\mathbf{v}\mathbf{v}^\mathsf{T} = \mathbf{V}^{1/2}[\mathbf{I} - c\mathbf{V}^{1/2}\mathbf{v}\mathbf{v}^\mathsf{T}\mathbf{V}^{1/2}]\mathbf{V}^{1/2}. \tag{31}$$

Defining $\tilde{\mathbf{v}} = \frac{\mathbf{v}}{\sqrt{\mathbf{v}}} = \sqrt{\mathbf{v}}$ component-wise, we have $\mathbf{V}^{1/2}\mathbf{v} = \tilde{\mathbf{v}}$. Combined with Equation (31) we have to find a matrix $\mathbf{M}$ such that

$$\mathbf{M}\mathbf{M}^\mathsf{T} = \mathbf{I} - c\tilde{\mathbf{v}}\tilde{\mathbf{v}}^\mathsf{T}. \tag{32}$$

Since $\mathbf{I} - c\tilde{\mathbf{v}}\tilde{\mathbf{v}}^\mathsf{T}$ is symmetric and rank-one, we assume $\mathbf{M}$ is of the same form

$$\mathbf{M} = \mathbf{I} - \gamma\tilde{\mathbf{v}}\tilde{\mathbf{v}}^\mathsf{T}, \tag{33}$$

for some scalar $\gamma \in \mathbb{R}$. Squaring $\mathbf{M}$, we get

$$\mathbf{M}^2 = (\mathbf{I} - \gamma\tilde{\mathbf{v}}\tilde{\mathbf{v}}^\mathsf{T})^2 = \mathbf{I} - 2\gamma\tilde{\mathbf{v}}\tilde{\mathbf{v}}^\mathsf{T} + \gamma^2(\tilde{\mathbf{v}}^\mathsf{T}\tilde{\mathbf{v}})\tilde{\mathbf{v}}\tilde{\mathbf{v}}^\mathsf{T} = \mathbf{I} - (2\gamma + \gamma^2\tilde{\mathbf{v}}^\mathsf{T}\tilde{\mathbf{v}})\tilde{\mathbf{v}}\tilde{\mathbf{v}}^\mathsf{T}, \tag{34}$$

where we used that $(\tilde{\mathbf{v}}\tilde{\mathbf{v}}^\mathsf{T})\tilde{\mathbf{v}}\tilde{\mathbf{v}}^\mathsf{T} = \tilde{\mathbf{v}}(\tilde{\mathbf{v}}^\mathsf{T}\tilde{\mathbf{v}})\tilde{\mathbf{v}}^\mathsf{T} = (\tilde{\mathbf{v}}^\mathsf{T}\tilde{\mathbf{v}})\tilde{\mathbf{v}}\tilde{\mathbf{v}}^\mathsf{T}$ as $(\tilde{\mathbf{v}}^\mathsf{T}\tilde{\mathbf{v}})$ is a scalar. It remains to match the coefficients of $\tilde{\mathbf{v}}\tilde{\mathbf{v}}^\mathsf{T}$ in Equation (32) and Equation (34) $c$ and $\gamma$, respectively.

This problem is quadratic in $\gamma$, so we get

$$2\gamma + \gamma^2\tilde{\mathbf{v}}^\mathsf{T}\tilde{\mathbf{v}} - c = 0 \implies \gamma = \frac{1 \pm \sqrt{1 - \tilde{\mathbf{v}}^\mathsf{T}\tilde{\mathbf{v}}c}}{\tilde{\mathbf{v}}^\mathsf{T}\tilde{\mathbf{v}}}. \tag{35}$$

Simplifying $\tilde{\mathbf{v}}^\mathsf{T}\tilde{\mathbf{v}} = \sum_i v_i = s$, we get

$$\gamma = \frac{1}{s}\left(1 \pm \sqrt{\frac{\lambda^{-1}}{\lambda^{-1} + s}}\right). \tag{36}$$

From the two possible roots, we pick the negative one and call it $\beta$, since then as $c \to 0$, we have $\mu \to \frac{1}{s}(1 - 1) = 0$, which implies $\mathbf{M} \to \mathbf{I}$ and therefore $\mathbf{L} \to \mathbf{V}^{1/2}$.

We can derive $\mathbf{L}$ efficient as

$$\mathbf{L} = \mathbf{V}^{1/2}\mathbf{M} = \mathbf{V}^{1/2}(\mathbf{I} - \beta\tilde{\mathbf{v}}\tilde{\mathbf{v}}^\mathsf{T}) = \mathbf{V}^{1/2} - \beta\mathbf{v}\tilde{\mathbf{v}}^\mathsf{T}. \tag{37}$$

**Efficient sampling from the probability path.** Sampling from the probability involves $\mathbf{L}_t = (1 - t)\mathbf{I} + t\mathbf{L}_1$ and thus requires computing $\mathbf{L}_t\boldsymbol{\varepsilon}$. Using the $\mathbf{L}$ identity from Equation (37) above, we can element-wise define a vector $\mathbf{d}_t$ with elements

$$[\mathbf{d}_t]_i = (1 - t) \cdot 1 + t \cdot \sqrt{v_i}. \tag{38}$$

This enables us to rewrite

$$\mathbf{L}_t = \operatorname{diag}(\mathbf{d}_t) - t\beta\mathbf{v}\tilde{\mathbf{v}}^\mathsf{T}, \tag{39}$$

and, in turn, to efficiently compute

$$\mathbf{L}_t\boldsymbol{\varepsilon} = \operatorname{diag}(\mathbf{d}_t)\boldsymbol{\varepsilon} - t\beta\mathbf{v}\tilde{\mathbf{v}}^\mathsf{T}\boldsymbol{\varepsilon} = \mathbf{d}_t \odot \boldsymbol{\varepsilon} - t\beta\mathbf{v}(\tilde{\mathbf{v}}^\mathsf{T}\boldsymbol{\varepsilon}). \tag{40}$$

**Efficiently computing the time-derivative.** The time-derivative in Equation (28) involves computing $(\mathbf{L}(\mathbf{x}) - \mathbf{I})\boldsymbol{\varepsilon}$. This can be efficiently computed similar to Equation (40) by defining a new vector $\mathbf{d}$ with elements

$$[\mathbf{d}]_i = \sqrt{v_i} - 1, \tag{41}$$

such that

$$\mathbf{L}(\mathbf{x}) - \mathbf{I} = \mathbf{d} - \beta\mathbf{v}\tilde{\mathbf{v}}^\mathsf{T}, \tag{42}$$

With this, we get

$$(\mathbf{L}(\mathbf{x}) - \mathbf{I})\boldsymbol{\varepsilon} = \mathbf{d} \odot \boldsymbol{\varepsilon} - \beta\mathbf{v}(\tilde{\mathbf{v}}^\mathsf{T}\boldsymbol{\varepsilon}). \tag{43}$$

**Inverting $\mathbf{L}_t$.** Lastly, computing the score $\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t|\mathbf{x}, y) = -\mathbf{L}_t^{-1}\boldsymbol{\varepsilon}$ requires inverting $\mathbf{L}_t$.

Using Equation (39), which is a rank-one update of $\mathbf{d}_t$, we can apply the Sherman-Morrison formula. Let $\mathbf{D} = \mathrm{diag}(\mathbf{d}_t)$ for brevity, then

$$\mathbf{L}_t^{-1} = (\mathbf{D} - t\beta\mathbf{v}\tilde{\mathbf{v}}^\mathsf{T})^{-1} = \mathbf{D}^{-1} + \frac{\mathbf{D}^{-1}\tilde{\mathbf{v}}t\beta\mathbf{v}^\mathsf{T}\mathbf{D}^{-1}}{1 - t\beta\mathbf{v}^\mathsf{T}\mathbf{D}^{-1}\tilde{\mathbf{v}}}. \tag{44}$$

We can now use that $\mathbf{D}^{-1}\tilde{\mathbf{v}} = \mathbf{1}$ and therefore $t\beta\mathbf{v}^\mathsf{T}\mathbf{D}^{-1}\tilde{\mathbf{v}} = t\beta\mathbf{v}^\mathsf{T}\mathbf{1}$. Also, we have $t\beta\mathbf{v}^\mathsf{T}\mathbf{D}^{-1}\boldsymbol{\varepsilon} = t\beta\tilde{\mathbf{v}}^\mathsf{T}\boldsymbol{\varepsilon}$ such that

$$\mathbf{L}_t^{-1}\boldsymbol{\varepsilon} = \frac{\boldsymbol{\varepsilon}}{\tilde{\mathbf{v}}} + \mathbf{1}\frac{t\beta\tilde{\mathbf{v}}^\mathsf{T}\boldsymbol{\varepsilon}}{1 - t\beta\mathbf{v}^\mathsf{T}\mathbf{1}}, \tag{45}$$

where division and multiplication are understood to be element-wise.

### A.5 Details on the adjustments for binary classification

To adapt DAM to the binary classification task, we utilize a probit model specification. Let $p(\mathbf{z}_1|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{V})$ as in Equation (9). The probit model is defined as

$$y^* = \mathcal{N}(\alpha + \mathbf{1}^\mathsf{T}\mathbf{z}_1, 1), \qquad \text{with } y = \mathbb{I}(y^* > 0), \tag{46}$$

where $y^*$ represents some latent utility. Since $\mathbf{z}_1$ is Gaussian, we get

$$p(y^*|\mathbf{x}) = \mathcal{N}(\alpha + \mathbf{1}^\mathsf{T}\mathbf{z}_1, 1 + \mathbf{1}^\mathsf{T}\mathbf{v}). \tag{47}$$

**Likelihood.** The success probability in a probit model is readily obtained as

$$p(y = 1|\mathbf{x}) = p(y^* > 0|\mathbf{x}) = \Phi\left(\frac{\alpha + \mathbf{1}^\mathsf{T}\mathbf{z}_1}{\sqrt{1 + \mathbf{1}^\mathsf{T}\mathbf{v}}}\right) \tag{48}$$

Letting $\mathrm{sign}(y) = 2y - 1$, the likelihood contribution of a single observation is given by

$$p(y|\mathbf{x}) = \Phi\left(\frac{\mathrm{sign}(y)(\alpha + \mathbf{1}^\mathsf{T}\mathbf{z}_1)}{\sqrt{1 + \mathbf{1}^\mathsf{T}\mathbf{v}}}\right). \tag{49}$$

We use this likelihood to train $m_i$ and $v_i$ in stage 1. The loss is given by the NLL

$$\mathcal{L}_{\text{cal}}^{\text{probit}} = -\log\Phi\left(\frac{\mathrm{sign}(y)\mu(\mathbf{x})}{\sqrt{1 + \mathbf{1}^\mathsf{T}\mathbf{v}}}\right). \tag{50}$$

**Closed-form Gaussian posterior.** To form the closed-form posterior and minimize the changes necessary relative to the regression setup, we use the exact, closed-form posterior moments to construct a Gaussian approximation of the probit posterior.

First, we note that of we combine the same prior $p(\mathbf{z}_1|\mathbf{x}) = \mathcal{N})\boldsymbol{\mu}, \mathbf{V})$ as in Equation (9) with probit likelihood, the posterior distribution $q(\mathbf{z}_1|\mathbf{x}, y)$ is known to be the unified skew-normal distribution [7]. Therefore, letting the scalar $s = \sqrt{1 + \mathbf{1}^\mathsf{T}\mathbf{V}\mathbf{1}} = \sqrt{1 + \mathbf{1}^\mathsf{T}\mathbf{v}}$, the moment generating function is known to be

$$M(\mathbf{t}) = \exp(\mathbf{m}^\mathsf{T}\mathbf{t} + 0.5\mathbf{t}^\mathsf{T}\mathbf{V}\mathbf{t})\frac{\Phi(c + \mathrm{sign}(y)s^{-1}\mathbf{1}^\mathsf{T}\mathbf{V}\mathbf{t})}{\Phi(c)}, \quad c = \frac{\mathrm{sign}(y)(\alpha + \mathbf{1}^\mathsf{T}\mathbf{m})}{\sqrt{1 + \mathbf{1}^\mathsf{T}\mathbf{V}\mathbf{1}}}. \tag{51}$$

Taking logs, the cumulant-generating function is

$$K(\mathbf{t}) = \log M(\mathbf{t}) = \mathbf{m}^\mathsf{T}\mathbf{t} + 0.5\mathbf{t}^\mathsf{T}\mathbf{V}\mathbf{t} + \log\Phi(c + \mathrm{sign}(y)s^{-1}\mathbf{1}^\mathsf{T}\mathbf{V}\mathbf{t}) - \log\Phi(c). \tag{52}$$

We can differentiate $K(\mathbf{t})$ wrt $\mathbf{t}$ and evaluate at $\mathbf{t} = 0$ to derive the first two moments. For this, define the inverse Mills ratio as $\lambda(u) = \frac{\psi(u)}{\Psi(u)}$ such that $\frac{\mathrm{d}}{\mathrm{d}u}\log\Phi(u) = \lambda(u)$.

To derive the posterior mean, we evaluate the gradient of $K(\mathbf{t})$ at $\mathbf{t} = \mathbf{0}$. We get

$$\begin{aligned}
\mathbb{E}[\mathbf{z}_1|\mathbf{x}, y] &= \nabla_{\mathbf{t}} K(\mathbf{t})|_{\mathbf{t}=\mathbf{0}} \\
&= [\mathbf{m} + \mathbf{V}\mathbf{t} + \mathrm{sign}(y)s^{-1}\mathbf{V}\mathbf{1}\lambda(c + \mathrm{sign}(y)s^{-1}\mathbf{1}^\mathsf{T}\mathbf{V}\mathbf{t})]_{\mathbf{t}=\mathbf{0}} \\
&= \mathbf{m} + \mathrm{sign}(y)s^{-1}\mathbf{V}\mathbf{1}\lambda(c).
\end{aligned}$$

Plugging in for $s$ and $c$, and noting that $\mathbf{V}$ is diagonal, we get the *feature-wise* posterior means

$$\mathbb{E}[z_1^{(i)}|\mathbf{x}, y] = m_i + \frac{\text{sign}(y)v_i}{\sqrt{1+\mathbf{1}^\mathsf{T}\mathbf{v}}}\lambda(c). \tag{53}$$

Before deriving the posterior variance, we first define the derivative of the inverse Mills ratio. We use that $\phi'(u) = -u\phi(u)$ and $\Phi'(u) = \phi(u)$ to get

$$\begin{aligned}
\lambda'(u) &= \frac{\phi'(u)\Phi(u) - \phi(u)\Phi'(u)}{\Phi(u)^2} \\
&= \frac{-u\phi(u)\Phi(u) - \phi(u)^2}{\Phi(u)^2} \\
&= -u\lambda(u) - \lambda(u)^2 \\
&= -\lambda(u)(u + \lambda(u)).
\end{aligned}$$

Let $\mathbf{b} = \frac{\text{sign}(y)\mathbf{v}}{\sqrt{1+\mathbf{1}^\mathsf{T}\mathbf{v}}}$, then the posterior covariance can be derived from the Hessian:

$$\begin{aligned}
\text{Cov}(\mathbf{z}_1|\mathbf{x}, y) &= \nabla^2_{\mathbf{t}} K(\mathbf{t})|_{\mathbf{t}=\mathbf{0}} \\
&= [\mathbf{V} + \mathbf{b}\mathbf{b}^\mathsf{T}\lambda'(c + \mathbf{b}^\mathsf{T}\mathbf{t})]_{\mathbf{t}=\mathbf{0}} \\
&= \mathbf{V} + \mathbf{b}\mathbf{b}^\mathsf{T}\lambda'(c) \\
&= \mathbf{V} - \mathbf{b}\mathbf{b}^\mathsf{T}\lambda(c)(c + \lambda(c)) \\
&= \mathbf{V} - \frac{\text{sign}(y)^2\mathbf{v}\mathbf{v}^\mathsf{T}}{1 + \sum_j v_j}\lambda(c)(c + \lambda(c)) \\
&= \mathbf{V} - \frac{\mathbf{v}\mathbf{v}^\mathsf{T}}{1 + \sum_j v_j}\lambda(c)(c + \lambda(c)).
\end{aligned}$$

With these two *exact* moments of the non-Gaussian unified skew-normal posterior, we can construct the Gaussian moment-matching approximation:

$$q(\mathbf{z}_1|\mathbf{x}, y) = \mathcal{N}(\mathbb{E}[\mathbf{z}_1|\mathbf{x}, y], \text{Cov}(\mathbf{z}_1|\mathbf{x}, y)), \tag{54}$$

which is structurally very similar to the posterior in the regression case in Equation (9). However, the posterior moments for the probit model are class-dependent, and therefore bimodal, and adjusted by terms determined by the inverse Mills ratio. Similar to the regression case, this definition of $q(\mathbf{z}_1|\mathbf{x}, y)$ gives us closed-form access to the aggregate posterior moments, which serve as target for the aggregate moment matching in Stage 3. It also allows for efficient sampling from the conditional marginal distribution $q(\mathbf{z}_t|\mathbf{x}, y)$ when training the prior SDEs and the efficient computation of the score, similar to the regression case discussed in Appendix A.4. The Cholesky factor of $\text{Cov}(\mathbf{z}_1|\mathbf{x}, y)$ denoted by $\mathbf{L}$ can be constructed very similarly as

$$\mathbf{L} = \mathbf{D} - \beta\mathbf{v}\tilde{\mathbf{v}}^\mathsf{T}, \tag{55}$$

where $\mathbf{D}$, $\mathbf{v}$ and $\tilde{\mathbf{v}}$ are defined as in Appendix A.4. Only the $\beta$ term changes to

$$\beta = \frac{1 - \sqrt{1 - \delta\mathbf{1}^\mathsf{T}\mathbf{v}}}{\mathbf{1}^\mathsf{T}\mathbf{v}} = \frac{\delta}{1 + \sqrt{1 - \delta\mathbf{1}^\mathsf{T}\mathbf{v}}}, \qquad \delta = \frac{\lambda(c)[\lambda(c) + c]}{1 - \mathbf{1}^\mathsf{T}\mathbf{v}}. \tag{56}$$

The other necessary computations then follow akin to the regression case.

### A.6 Detailed Model Setup

**Algorithm 1** DAM Training

**# Stage 1: Calibrate sum and route homoskedastic variance.**
Regression: Train all $m_i$ and $v_i$ for all features using Equation (8).
Classification: Train all $m_i$ and $v_i$ for all features using Equation (50).
Freeze $m_i$ and $v_i$.
Apply noise-sink normalization:
For each $i = 1, \ldots, K$, let $v_i^{\text{homo}} = \min_{x^{(i)} \in p_{\text{data}}(x^{(i)})} v_i(x^{(i)})$.
Remove homoskedastic variance via $v_i^{\text{routed}} = v_i - v_i^{\text{homo}} = v_i^{\text{het}}$ for all $i = 1, \ldots, K$.
Route all $v_i^{\text{homo}}$ into the noise sink: $v_0^{\text{routed}} = v_0 + \sum_{i=1}^{K} v_i^{\text{homo}}$.

**# Stage 2: Derive the closed-form posterior $q(\mathbf{z}_1|\mathbf{x}, y)$.**
Let $\mu = (m_i)_{i=0}^{K}$ and $\mathbf{V} = \text{diag}(\mathbf{v})$ with $\mathbf{v} = (v_i^{\text{routed}})_{i=0}^{K}$.
Regression: Use Proposition 1 to construct the closed-form posterior.
Classification: Use Equation (54) to construct the closed-form posterior.

**# Stage 3: Simplify the posterior distribution.**
Learn the aggregate posterior moments using Equation (11).
Standardize $\mathbf{z}_1 \sim q(\mathbf{z}_1|\mathbf{x}, y)$ using the learned posterior moments.

**# Stage 4: Construct posterior processes and train prior SDEs.**
Construct the drift of the posterior process in Equation (14) using the closed-form time-derivative and score of the conditional marginal distribution $q(\mathbf{z}_t|\mathbf{x}, y)$ from Equation (12).
Train prior drifts $h_{\boldsymbol{\theta}}^{(i)}$ and volatilities $g_{\boldsymbol{\theta}}^{(i)}$ using Equation (15).
Finally, identify the means via centering based on the training set, as described in Section 3.4.

**Algorithm 2** DAM Inference

**# Sample from each $p_{\boldsymbol{\theta}}(z^{(i)}|x^{(i)})$ in parallel**
SDE: Integrate the trained prior SDE from Equation (6) from $t = 0$ to $t = 1$ using the Euler–Maruyama sampler with 200 steps.
ODE: Integrate the trained prior ODE from $t = 0$ to $t = 1$ using the RK4 solver with 50 steps.
Repeat 1000 times per observed $\mathbf{x}$.
If used, undo the standardization from Stage 3.
Apply post-hoch centering as descriebd in Section 3.4.

**# Push samples $\mathbf{z}_1$ forward through $p(y|\mathbf{z})$**
Regression: Use Equation (5).
Classification: Use Equation (49).

#### A.6.1 Stage 1

We parameterize each $m_i$ and $v_i$ with a single feature-wise neural network that produces two outputs $a_i$ and $b_i$. We use an MLP with hidden layers of size [128, 64, 32] and with an embedding dimension of 8. We learn embeddings for categorical features and learnable Fourier features for numerical features, following the state-of-the-art [8]. We let $m_i = a_i$ and $v_i = v_{\text{min}} + \text{sigmoid}(b_i) \cdot v_{\text{max}}$, where $v_{\text{min}} = 0.002^2$ and $v_{\text{max}} = 1.0$, as safeguards. For robustness, we first train the neural works for 100 steps on mean-matching objective with learning rate of 2e-4 and a maximum batch size of 2048, disregarding the variance. Then, we train on the full likelihood objective $\mathcal{L}_{\text{cal}}$ for a maximum of 30000 update steps. We use early stopping based on the validation loss computed after each epoch with a patience of 20 epochs. We reduce the learning rate on plateau of the validation loss by a factor 0.5 and a patience of 5 epochs. We also apply an EMA to the model parameters with decay of 0.95. After training terminates we load the best EMA checkpoint based on validation loss and freeze the parameters.

We emphasize that $\text{DAM}_{\text{moments}}$ follows exactly the same parameterizaton, training procedure and noise sink normalization. NAMLSS exactly follows the training procedure but does note do noise sink normalization and is slightly differently parameterized for classification. Specifically, we let NAMLSS only parameterize the mean using a logit model setup, as is common in the literature, to differentiate it from the $\text{DAM}_{\text{moments}}$ probit implementation. All other hyperparameters are equal.

#### A.6.2 STAGE 2

The posterior is constructed in closed-form using the introduced formulas and the frozen $m_i$ and $v_i$ parameterization from Stage 1. There are no trainable parameters.

#### A.6.3 STAGE 3

We apply the exact same training procedure from Stage 1 to learn $\tilde{m}_i$ and $\tilde{v}_i$. These are parameterized by mostly the same feature-wise neural network architecture that outputs scalars $a_i$ and $b_i$. However, we utilize the fact that the learned $m_i$ and $v_i$ can be expected to be relatively close to the aggregated posterior mean and variance. Therefore, we learn an additive adjustment for the mean, such that $\tilde{m}_i = m_i + a_i$. For the variance, we learn a non-negative scaling factor such that $\tilde{v}_i = v_i \exp(b_i)$.

#### A.6.4 STAGE 4

We again use the same feature-wise neural network architecture as before to parameterize the prior SDE drift and preserve the interpretability of the generative process. The inputs to the MLP are now $z_t^{(i)}$, an 8-dimensional time-step embedding and the 8-dimensional embedding associated with $x^{(i)}$. These inputs are pushed through three hidden layers of sizes [128, 128, 64] to then a single, unrestricted output dimension. We minimize $\mathcal{L}_{\text{diff}}$ for a fixed 30000 update steps, using learning rate 2e-4, a linear rate decay, and take an EMA of the parameters with decay = 0.999.

The volatilities are trained by the same optimizer but are parameterized by a smaller MLP with hidden layers [64, 64, 64]. It takes a time embedding of dimension 8 as the input and its scalar output is transformed by a softplus function to ensure non-negativity.

### A.7 DATASETS

We compare model performance on 13 classification datasets and 9 regression datasets summarized in Table 4. These are derived from prior work on additive models [6] and the BeyondArena benchmark [18]. For brevity, we renamed some datasets. To avoid inference issues with GAMLSS and gamboostLSS, which do not function on unseen categories in the test set, we pool categories with fewer than 20 observations in a "rare" category. We standardize numerical features and ordinal encode categorical features, except for CARD which requires one-hot encoded features. Any rows with missings are removed.

Table 4: Benchmark datasets. We indicate the original and renamed dataset names.

| Dataset | Renamed | Task | Observations | Features |
|---|---|---|---|---|
| adult | - | class. | 45175 | 14 |
| bad_customer_detection | bad_customer | class. | 1723 | 13 |
| bank_customer_churn | bank_churn | class. | 10000 | 10 |
| bank_marketing | - | class. | 45211 | 13 |
| churn | - | class. | 5000 | 19 |
| credit_card_clients_default | credit_card | class. | 30000 | 23 |
| customer_satisfaction_in_airline | airline | class. | 129487 | 21 |
| fico | - | class. | 9871 | 23 |
| fitness_club | - | class. | 1480 | 6 |
| hr_analytics | - | class. | 8955 | 12 |
| iranian_churn | - | class. | 2850 | 13 |
| kin8nm | - | class. | 8192 | 8 |
| online_shoppers_purchasing_intention_dataset | shoppers | class. | 12330 | 17 |
| beijing | - | regression | 41757 | 11 |
| concrete | - | regression | 1030 | 8 |
| diamonds | - | regression | 53940 | 9 |
| food_delivery_time | food | regression | 45451 | 9 |
| healthcare_insurance_expenses | healthcare | regression | 1338 | 6 |
| housing | - | regression | 20433 | 9 |
| insurance | - | regression | 1338 | 6 |
| miami_housing | miami | regression | 13776 | 15 |
| protein | - | regression | 45730 | 9 |

### A.8 Benchmark models

**GAMLSS [20].** We utilize the official R package of the same name. We parameterize the feature contributions to the mean penalized basis splines if the numerical features has more than five unique values. The packages includes its own routine of automatically determining the appropriate degrees of freedom. Categorical and numerical features with less than five unique values, we model linearly. Also, since we encountered severe convergence issues on regression datasets, we parameterize all feature contributions determining the log scale only linearly. As suggested by the authors of the package, we use the Rigby and Stasinopoulos (RS) algorithm for the first 20 steps before switching to the Cole and Green (CG) algorithm for a maximum of 3980 steps. Further, we reduce the step size for the mean parameters to 0.5 and for the scale parameters to 0.3, to limit the possibility of the algorithms overshooting. Otherwise we rely on the default hyperparameters of the package.

**gamboostLSS [16].** We utilize the official R package of the same name, use the noncyclic fitting routine and tune the mstop hyperparameter 3-fold CV with a minimum of 20 and maximum of 5000. For training (and tuning), we use 24 AMD Genoa CPUs and conduct tuning iterations in parallel. We parameterize each feature contribution of numerical features with less than five unique values and categorical features linearly without an intercept. Feature contributions of numerical features with more unique values are parameterized using penalized regressions splines. Otherwise we rely on the default hyperparameters of the package.

**NAMLSS [27].** We implement our own variant of the NAMLSS$^2$ variant, where each feature net predicts the distributional parameters jointly. This allows us to align NAMLSS as best as possible with $\text{DAM}_{\text{moments}}$ and DAM. Unlike the usual setup, we therefore parameterize the mean and variance of Gaussian distribution, instead of the log scale, similar to $\text{DAM}_{\text{moments}}$ but without applying the noise sink normalization. For classification, we parameterize the log odds of a logit model, unlike DAM, which is based on a probit specification. Otherwise, all architectural choices, the training procedure with mean warmup, etc. mirror those of $\text{DAM}_{\text{moments}}$

**CARD [9].** We adopt the code published by the authors at `https://github.com/XzwHan/CARD`. We mostly use default hyperparameters but increase the batch size and align the total number of training steps with DAM. To align CARD further with the DAM stage-wise procedure, we first pre-train a small neural network which is used to define the mean of the source distribution [9].

### A.9 Results on Synthetic Data

To investigate qualitatively how well $\text{DAM}_{\text{moments}}$, DAM and $\text{DAM}_{\text{ODE}}$ can recover the ground-truth feature contributions, we rely on synthetic data from three different data generating processes (DGPs). To set the right expectation, we first emphasize that in DAM only the mean (via post-hoc centering) and the variances (via noise sink normalization) are identified. This leaves higher-order moments unidentified. A perfect recovery of any given ground-truth distribution of feature contributions is therefore impossible.

#### A.9.1 DGP construction

We consider three DGPs. Each consists of three numerical features and the noise sink. We construct them based on the assumption underlying the noise sink normalization. That is, we compose the true $\text{Var}(z^{(i)}|x^{(i)})$ additively with a homoskedastic part and a heteroskedastic part.

We generate each feature from a uniform distribution, i.e., $x^{(i)} \sim \mathcal{U}(0,1)$. The outcome is defined by an additive model in terms of the feature contributions $z^{(i)}$:

$$y = z^{(0)} + \sum_{i=1}^{K} z^{(i)}, \tag{57}$$

where $z^{(0)}$ is the noise sink, which we fix to $z^{(0)} = 0.2\epsilon^{(0)}$ with $\epsilon_0 \sim \mathcal{N}(0,1)$. The first three feature contributions are generated using the common structure

$$z^{(i)} = \mu_i(x^{(i)}) + \underbrace{0.1\eta_i}_{\text{homoskedastic}} + \underbrace{a_i h_i(x^{(i)})\epsilon_i}_{\text{heteroskedastic}}, \quad \eta_i \sim \mathcal{N}(0,1), \quad \epsilon_i \sim \mathcal{N}(0,1), \text{for } i = 1,2,3. \tag{58}$$

Here, the constant noise floor is given by $\text{Var}(0.1\eta_i) = 0.1^2\text{Var}(\eta_i) = 0.01$, independent of $x^{(i)}$. We define $\mathbf{a} = (0.35, 0.4, 0.45)$ and the heteroskedastic standard deviations as

$$\begin{aligned} h_1(x^{(1)}) &= x^{(1)} \\ h_2(x^{(2)}) &= 1 - x^{(2)} \\ h_3(x^{(3)}) &= \frac{\exp(-8(x-0.5)^2) - \exp(-2)}{1 - \exp(-2)}, \end{aligned}$$

such that for generated $x^{(3)} \in [0,1]$, we have $h_3(0) = h_3(1) = 0$ at the boundaries and $h_3(0.5) = 1$ in the middle, leading to an inverted U-shape. The goal of noise sink normalization is to move the homoskedastic variance floor into the noise sink. Knowing the DGP, we therefore have access to the ground-truth distribution after perfect noise sink normalization:

$$z^{(0,\text{routed})} = z^{(0)} + 0.1\sum_{i=1}^{K} \eta_i, \tag{59}$$

with each feature contribution besides the noise sink changing to

$$z^{(i,\text{routed})} = \mu_i(x^{(i)}) + a_j h_i(x^{(i)})\epsilon_i. \tag{60}$$

These routed contributions distributions are what DAM aims to recover.

**DGP 1: Skewed contribution.** We let $r \sim \mathcal{SN}(0,1,\alpha)$ follow a skew-normal distribution with $\alpha = \frac{\delta}{\sqrt{1-\delta^2}}$ and $\delta = 0.99$. Transforming $r$ into its standardized version using mean $m_\delta = \delta\sqrt{\frac{2}{\pi}}$ and variance $v_\delta = 1 - \frac{2\delta^2}{\pi}$, we get

$$\epsilon_1^{\text{skew}} \sim \mathcal{SN}(-\frac{m_\delta}{\sqrt{v_\delta}}, \frac{1}{\sqrt{v_\delta}}, \alpha), \tag{61}$$

such that $\mathbb{E}[\epsilon_1^{\text{skew}}] = 0$ and $\text{Var}(\epsilon_1^{\text{skew}}) = 1$ is preserved. Replacing $\epsilon_1$ in Equation (58) with $\epsilon_1^{\text{skew}}$ makes $z^{(1)}$ highly skewed, while leaving the first moments unchanged.

**DGP 2: Heavy-tailed contribution.** Instead of adding skewness, DGP 2 adds heavy tails to $z^{(1)}$. We let

$$\epsilon_1^{\text{tail}} = \frac{T_5}{\sqrt{5/3}}, \quad T_5 \sim t_5, \tag{62}$$

where $t_5$ denotes the heavy-tailed Student's-t distribution with five degrees of freedom. This ensures $\mathbb{E}[\epsilon_1^{\text{tail}}] = 0$ and $\text{Var}(\epsilon_1^{\text{tail}}) = 1$ but significantly increases the excess kurtosis to six.

**DGP 3: Adding a bi-modal contribution.** Lastly, we leave the original features from Equation (58) intact but add a fourth, bimodal feature contribution $z^{(4)}$. We define two branches identified by $C \in \{-1, 1\}$ with $p(C = 1) = p(C = -1) = 0.5$. A bi-modal feature can then be constructed as

$$z^{(4)} = \mu(x^{(4)}, C) + 0.1\eta_4 + \sigma(x^{(4)}, C)\epsilon_4, \quad \eta_4 \sim \mathcal{N}(0, 1) \quad \epsilon_4 \sim \mathcal{N}(0, 1), \tag{63}$$

with component-specific mean

$$\mu(x^{(4)}, C) = Cs(x^{(4)}) = C[0.35 \tanh(5(x^{(4)} - 0.5))], \tag{64}$$

and component-specific standard deviation

$$\sigma(x^{(4)}, C) = \begin{cases} 0.25x^{(4)}, & C = 1 \\ 0.25(1 - x^{(4)}), & C = -1. \end{cases} \tag{65}$$

Hence, the component noises remain Gaussian but the distribution is a bi-modal Gaussian mixture.

### A.9.2 Results

For each DGP, we plot the ground-truth noise sink normalized feature contribution distributions in comparison to the learned distribution from $\text{DAM}_{\text{moments}}$, DAM and $\text{DAM}_{\text{ODE}}$ in Figures 2, 3 and 4 below.

First, it appears that the noise sink normalization is working as intended. All DAM variants learn the ground-truth homoskedastic part reasonable well, leading to similar variance minima in the densities. For DGP 1, we can see in Figure 2 that all models produce $z^{(i)}$ samples that are overdispersed. This overdispersions appears to be slightly lower for DAM and $\text{DAM}_{\text{ODE}}$. But considering also the the densities generated for DGP 2 and 3, the models fail to identify the ground-truth higher-order moments of $z^{(1)}$ for DGP 1 and 2 and $z^{(4)}$ for DGP 4. The bi-modal structure is not recovered at all. This might be caused by the closed-form posterior depending only on the non-mixture means $m_i$ and $v_i$ estimated from the data. Since the allocation of $r(\mathbf{x}, y)$ to each feature contribution depends on weights $\mathbf{w}$, which are proportional to the identified variance $v_i$, this implies that if $v_i$ is not dominating, bi-modal signal is likely to be split across feature contributions instead of the signal going to a single $z^{(i)}$. To conclude, the main results indicate that the closed-form posterior is effective in refining the predictions of a first-stage NAMLSS model. However, these improvements are not driven by the ability to identify non-Gaussian structure highly accurately. Rather, with very limited information (since only the sum of the contributions is observed), the model conservatively splits signal proportional to the estimated variance. As in practice, and in the constructed DGPs, the variances of the feature contributions are moderate and similar to each other, no extreme allocation of signal into a single latent $z^{(i)}$ can occur.

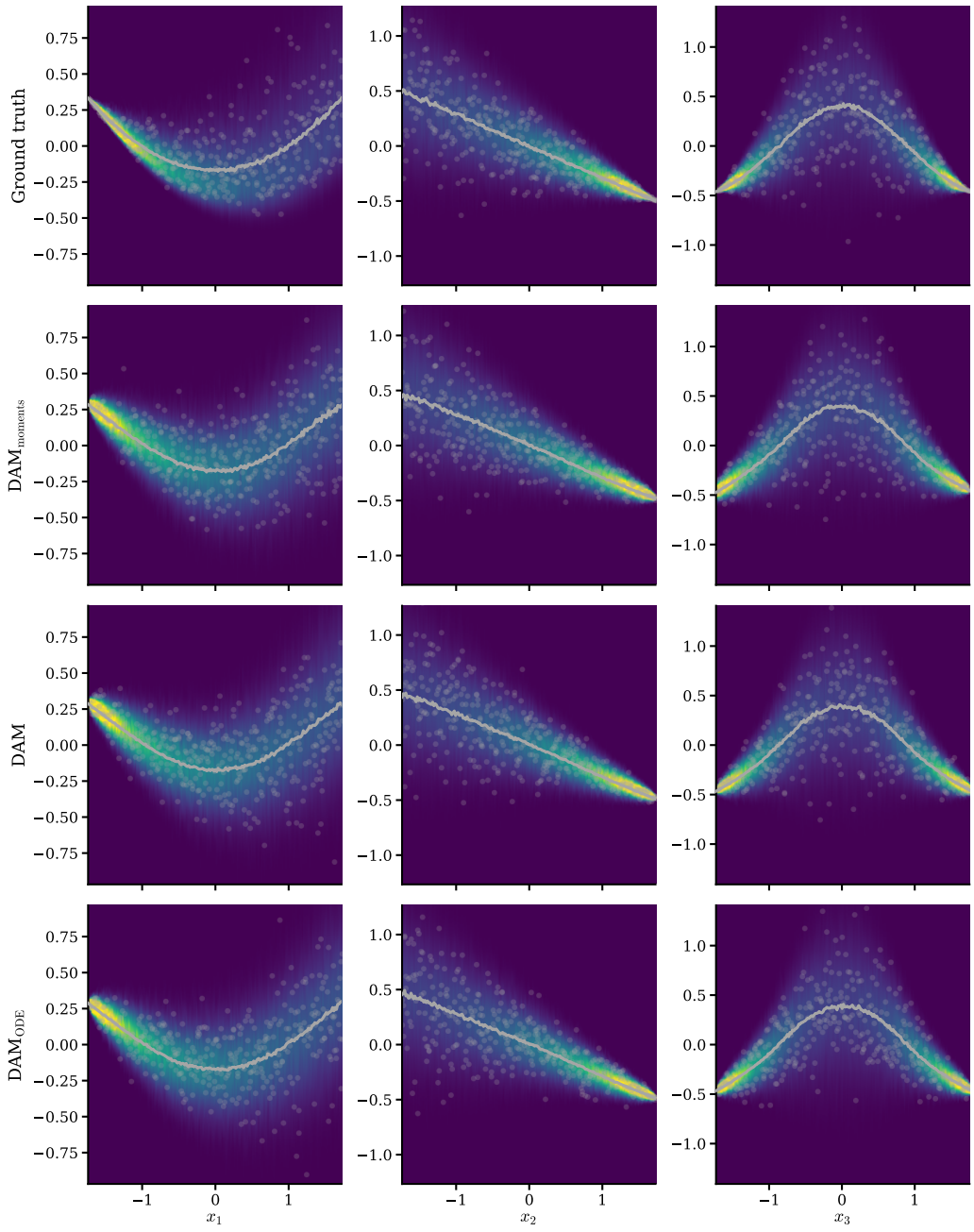



Figure 2: DGP 1: Density plots comparing the ground-truth noise sink normalized distributions of the feature contributes to the distributions generated by different DAM variants

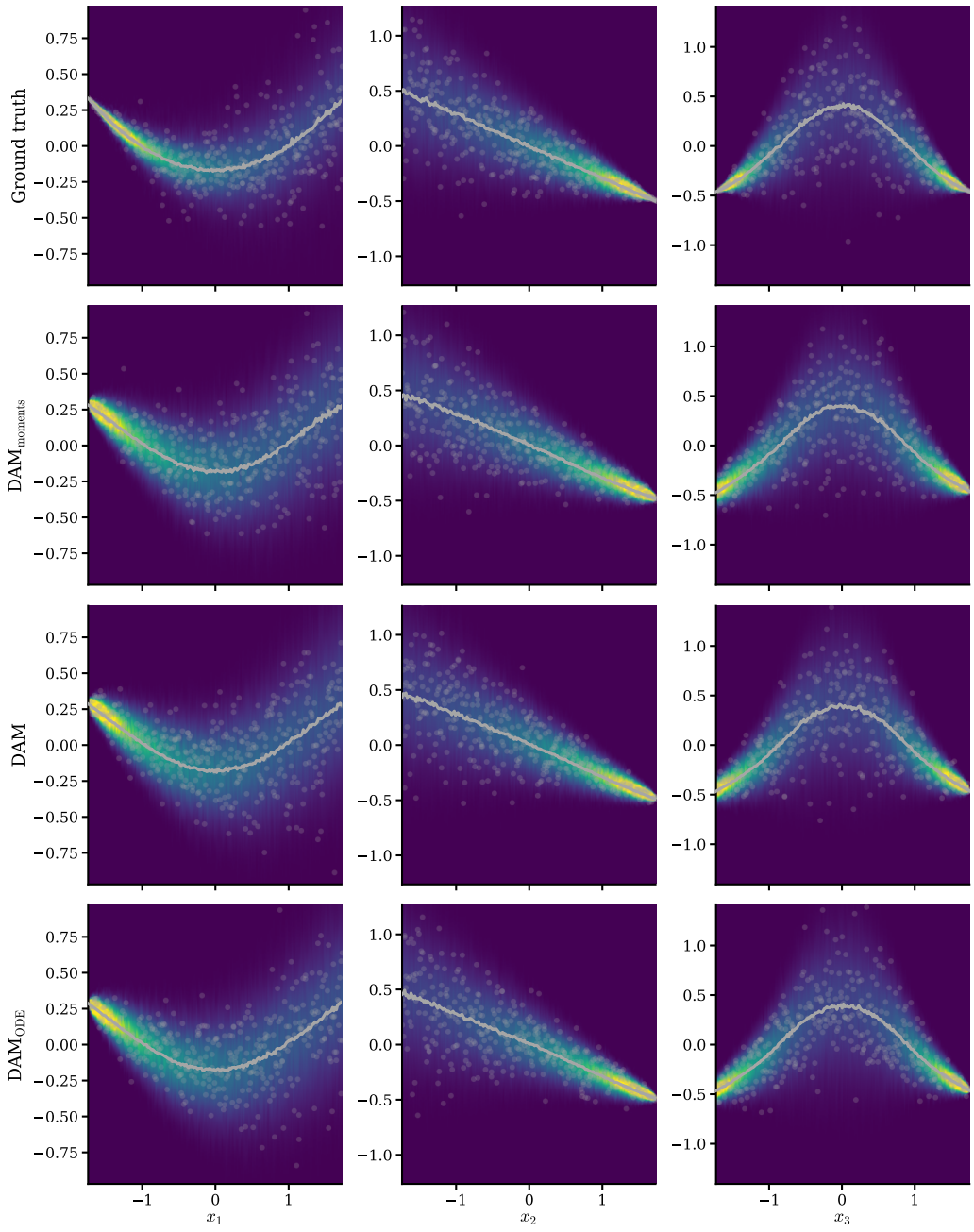



Figure 3: DGP 2: Density plots comparing the ground-truth noise sink normalized distributions of the feature contributes to the distributions generated by different DAM variants

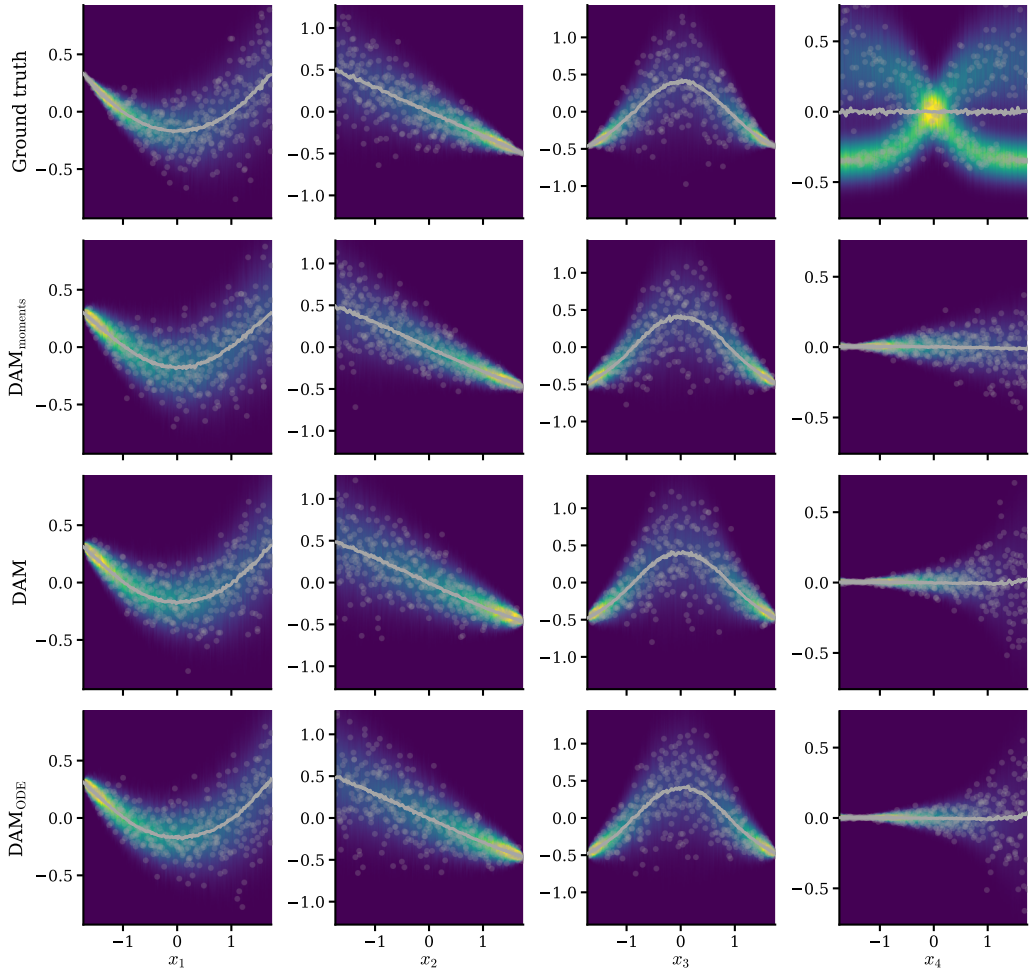



Figure 4: DGP 3: Density plots comparing the ground-truth noise sink normalized distributions of the feature contributes to the distributions generated by different DAM variants

## A.10 Detailed results

### A.10.1 Real data

| Dataset | DAM$_{\text{moments}}$ | DAM$_{\text{ODE}}$ | DAM | NAMLSS | GAMLSS | gamboostLSS | CARD |
|---|---|---|---|---|---|---|---|
| beijing | $0.366 \pm 0.001$ | $0.366 \pm 0.001$ | $0.366 \pm 0.001$ | $0.365 \pm 0.001$ | $0.364 \pm 0.001$ | $0.363 \pm 0.001$ | $\mathbf{0.263 \pm 0.003}$ |
| concrete | $3.520 \pm 0.209$ | $3.310 \pm 0.286$ | $3.422 \pm 0.252$ | $3.583 \pm 0.120$ | $48.559 \pm 31.970$ | $3.400 \pm 0.210$ | $\mathbf{2.927 \pm 0.300}$ |
| diamonds | $0.064 \pm 0.005$ | $0.060 \pm 0.004$ | $0.061 \pm 0.005$ | $0.060 \pm 0.005$ | – | $0.089 \pm 0.001$ | $\mathbf{0.056 \pm 0.006}$ |
| food | $4.452 \pm 0.040$ | $4.445 \pm 0.046$ | $\mathbf{4.445 \pm 0.045}$ | $4.453 \pm 0.045$ | – | – | $4.596 \pm 0.026$ |
| healthcare | $0.225 \pm 0.010$ | $0.213 \pm 0.017$ | $0.214 \pm 0.017$ | $0.221 \pm 0.012$ | $0.224 \pm 0.013$ | $0.236 \pm 0.013$ | $\mathbf{0.134 \pm 0.018}$ |
| housing | $0.148 \pm 0.002$ | $0.146 \pm 0.003$ | $0.146 \pm 0.003$ | $0.150 \pm 0.001$ | $0.150 \pm 0.004$ | $0.156 \pm 0.010$ | $\mathbf{0.121 \pm 0.002}$ |
| insurance | $0.226 \pm 0.010$ | $0.214 \pm 0.011$ | $0.215 \pm 0.011$ | $0.222 \pm 0.012$ | $0.224 \pm 0.011$ | $0.231 \pm 0.011$ | $\mathbf{0.134 \pm 0.012}$ |
| miami | $0.097 \pm 0.002$ | $0.095 \pm 0.001$ | $0.096 \pm 0.002$ | $0.095 \pm 0.002$ | – | $0.120 \pm 0.006$ | $\mathbf{0.073 \pm 0.002}$ |
| protein | $2.762 \pm 0.010$ | $2.752 \pm 0.022$ | $2.755 \pm 0.023$ | $2.780 \pm 0.030$ | – | $2.794 \pm 0.023$ | $\mathbf{1.838 \pm 0.018}$ |

Table 5: CRPS (↓): Average and standard deviation were computed over CV folds. Best result per dataset is indicated in bold.

| Dataset | $\text{DAM}_{\text{moments}}$ | $\text{DAM}_{\text{ODE}}$ | DAM | NAMLSS | GAMLSS | gamboostLSS | CARD |
|---|---|---|---|---|---|---|---|
| beijing | $0.949 \pm 0.003$ | $0.950 \pm 0.003$ | $\mathbf{0.950 \pm 0.003}$ | $0.951 \pm 0.003$ | $0.950 \pm 0.002$ | $0.950 \pm 0.003$ | $0.955 \pm 0.004$ |
| concrete | $0.946 \pm 0.029$ | $\mathbf{0.949 \pm 0.031}$ | $0.943 \pm 0.030$ | $0.938 \pm 0.025$ | $0.904 \pm 0.041$ | $0.923 \pm 0.031$ | $0.906 \pm 0.028$ |
| diamonds | $0.952 \pm 0.007$ | $0.951 \pm 0.005$ | $\mathbf{0.950 \pm 0.004}$ | $0.949 \pm 0.003$ | – | $0.958 \pm 0.003$ | $0.972 \pm 0.007$ |
| food | $0.942 \pm 0.003$ | $0.943 \pm 0.002$ | $0.943 \pm 0.002$ | $0.942 \pm 0.003$ | – | – | $\mathbf{0.953 \pm 0.007}$ |
| healthcare | $0.932 \pm 0.018$ | $\mathbf{0.943 \pm 0.016}$ | $0.942 \pm 0.017$ | $0.928 \pm 0.021$ | $0.927 \pm 0.020$ | $0.935 \pm 0.021$ | $0.936 \pm 0.018$ |
| housing | $0.945 \pm 0.004$ | $0.945 \pm 0.005$ | $0.944 \pm 0.004$ | $0.947 \pm 0.004$ | $0.942 \pm 0.003$ | $0.948 \pm 0.005$ | $\mathbf{0.951 \pm 0.006}$ |
| insurance | $0.938 \pm 0.019$ | $\mathbf{0.942 \pm 0.016}$ | $0.939 \pm 0.018$ | $0.930 \pm 0.007$ | $0.923 \pm 0.014$ | $0.933 \pm 0.014$ | $0.933 \pm 0.014$ |
| miami | $0.948 \pm 0.005$ | $0.951 \pm 0.006$ | $0.949 \pm 0.004$ | $\mathbf{0.951 \pm 0.004}$ | – | $0.952 \pm 0.008$ | $0.944 \pm 0.006$ |
| protein | $\mathbf{0.956 \pm 0.001}$ | $0.958 \pm 0.002$ | $0.958 \pm 0.002$ | $0.959 \pm 0.002$ | – | $0.959 \pm 0.002$ | $0.968 \pm 0.003$ |

Table 6: PICP (closer to 0.95 is better): Average and standard deviation were computed over CV folds. Best result per dataset is indicated in bold.

| Dataset | $\text{DAM}_{\text{moments}}$ | $\text{DAM}_{\text{ODE}}$ | DAM | NAMLSS | GAMLSS | gamboostLSS | CARD |
|---|---|---|---|---|---|---|---|
| beijing | $0.006 \pm 0.001$ | $0.007 \pm 0.001$ | $\mathbf{0.006 \pm 0.001}$ | $0.007 \pm 0.001$ | $0.006 \pm 0.000$ | $0.007 \pm 0.001$ | $0.012 \pm 0.002$ |
| concrete | $0.019 \pm 0.006$ | $0.018 \pm 0.003$ | $\mathbf{0.018 \pm 0.004}$ | $0.023 \pm 0.004$ | $0.028 \pm 0.008$ | $0.021 \pm 0.005$ | $0.019 \pm 0.005$ |
| diamonds | $0.007 \pm 0.001$ | $0.007 \pm 0.001$ | $\mathbf{0.006 \pm 0.002}$ | $0.006 \pm 0.002$ | – | $0.007 \pm 0.001$ | $0.022 \pm 0.003$ |
| food | $0.016 \pm 0.001$ | $0.015 \pm 0.001$ | $0.015 \pm 0.001$ | $0.016 \pm 0.001$ | – | – | $\mathbf{0.006 \pm 0.001}$ |
| healthcare | $0.068 \pm 0.009$ | $0.064 \pm 0.008$ | $0.066 \pm 0.008$ | $0.074 \pm 0.005$ | $0.073 \pm 0.003$ | $0.062 \pm 0.005$ | $\mathbf{0.060 \pm 0.006}$ |
| housing | $0.011 \pm 0.003$ | $0.011 \pm 0.002$ | $0.011 \pm 0.002$ | $\mathbf{0.010 \pm 0.002}$ | $0.013 \pm 0.003$ | $0.012 \pm 0.003$ | $0.019 \pm 0.004$ |
| insurance | $0.071 \pm 0.008$ | $0.057 \pm 0.008$ | $0.059 \pm 0.009$ | $0.072 \pm 0.005$ | $0.075 \pm 0.007$ | $0.065 \pm 0.004$ | $\mathbf{0.057 \pm 0.009}$ |
| miami | $0.008 \pm 0.002$ | $0.008 \pm 0.002$ | $0.008 \pm 0.002$ | $\mathbf{0.008 \pm 0.002}$ | – | $0.009 \pm 0.003$ | $0.019 \pm 0.003$ |
| protein | $0.026 \pm 0.001$ | $0.024 \pm 0.001$ | $0.024 \pm 0.002$ | $0.030 \pm 0.001$ | – | $0.031 \pm 0.001$ | $\mathbf{0.024 \pm 0.001}$ |

Table 7: QICE coverage (↓): Average and standard deviation were computed over CV folds. Best result per dataset is indicated in bold.

| Dataset | $\text{DAM}_{\text{moments}}$ | $\text{DAM}_{\text{ODE}}$ | DAM | NAMLSS | GAMLSS | gamboostLSS | CARD |
|---|---|---|---|---|---|---|---|
| adult | $0.913 \pm 0.002$ | $0.913 \pm 0.002$ | $0.913 \pm 0.002$ | $0.913 \pm 0.002$ | $\mathbf{0.915 \pm 0.002}$ | $0.914 \pm 0.002$ | $0.902 \pm 0.007$ |
| airline | $0.969 \pm 0.001$ | $0.969 \pm 0.001$ | $0.969 \pm 0.001$ | $0.964 \pm 0.001$ | $0.965 \pm 0.001$ | $0.965 \pm 0.001$ | $\mathbf{0.993 \pm 0.001}$ |
| bad_customer | $\mathbf{0.754 \pm 0.029}$ | $0.747 \pm 0.023$ | $0.749 \pm 0.024$ | $0.741 \pm 0.037$ | $0.750 \pm 0.033$ | $0.745 \pm 0.038$ | $0.685 \pm 0.035$ |
| bank_churn | $\mathbf{0.851 \pm 0.009}$ | $0.850 \pm 0.008$ | $0.850 \pm 0.009$ | $0.849 \pm 0.009$ | $0.799 \pm 0.011$ | $0.799 \pm 0.011$ | $0.825 \pm 0.016$ |
| bank_marketing | $0.759 \pm 0.007$ | $0.758 \pm 0.006$ | $0.757 \pm 0.006$ | $0.756 \pm 0.008$ | $\mathbf{0.760 \pm 0.008}$ | $0.757 \pm 0.006$ | $0.674 \pm 0.055$ |
| churn | $0.880 \pm 0.014$ | $0.881 \pm 0.014$ | $0.881 \pm 0.013$ | $0.878 \pm 0.016$ | $0.879 \pm 0.013$ | $0.880 \pm 0.013$ | $\mathbf{0.898 \pm 0.012}$ |
| credit_card | $\mathbf{0.780 \pm 0.005}$ | $0.780 \pm 0.006$ | $0.780 \pm 0.006$ | $0.779 \pm 0.005$ | $0.777 \pm 0.005$ | $0.776 \pm 0.006$ | $0.760 \pm 0.007$ |
| fico | $0.803 \pm 0.011$ | $0.804 \pm 0.012$ | $0.804 \pm 0.012$ | $0.802 \pm 0.011$ | $\mathbf{0.810 \pm 0.008}$ | $0.809 \pm 0.010$ | $0.701 \pm 0.053$ |
| fitness_club | $0.818 \pm 0.033$ | $0.818 \pm 0.031$ | $0.819 \pm 0.033$ | $0.820 \pm 0.033$ | $0.820 \pm 0.031$ | $\mathbf{0.821 \pm 0.032}$ | $0.787 \pm 0.040$ |
| hr_analytics | $0.762 \pm 0.012$ | $0.765 \pm 0.010$ | $0.768 \pm 0.009$ | $0.761 \pm 0.016$ | $\mathbf{0.769 \pm 0.010}$ | $0.766 \pm 0.018$ | $0.696 \pm 0.024$ |
| iranian_churn | $0.962 \pm 0.023$ | $0.961 \pm 0.020$ | $0.961 \pm 0.020$ | $0.960 \pm 0.019$ | $0.977 \pm 0.007$ | $0.963 \pm 0.008$ | $\mathbf{0.981 \pm 0.007}$ |
| kin8nm | $0.830 \pm 0.012$ | $0.834 \pm 0.012$ | $0.834 \pm 0.012$ | $0.830 \pm 0.013$ | $0.831 \pm 0.013$ | $0.831 \pm 0.013$ | $\mathbf{0.964 \pm 0.005}$ |
| shoppers | $\mathbf{0.921 \pm 0.006}$ | $0.919 \pm 0.005$ | $0.919 \pm 0.005$ | $0.916 \pm 0.007$ | $0.914 \pm 0.008$ | $0.913 \pm 0.006$ | $0.870 \pm 0.022$ |

Table 8: AUROC (↑): Average and standard deviation were computed over CV folds. Best result per dataset is indicated in bold.

| Dataset | $\text{DAM}_{\text{moments}}$ | $\text{DAM}_{\text{ODE}}$ | DAM | NAMLSS | GAMLSS | gamboostLSS | CARD |
|---|---|---|---|---|---|---|---|
| adult | $0.099 \pm 0.001$ | $0.099 \pm 0.001$ | $0.099 \pm 0.001$ | $0.099 \pm 0.001$ | $\mathbf{0.098 \pm 0.001}$ | $0.099 \pm 0.001$ | $0.143 \pm 0.004$ |
| airline | $\mathbf{0.068 \pm 0.002}$ | $0.068 \pm 0.002$ | $0.068 \pm 0.002$ | $0.072 \pm 0.002$ | $0.071 \pm 0.002$ | $0.071 \pm 0.002$ | $0.095 \pm 0.003$ |
| bad_customer | $\mathbf{0.093 \pm 0.002}$ | $0.093 \pm 0.002$ | $0.093 \pm 0.001$ | $0.094 \pm 0.002$ | $0.093 \pm 0.003$ | $0.093 \pm 0.002$ | $0.133 \pm 0.006$ |
| bank_churn | $\mathbf{0.107 \pm 0.002}$ | $0.107 \pm 0.001$ | $0.107 \pm 0.001$ | $0.108 \pm 0.002$ | $0.123 \pm 0.002$ | $0.124 \pm 0.002$ | $0.166 \pm 0.026$ |
| bank_marketing | $0.088 \pm 0.001$ | $0.088 \pm 0.001$ | $0.088 \pm 0.001$ | $0.088 \pm 0.001$ | $\mathbf{0.087 \pm 0.001}$ | $0.088 \pm 0.001$ | $0.184 \pm 0.029$ |
| churn | $0.081 \pm 0.003$ | $0.080 \pm 0.003$ | $\mathbf{0.080 \pm 0.003}$ | $0.082 \pm 0.004$ | $0.081 \pm 0.004$ | $0.081 \pm 0.004$ | $0.135 \pm 0.017$ |
| credit_card | $\mathbf{0.135 \pm 0.002}$ | $0.135 \pm 0.002$ | $0.135 \pm 0.002$ | $0.135 \pm 0.002$ | $0.135 \pm 0.002$ | $0.136 \pm 0.002$ | $0.182 \pm 0.004$ |
| fico | $0.181 \pm 0.005$ | $0.180 \pm 0.005$ | $0.180 \pm 0.006$ | $0.181 \pm 0.005$ | $\mathbf{0.177 \pm 0.004}$ | $0.178 \pm 0.005$ | $0.220 \pm 0.015$ |
| fitness_club | $0.152 \pm 0.011$ | $0.153 \pm 0.011$ | $0.152 \pm 0.011$ | $0.152 \pm 0.011$ | $0.152 \pm 0.011$ | $\mathbf{0.152 \pm 0.011}$ | $0.191 \pm 0.009$ |
| hr_analytics | $0.107 \pm 0.001$ | $0.107 \pm 0.001$ | $\mathbf{0.107 \pm 0.001}$ | $0.107 \pm 0.001$ | $0.108 \pm 0.002$ | $0.107 \pm 0.001$ | $0.151 \pm 0.006$ |
| iranian_churn | $0.051 \pm 0.012$ | $0.052 \pm 0.012$ | $0.052 \pm 0.012$ | $0.054 \pm 0.011$ | $\mathbf{0.041 \pm 0.006}$ | $0.052 \pm 0.005$ | $0.108 \pm 0.009$ |
| kin8nm | $0.168 \pm 0.006$ | $0.166 \pm 0.006$ | $0.166 \pm 0.006$ | $0.168 \pm 0.006$ | $0.168 \pm 0.006$ | $0.168 \pm 0.006$ | $\mathbf{0.119 \pm 0.006}$ |
| shoppers | $\mathbf{0.074 \pm 0.002}$ | $0.075 \pm 0.003$ | $0.075 \pm 0.003$ | $0.075 \pm 0.002$ | $0.075 \pm 0.003$ | $0.077 \pm 0.002$ | $0.139 \pm 0.012$ |

Table 9: Brier score (↓): Average and standard deviation were computed over CV folds. Best result per dataset is indicated in bold.

| Dataset | $\text{DAM}_{\text{moments}}$ | $\text{DAM}_{\text{ODE}}$ | DAM | NAMLSS | GAMLSS | gamboostLSS | CARD |
|---|---|---|---|---|---|---|---|
| adult | $0.007 \pm 0.002$ | $0.008 \pm 0.002$ | $0.007 \pm 0.002$ | $\mathbf{0.007 \pm 0.002}$ | $0.008 \pm 0.002$ | $0.009 \pm 0.001$ | $0.178 \pm 0.006$ |
| airline | $0.006 \pm 0.002$ | $0.006 \pm 0.001$ | $0.006 \pm 0.002$ | $0.008 \pm 0.001$ | $\mathbf{0.005 \pm 0.001}$ | $0.008 \pm 0.001$ | $0.247 \pm 0.004$ |
| bad_customer | $0.024 \pm 0.009$ | $0.025 \pm 0.006$ | $0.027 \pm 0.005$ | $\mathbf{0.023 \pm 0.012}$ | $0.031 \pm 0.013$ | $0.024 \pm 0.010$ | $0.192 \pm 0.021$ |
| bank_churn | $0.020 \pm 0.005$ | $0.019 \pm 0.005$ | $0.019 \pm 0.006$ | $0.020 \pm 0.004$ | $0.018 \pm 0.006$ | $\mathbf{0.018 \pm 0.006}$ | $0.226 \pm 0.047$ |
| bank_marketing | $\mathbf{0.009 \pm 0.002}$ | $0.010 \pm 0.002$ | $0.010 \pm 0.002$ | $0.011 \pm 0.002$ | $0.010 \pm 0.002$ | $0.010 \pm 0.003$ | $0.292 \pm 0.041$ |
| churn | $0.031 \pm 0.006$ | $0.032 \pm 0.004$ | $0.032 \pm 0.005$ | $0.033 \pm 0.006$ | $\mathbf{0.027 \pm 0.002}$ | $0.035 \pm 0.008$ | $0.282 \pm 0.026$ |
| credit_card | $\mathbf{0.013 \pm 0.003}$ | $0.014 \pm 0.004$ | $0.014 \pm 0.005$ | $0.016 \pm 0.003$ | $0.014 \pm 0.004$ | $0.016 \pm 0.002$ | $0.196 \pm 0.009$ |
| fico | $0.026 \pm 0.008$ | $\mathbf{0.025 \pm 0.007}$ | $0.025 \pm 0.009$ | $0.025 \pm 0.010$ | $0.025 \pm 0.003$ | $0.027 \pm 0.007$ | $0.053 \pm 0.014$ |
| fitness_club | $0.049 \pm 0.015$ | $\mathbf{0.047 \pm 0.017}$ | $0.048 \pm 0.018$ | $0.059 \pm 0.008$ | $0.051 \pm 0.013$ | $0.058 \pm 0.017$ | $0.155 \pm 0.029$ |
| hr_analytics | $0.019 \pm 0.008$ | $0.017 \pm 0.004$ | $\mathbf{0.016 \pm 0.006}$ | $0.021 \pm 0.008$ | $0.019 \pm 0.004$ | $0.018 \pm 0.007$ | $0.169 \pm 0.021$ |
| iranian_churn | $\mathbf{0.020 \pm 0.006}$ | $0.023 \pm 0.003$ | $0.024 \pm 0.003$ | $0.025 \pm 0.002$ | $0.022 \pm 0.004$ | $0.025 \pm 0.008$ | $0.264 \pm 0.017$ |
| kin8nm | $0.035 \pm 0.008$ | $0.033 \pm 0.011$ | $0.036 \pm 0.014$ | $0.036 \pm 0.009$ | $0.032 \pm 0.006$ | $\mathbf{0.032 \pm 0.005}$ | $0.219 \pm 0.009$ |
| shoppers | $\mathbf{0.011 \pm 0.003}$ | $0.014 \pm 0.002$ | $0.014 \pm 0.002$ | $0.014 \pm 0.002$ | $0.015 \pm 0.002$ | $0.020 \pm 0.003$ | $0.216 \pm 0.039$ |

Table 10: ECE (↓): Average and standard deviation were computed over CV folds. Best result per dataset is indicated in bold.

| Dataset | $\text{DAM}_{\text{moments}}$ | $\text{DAM}_{\text{ODE}}$ | DAM | NAMLSS | GAMLSS | gamboostLSS | CARD |
|---|---|---|---|---|---|---|---|
| adult | $0.310 \pm 0.004$ | $0.310 \pm 0.004$ | $0.310 \pm 0.004$ | $0.311 \pm 0.004$ | $\mathbf{0.307 \pm 0.004}$ | $0.309 \pm 0.004$ | $0.466 \pm 0.009$ |
| airline | $\mathbf{0.225 \pm 0.005}$ | $0.225 \pm 0.005$ | $0.225 \pm 0.005$ | $0.241 \pm 0.004$ | $0.237 \pm 0.004$ | $0.239 \pm 0.004$ | $0.362 \pm 0.006$ |
| bad_customer | $\mathbf{0.314 \pm 0.010}$ | $0.319 \pm 0.011$ | $0.318 \pm 0.011$ | $0.328 \pm 0.020$ | $0.317 \pm 0.015$ | $0.316 \pm 0.010$ | $0.444 \pm 0.015$ |
| bank_churn | $\mathbf{0.352 \pm 0.008}$ | $0.352 \pm 0.006$ | $0.352 \pm 0.006$ | $0.353 \pm 0.007$ | $0.399 \pm 0.007$ | $0.400 \pm 0.007$ | $0.517 \pm 0.054$ |
| bank_marketing | $\mathbf{0.306 \pm 0.003}$ | $0.306 \pm 0.002$ | $0.307 \pm 0.002$ | $0.307 \pm 0.003$ | $0.306 \pm 0.004$ | $0.308 \pm 0.003$ | $0.558 \pm 0.060$ |
| churn | $0.272 \pm 0.011$ | $0.272 \pm 0.011$ | $\mathbf{0.271 \pm 0.011}$ | $0.276 \pm 0.013$ | $0.273 \pm 0.012$ | $0.273 \pm 0.010$ | $0.454 \pm 0.038$ |
| credit_card | $\mathbf{0.429 \pm 0.005}$ | $0.430 \pm 0.005$ | $0.430 \pm 0.005$ | $0.431 \pm 0.006$ | $0.433 \pm 0.006$ | $0.433 \pm 0.006$ | $0.553 \pm 0.009$ |
| fico | $0.542 \pm 0.013$ | $0.542 \pm 0.013$ | $0.541 \pm 0.013$ | $0.544 \pm 0.013$ | $\mathbf{0.534 \pm 0.010}$ | $0.535 \pm 0.012$ | $0.630 \pm 0.033$ |
| fitness_club | $0.466 \pm 0.031$ | $0.468 \pm 0.030$ | $0.467 \pm 0.031$ | $0.465 \pm 0.031$ | $0.465 \pm 0.031$ | $\mathbf{0.464 \pm 0.028}$ | $0.571 \pm 0.019$ |
| hr_analytics | $0.361 \pm 0.004$ | $0.360 \pm 0.004$ | $\mathbf{0.360 \pm 0.004}$ | $0.362 \pm 0.005$ | $0.375 \pm 0.014$ | $0.360 \pm 0.004$ | $0.482 \pm 0.014$ |
| iranian_churn | $0.165 \pm 0.041$ | $0.174 \pm 0.036$ | $0.173 \pm 0.036$ | $0.176 \pm 0.037$ | $\mathbf{0.140 \pm 0.025}$ | $0.172 \pm 0.014$ | $0.393 \pm 0.020$ |
| kin8nm | $0.510 \pm 0.016$ | $0.502 \pm 0.016$ | $0.502 \pm 0.016$ | $0.509 \pm 0.015$ | $0.508 \pm 0.018$ | $0.507 \pm 0.016$ | $\mathbf{0.416 \pm 0.013}$ |
| shoppers | $\mathbf{0.241 \pm 0.007}$ | $0.245 \pm 0.007$ | $0.245 \pm 0.007$ | $0.249 \pm 0.007$ | $0.251 \pm 0.009$ | $0.254 \pm 0.006$ | $0.458 \pm 0.028$ |

Table 11: Classification NLL (↓): Average and standard deviation were computed over CV folds. For DAM, $\text{DAM}_{\text{ODE}}$ and CARD, we use a Monte Carlo estimate that first constructs the average probability per observation and uses these averages to compute the expected NLL. Best result per dataset is indicated in bold.

| Dataset | $\text{DAM}_{\text{moments}}$ | NAMLSS | GAMLSS | gamboostLSS |
|---|---|---|---|---|
| beijing | $0.978 \pm 0.006$ | $0.969 \pm 0.005$ | $0.977 \pm 0.004$ | $\mathbf{0.964 \pm 0.005}$ |
| concrete | $3.236 \pm 0.064$ | $3.224 \pm 0.028$ | $3.638 \pm 0.291$ | $\mathbf{3.180 \pm 0.070}$ |
| diamonds | $\mathbf{-0.730 \pm 0.074}$ | $16386.421 \pm 36642.183$ | – | $-0.047 \pm 1.032$ |
| food | $\mathbf{3.482 \pm 0.008}$ | $3.483 \pm 0.010$ | – | – |
| healthcare | $0.610 \pm 0.077$ | $0.541 \pm 0.167$ | $\mathbf{0.523 \pm 0.137}$ | $0.526 \pm 0.125$ |
| housing | $0.074 \pm 0.022$ | $\mathbf{0.067 \pm 0.010}$ | $14.555 \pm 32.084$ | $0.124 \pm 0.069$ |
| insurance | $0.543 \pm 0.080$ | $\mathbf{0.483 \pm 0.059}$ | $0.502 \pm 0.064$ | $0.487 \pm 0.056$ |
| miami | $-0.427 \pm 0.017$ | $\mathbf{-0.475 \pm 0.018}$ | – | $-0.313 \pm 0.044$ |
| protein | $2.987 \pm 0.005$ | $\mathbf{2.980 \pm 0.010}$ | – | $2.991 \pm 0.020$ |

Table 12: Regression NLL (↓): Average and standard deviation were computed over CV folds. Results are not available for DAM and CARD as their regression NLL cannot be directly evaluated. Best result per dataset is indicated in bold.

| Dataset | $\text{DAM}_{\text{moments}}$ | $\text{DAM}_{\text{ODE}}$ | DAM | NAMLSS | GAMLSS | gamboostLSS | CARD |
|---|---|---|---|---|---|---|---|
| beijing | $11.476 \pm 2.546$ | $218.245 \pm 3.536$ | $304.117 \pm 3.585$ | $\mathbf{10.843 \pm 2.763}$ | $304.005 \pm 19.584$ | $648.810 \pm 10.521$ | $342.291 \pm 31.654$ |
| concrete | $4.323 \pm 2.192$ | $196.369 \pm 1.024$ | $277.559 \pm 1.251$ | $\mathbf{2.599 \pm 1.173}$ | $1008.756 \pm 545.685$ | $302.203 \pm 0.900$ | $117.441 \pm 1.348$ |
| diamonds | $\mathbf{21.844 \pm 11.054}$ | $260.530 \pm 29.029$ | $353.427 \pm 29.403$ | $32.476 \pm 17.120$ | – | $684.192 \pm 8.161$ | $264.193 \pm 36.864$ |
| food | $6.397 \pm 0.588$ | $215.174 \pm 1.229$ | $306.000 \pm 1.402$ | $\mathbf{5.378 \pm 0.822}$ | – | – | $242.962 \pm 1.889$ |
| healthcare | $2.374 \pm 1.089$ | $204.421 \pm 0.784$ | $288.783 \pm 0.591$ | $\mathbf{2.311 \pm 0.802}$ | $2.749 \pm 0.916$ | $286.984 \pm 0.767$ | $117.694 \pm 0.827$ |
| housing | $16.190 \pm 3.851$ | $227.998 \pm 3.557$ | $305.515 \pm 5.626$ | $\mathbf{9.735 \pm 3.715}$ | $796.117 \pm 373.599$ | $472.785 \pm 3.534$ | $281.223 \pm 21.400$ |
| insurance | $2.726 \pm 2.296$ | $203.346 \pm 1.456$ | $288.551 \pm 1.638$ | $\mathbf{1.913 \pm 1.081}$ | $2.139 \pm 0.685$ | $288.199 \pm 1.055$ | $118.524 \pm 0.285$ |
| miami | $9.766 \pm 2.230$ | $215.381 \pm 1.955$ | $296.788 \pm 3.633$ | $\mathbf{9.136 \pm 1.132}$ | – | $558.837 \pm 4.293$ | $239.236 \pm 5.068$ |
| protein | $19.047 \pm 6.086$ | $221.317 \pm 2.478$ | $302.332 \pm 5.421$ | $\mathbf{12.595 \pm 2.979}$ | – | $786.046 \pm 4.158$ | $400.111 \pm 19.786$ |

Table 13: Training times in seconds (↓) on the regression datasets. Average and standard deviation computed over CV folds. Best result per dataset is indicated in bold.

| Dataset | $\text{DAM}_{\text{moments}}$ | $\text{DAM}_{\text{ODE}}$ | DAM | NAMLSS | GAMLSS | gamboostLSS | CARD |
|---|---|---|---|---|---|---|---|
| adult | $7.790 \pm 1.332$ | $224.636 \pm 1.723$ | $313.025 \pm 3.149$ | $\mathbf{4.766 \pm 0.523}$ | $117.326 \pm 23.686$ | $138.737 \pm 1.577$ | $420.059 \pm 3.527$ |
| airline | $22.396 \pm 2.799$ | $250.041 \pm 4.177$ | $342.823 \pm 3.424$ | $\mathbf{13.222 \pm 1.920}$ | $1143.064 \pm 355.137$ | $475.668 \pm 6.220$ | $545.043 \pm 8.202$ |
| bad_customer | $1.570 \pm 0.553$ | $218.746 \pm 1.203$ | $309.134 \pm 1.082$ | $\mathbf{1.286 \pm 0.642}$ | $4.266 \pm 1.023$ | $11.492 \pm 0.103$ | $227.305 \pm 0.352$ |
| bank_churn | $3.054 \pm 1.183$ | $213.030 \pm 1.288$ | $303.402 \pm 0.654$ | $\mathbf{2.413 \pm 1.022}$ | $25.476 \pm 7.075$ | $28.928 \pm 0.840$ | $401.459 \pm 3.349$ |
| bank_marketing | $7.561 \pm 0.882$ | $222.885 \pm 1.835$ | $325.316 \pm 28.268$ | $\mathbf{5.146 \pm 1.184}$ | $162.405 \pm 17.095$ | $125.369 \pm 1.422$ | $436.288 \pm 1.516$ |
| churn | $2.013 \pm 0.700$ | $224.699 \pm 0.647$ | $334.636 \pm 28.295$ | $\mathbf{1.480 \pm 0.534}$ | $62.421 \pm 4.707$ | $32.183 \pm 0.589$ | $328.216 \pm 1.948$ |
| credit_card | $5.261 \pm 1.231$ | $227.637 \pm 1.028$ | $319.355 \pm 6.112$ | $\mathbf{3.568 \pm 1.274}$ | $1305.763 \pm 1574.745$ | $143.946 \pm 1.843$ | $413.490 \pm 2.798$ |
| fico | $2.032 \pm 0.841$ | $229.857 \pm 0.542$ | $322.508 \pm 0.861$ | $\mathbf{1.562 \pm 0.757}$ | $75.671 \pm 21.581$ | $63.157 \pm 0.701$ | $398.244 \pm 1.753$ |
| fitness_club | $1.790 \pm 1.042$ | $212.497 \pm 0.837$ | $302.418 \pm 1.065$ | $\mathbf{1.513 \pm 0.743}$ | $1.517 \pm 0.369$ | $6.181 \pm 0.205$ | $219.164 \pm 0.191$ |
| hr_analytics | $2.370 \pm 0.556$ | $214.494 \pm 0.752$ | $303.316 \pm 1.191$ | $\mathbf{2.005 \pm 0.718}$ | $63.012 \pm 22.586$ | $29.196 \pm 0.757$ | $383.281 \pm 2.775$ |
| iranian_churn | $3.273 \pm 0.951$ | $225.535 \pm 1.754$ | $314.699 \pm 1.632$ | $\mathbf{2.691 \pm 0.608}$ | $57.653 \pm 25.389$ | $15.828 \pm 0.355$ | $268.754 \pm 1.786$ |
| kin8nm | $\mathbf{1.737 \pm 0.412}$ | $207.351 \pm 2.168$ | $290.572 \pm 1.345$ | $2.050 \pm 1.306$ | $42.839 \pm 15.666$ | $24.283 \pm 0.731$ | $399.162 \pm 3.709$ |
| shoppers | $4.216 \pm 0.840$ | $222.883 \pm 0.789$ | $312.839 \pm 0.774$ | $\mathbf{2.683 \pm 0.647}$ | $112.869 \pm 29.950$ | $55.313 \pm 0.969$ | $403.233 \pm 1.195$ |

Table 14: Training times in seconds (↓) on the classification datasets. Average and standard deviation computed over CV folds. Best result per dataset is indicated in bold.

### A.10.2 Learned Feature Contributions for Numerical Features

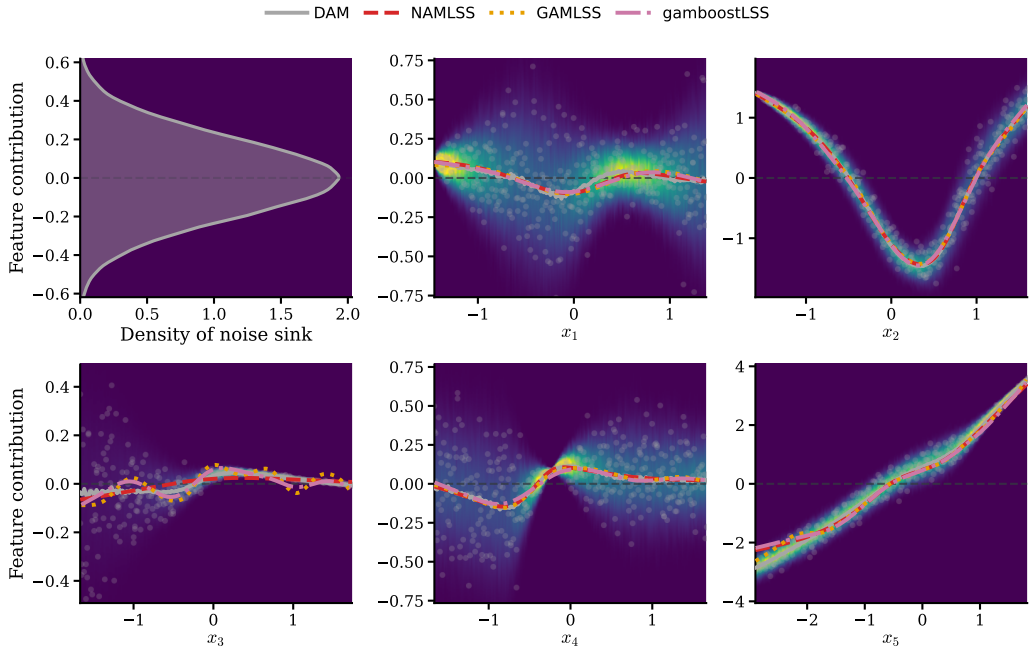



Figure 5: Examples of learned feature contributions for the beijing dataset. DAM generates full distributions and a noise sink $z_0$. We plot the deterministic means for the baselines and the implied mean for DAM.

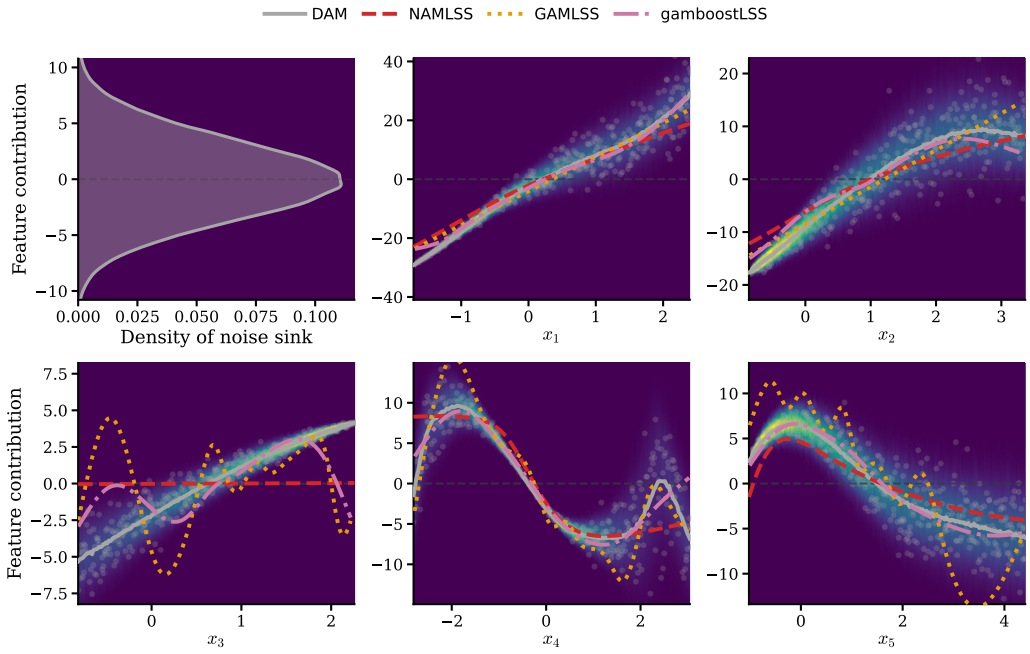



Figure 6: Examples of learned feature contributions for the concrete dataset. DAM generates full distributions and a noise sink $z_0$. We plot the deterministic means for the baselines and the implied mean for DAM.

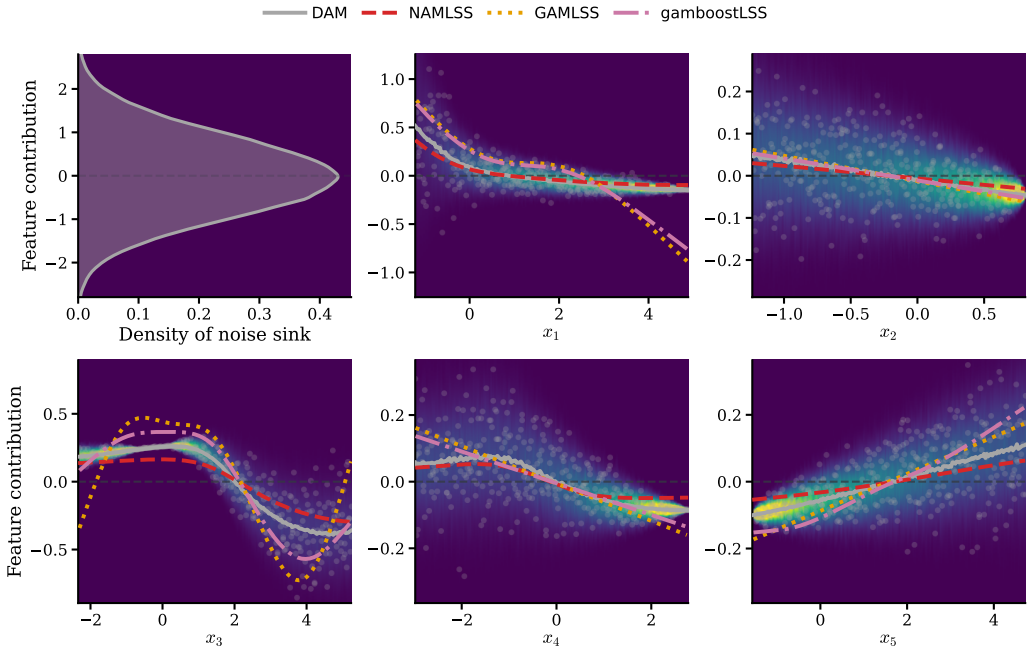



Figure 7: Examples of learned feature contributions for the credit_card dataset. DAM generates full distributions and a noise sink $z_0$. We plot the deterministic means for the baselines and the implied mean for DAM.

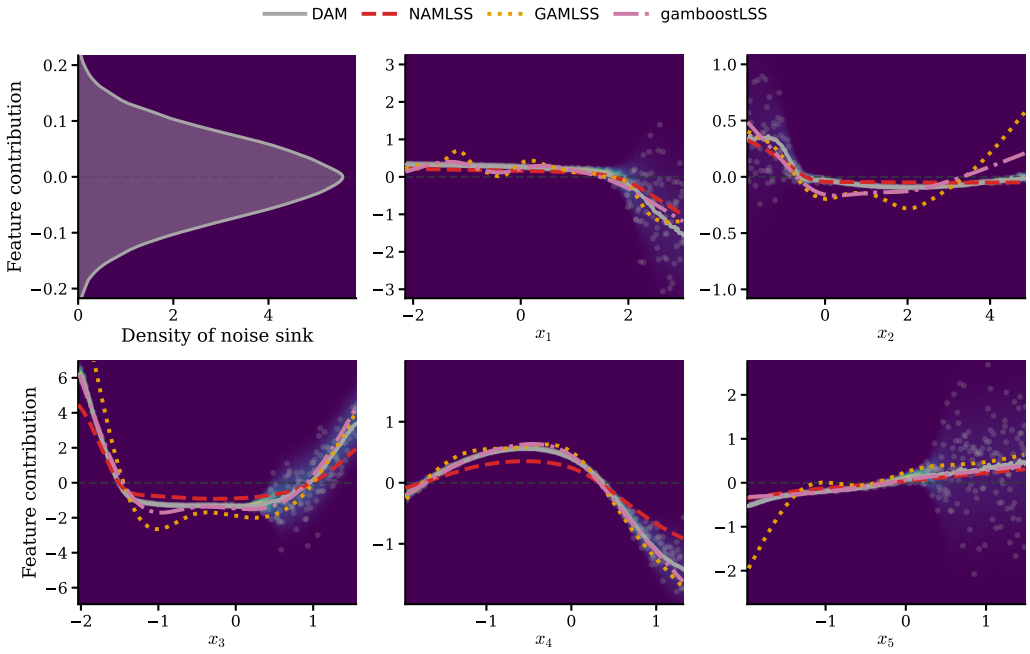



Figure 8: Examples of learned feature contributions for the airline dataset. DAM generates full distributions and a noise sink $z_0$. We plot the deterministic means for the baselines and the implied mean for DAM.

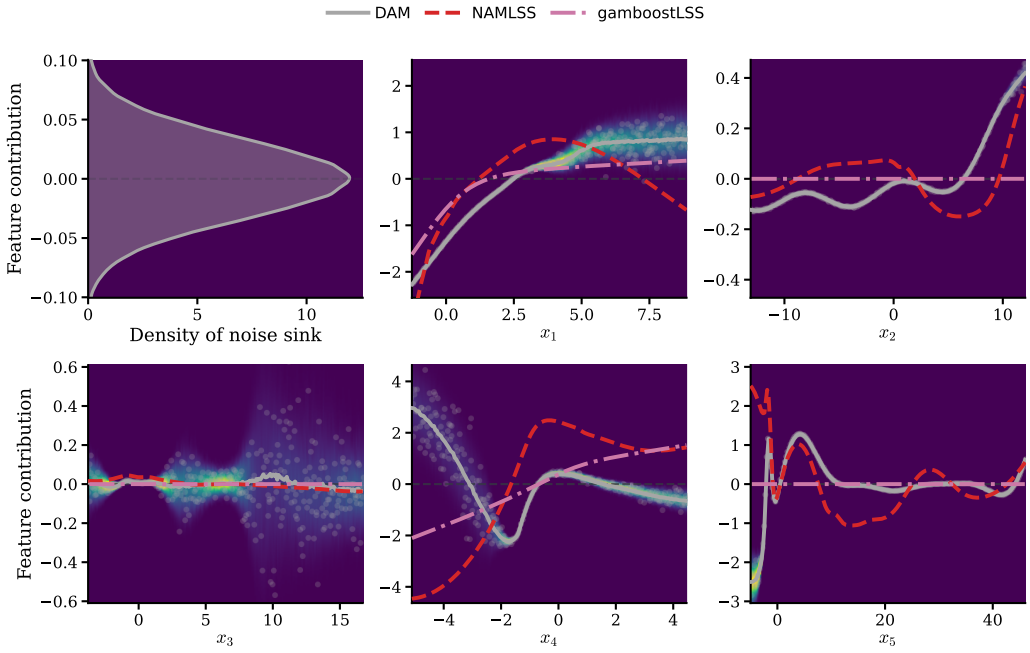



Figure 9: Examples of learned feature contributions for the diamonds dataset. DAM generates full distributions and a noise sink $z_0$. We plot the deterministic means for the baselines and the implied mean for DAM. GAMLSS does not converge for at least one fold and is therefore excluded.

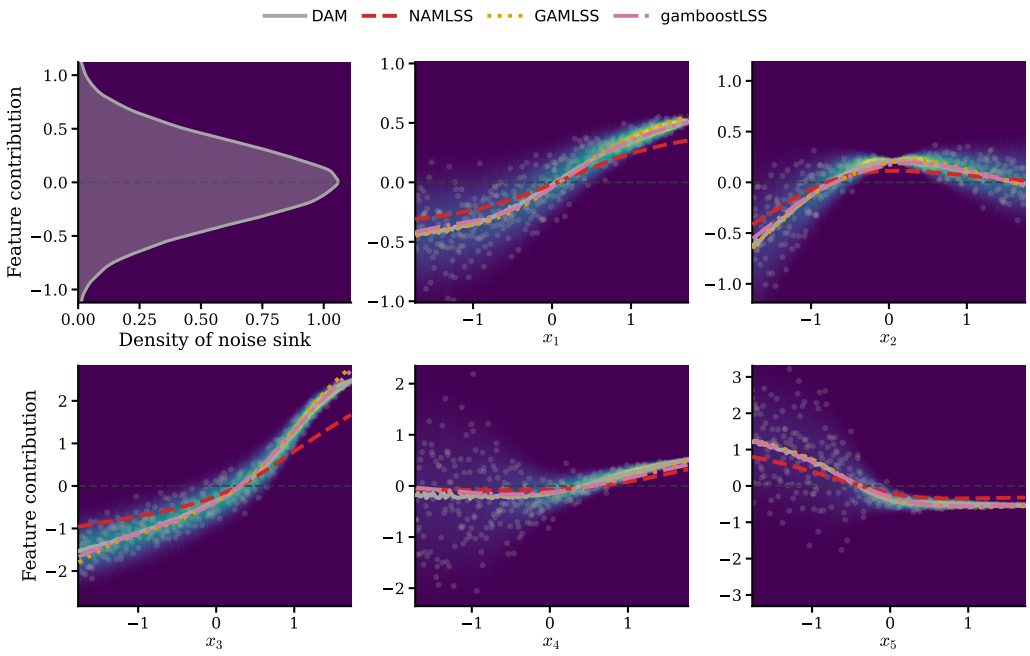



Figure 10: Examples of learned feature contributions for the kin8nm dataset. DAM generates full distributions and a noise sink $z_0$. We plot the deterministic means for the baselines and the implied mean for DAM.

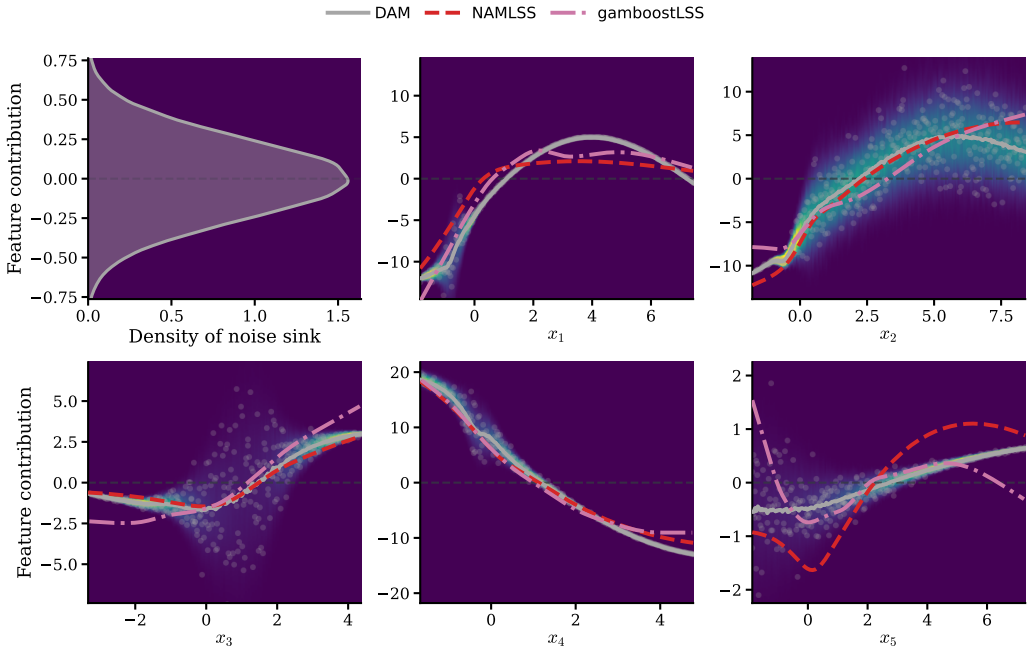



Figure 11: Examples of learned feature contributions for the protein dataset. DAM generates full distributions and a noise sink $z_0$. We plot the deterministic means for the baselines and the implied mean for DAM. GAMLSS does not converge for at least one fold and is therefore excluded.

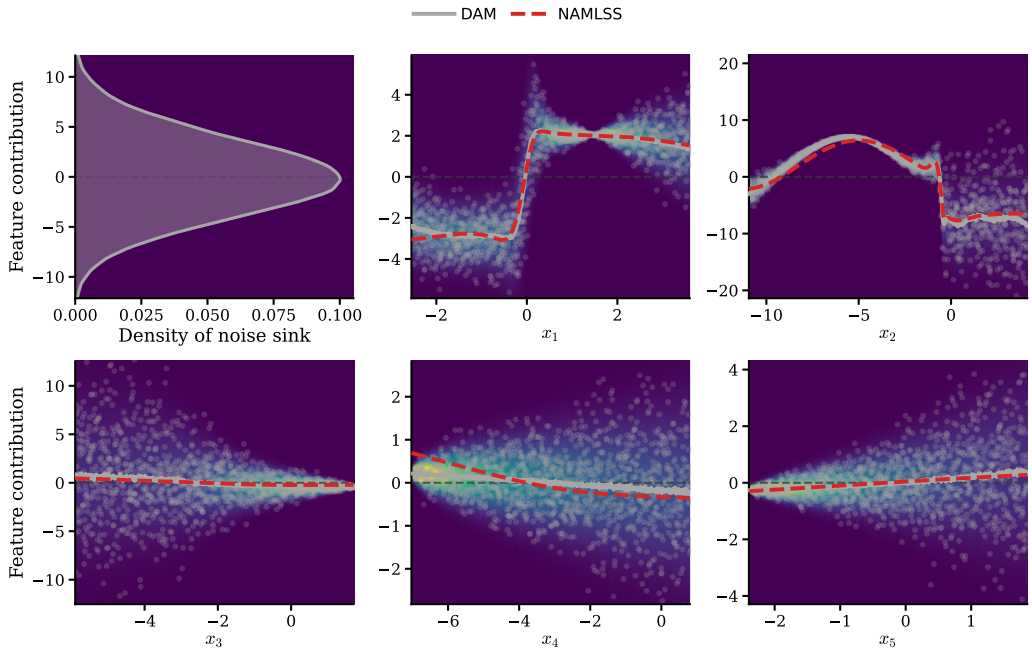



Figure 12: Examples of learned feature contributions for the food dataset. DAM generates full distributions and a noise sink $z_0$. We plot the deterministic means for the baselines and the implied mean for DAM. GAMLSS and gamboostLSS do not converge for at least one fold and are therefore excluded.

### A.10.3 Learned Feature Contributions for Categorical Features

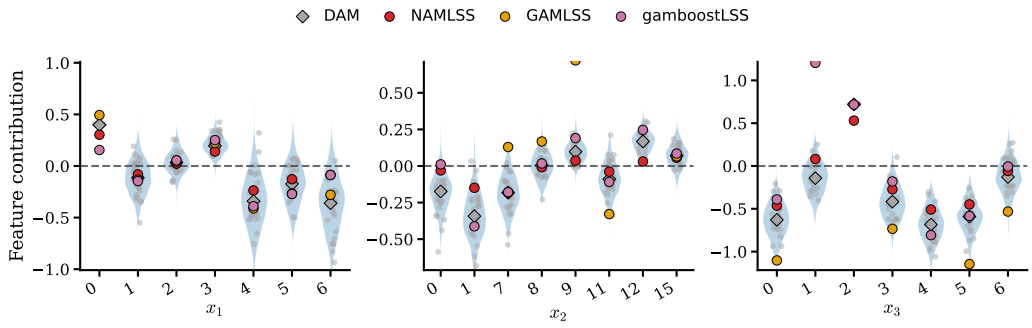



Figure 13: Examples of learned categorical feature contributions for the top eight categories per feature in the adult dataset. DAM captures the uncertainty in the contribution of each category. We plot the deterministic means for the baselines and the implied mean for DAM.

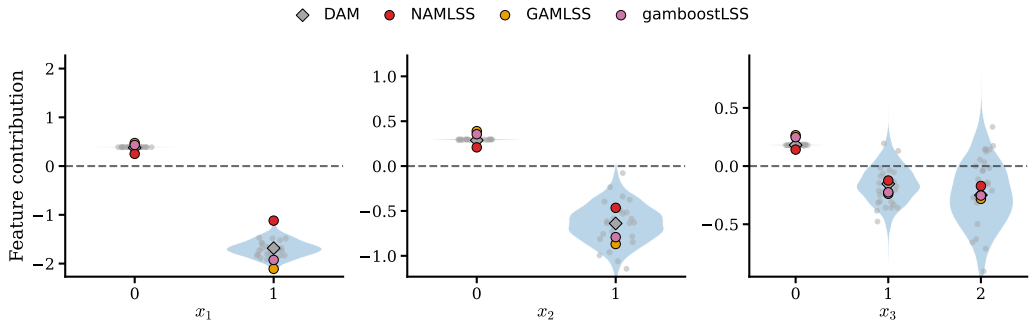



Figure 14: Examples of learned categorical feature contributions for the top eight categories per feature in the airline dataset. DAM captures the uncertainty in the contribution of each category. We plot the deterministic means for the baselines and the implied mean for DAM.

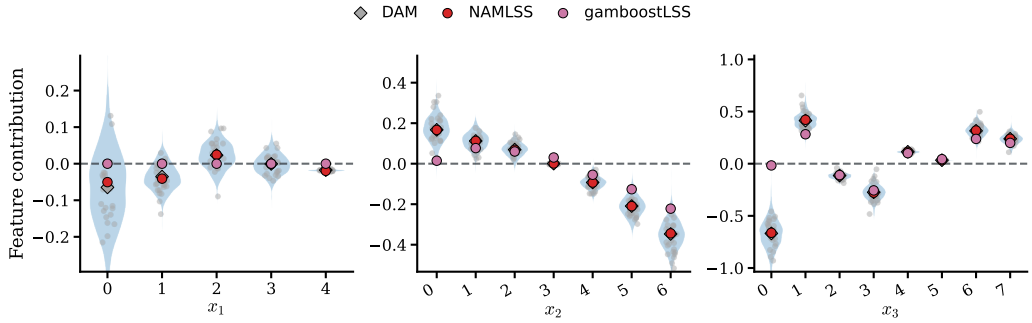



Figure 15: Examples of learned categorical feature contributions for the top eight categories per feature in the diamonds dataset. DAM captures the uncertainty in the contribution of each category. We plot the deterministic means for the baselines and the implied mean for DAM. GAMLSS does not converge for at least one fold and is therefore excluded.

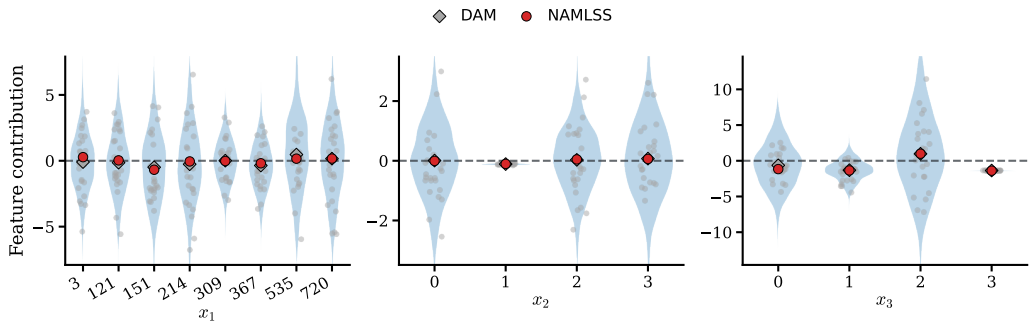



Figure 16: Examples of learned categorical feature contributions for the top eight categories per feature in the food dataset. DAM captures the uncertainty in the contribution of each category. We plot the deterministic means for the baselines and the implied mean for DAM. GAMLSS and gamboostLSS do not converge for at least one fold and are therefore excluded.

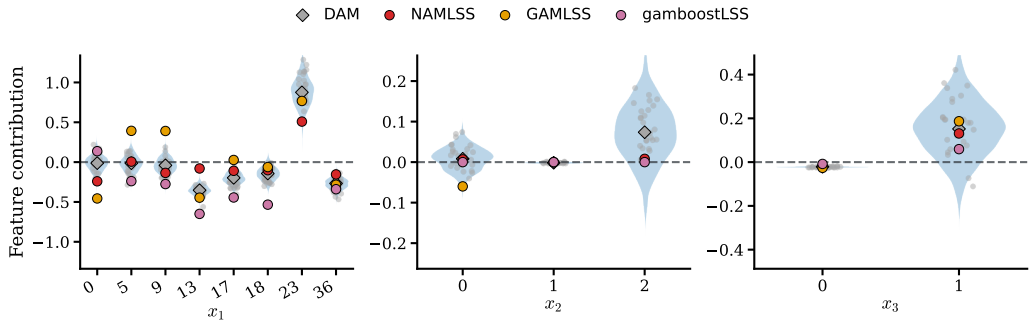



Figure 17: Examples of learned categorical feature contributions for the top eight categories per feature in the hr_analytics dataset. DAM captures the uncertainty in the contribution of each category. We plot the deterministic means for the baselines and the implied mean for DAM.

### A.10.4 Ablations

| Dataset | $\text{DAM}_{\text{moments}}$ | $\text{DAM}_{\text{ODE}}$ | DAM | (A): no standard. | (B): no routing | (C): collab. |
|---|---|---|---|---|---|---|
| beijing | $0.366 \pm 0.001$ | $0.366 \pm 0.001$ | $0.366 \pm 0.001$ | $0.366 \pm 0.001$ | $0.366 \pm 0.001$ | $\mathbf{0.365 \pm 0.001}$ |
| concrete | $3.520 \pm 0.209$ | $\mathbf{3.310 \pm 0.286}$ | $3.422 \pm 0.252$ | $3.385 \pm 0.271$ | $3.400 \pm 0.284$ | $3.425 \pm 0.250$ |
| diamonds | $0.064 \pm 0.005$ | $\mathbf{0.060 \pm 0.004}$ | $0.061 \pm 0.005$ | $0.061 \pm 0.005$ | $0.063 \pm 0.005$ | $0.061 \pm 0.005$ |
| food | $4.452 \pm 0.040$ | $4.445 \pm 0.046$ | $4.445 \pm 0.045$ | $4.446 \pm 0.047$ | $4.447 \pm 0.047$ | $\mathbf{4.437 \pm 0.047}$ |
| healthcare | $0.225 \pm 0.010$ | $0.213 \pm 0.017$ | $0.214 \pm 0.017$ | $0.214 \pm 0.017$ | $0.217 \pm 0.018$ | $\mathbf{0.211 \pm 0.018}$ |
| housing | $0.148 \pm 0.002$ | $\mathbf{0.146 \pm 0.003}$ | $0.146 \pm 0.003$ | $0.146 \pm 0.003$ | $0.147 \pm 0.004$ | $0.146 \pm 0.003$ |
| insurance | $0.226 \pm 0.010$ | $0.214 \pm 0.011$ | $0.215 \pm 0.011$ | $0.214 \pm 0.012$ | $0.216 \pm 0.015$ | $\mathbf{0.212 \pm 0.010}$ |
| miami | $0.097 \pm 0.002$ | $\mathbf{0.095 \pm 0.001}$ | $0.096 \pm 0.002$ | $0.095 \pm 0.002$ | $0.095 \pm 0.004$ | $0.096 \pm 0.002$ |
| protein | $2.762 \pm 0.010$ | $2.752 \pm 0.022$ | $2.755 \pm 0.023$ | $2.757 \pm 0.022$ | $2.755 \pm 0.023$ | $\mathbf{2.749 \pm 0.020}$ |

Table 15: CRPS (↓): Average and standard deviation computed over CV folds for all ablation models. Best result per dataset is indicated in bold.

| Dataset | $\text{DAM}_{\text{moments}}$ | $\text{DAM}_{\text{ODE}}$ | DAM | (A): no standard. | (B): no routing | (C): collab. |
|---|---|---|---|---|---|---|
| beijing | $0.949 \pm 0.003$ | $0.950 \pm 0.003$ | $\mathbf{0.950 \pm 0.003}$ | $0.950 \pm 0.002$ | $0.950 \pm 0.003$ | $0.952 \pm 0.002$ |
| concrete | $0.946 \pm 0.029$ | $\mathbf{0.949 \pm 0.031}$ | $0.943 \pm 0.030$ | $0.944 \pm 0.029$ | $0.941 \pm 0.028$ | $0.942 \pm 0.026$ |
| diamonds | $0.952 \pm 0.007$ | $0.951 \pm 0.005$ | $\mathbf{0.950 \pm 0.004}$ | $0.951 \pm 0.004$ | $0.951 \pm 0.005$ | $0.951 \pm 0.004$ |
| food | $0.942 \pm 0.003$ | $0.943 \pm 0.002$ | $0.943 \pm 0.002$ | $0.943 \pm 0.001$ | $0.943 \pm 0.002$ | $\mathbf{0.946 \pm 0.003}$ |
| healthcare | $0.932 \pm 0.018$ | $0.943 \pm 0.016$ | $0.942 \pm 0.017$ | $0.941 \pm 0.019$ | $0.940 \pm 0.018$ | $\mathbf{0.944 \pm 0.021}$ |
| housing | $0.945 \pm 0.004$ | $0.945 \pm 0.005$ | $0.944 \pm 0.004$ | $0.945 \pm 0.005$ | $0.944 \pm 0.005$ | $\mathbf{0.946 \pm 0.005}$ |
| insurance | $0.938 \pm 0.019$ | $0.942 \pm 0.016$ | $0.939 \pm 0.018$ | $0.942 \pm 0.018$ | $0.938 \pm 0.013$ | $\mathbf{0.945 \pm 0.011}$ |
| miami | $0.948 \pm 0.005$ | $0.951 \pm 0.006$ | $0.949 \pm 0.004$ | $0.950 \pm 0.005$ | $0.949 \pm 0.005$ | $\mathbf{0.950 \pm 0.006}$ |
| protein | $\mathbf{0.956 \pm 0.001}$ | $0.958 \pm 0.002$ | $0.958 \pm 0.002$ | $0.959 \pm 0.001$ | $0.959 \pm 0.002$ | $0.961 \pm 0.000$ |

Table 16: PICP (closer to 0.95 is better): Average and standard deviation computed over CV folds for all ablation models. Best result per dataset is indicated in bold.

| Dataset | $\text{DAM}_{\text{moments}}$ | $\text{DAM}_{\text{ODE}}$ | DAM | (A): no standard. | (B): no routing | (C): collab. |
|---|---|---|---|---|---|---|
| beijing | $0.006 \pm 0.001$ | $0.007 \pm 0.001$ | $0.006 \pm 0.001$ | $0.006 \pm 0.001$ | $0.006 \pm 0.001$ | $\mathbf{0.004 \pm 0.001}$ |
| concrete | $0.019 \pm 0.006$ | $0.018 \pm 0.003$ | $\mathbf{0.018 \pm 0.004}$ | $0.018 \pm 0.003$ | $0.019 \pm 0.004$ | $0.019 \pm 0.002$ |
| diamonds | $0.007 \pm 0.001$ | $0.007 \pm 0.001$ | $0.006 \pm 0.002$ | $0.007 \pm 0.003$ | $0.006 \pm 0.002$ | $\mathbf{0.005 \pm 0.002}$ |
| food | $0.016 \pm 0.001$ | $0.015 \pm 0.001$ | $0.015 \pm 0.001$ | $0.015 \pm 0.001$ | $0.015 \pm 0.001$ | $\mathbf{0.011 \pm 0.001}$ |
| healthcare | $0.068 \pm 0.009$ | $0.064 \pm 0.008$ | $0.066 \pm 0.008$ | $0.065 \pm 0.007$ | $0.067 \pm 0.005$ | $\mathbf{0.060 \pm 0.006}$ |
| housing | $0.011 \pm 0.003$ | $0.011 \pm 0.002$ | $0.011 \pm 0.002$ | $0.011 \pm 0.002$ | $0.010 \pm 0.002$ | $\mathbf{0.009 \pm 0.002}$ |
| insurance | $0.071 \pm 0.008$ | $0.057 \pm 0.008$ | $0.059 \pm 0.009$ | $0.059 \pm 0.008$ | $0.067 \pm 0.007$ | $\mathbf{0.056 \pm 0.007}$ |
| miami | $0.008 \pm 0.002$ | $0.008 \pm 0.002$ | $0.008 \pm 0.002$ | $0.009 \pm 0.002$ | $0.008 \pm 0.001$ | $\mathbf{0.007 \pm 0.002}$ |
| protein | $0.026 \pm 0.001$ | $0.024 \pm 0.001$ | $0.024 \pm 0.002$ | $0.023 \pm 0.001$ | $0.024 \pm 0.001$ | $\mathbf{0.020 \pm 0.002}$ |

Table 17: QICE coverage (↓): Average and standard deviation computed over CV folds for all ablation models. Best result per dataset is indicated in bold.