Neural Additive Models for Location Scale and Shape: A Framework for Interpretable Neural Regression Beyond the Mean

Anton Thielmann*
Clausthal University of Technology
*Equal contribution

René-Marcel Kruse*
University of Göttingen
*Equal contribution

Thomas Kneib
University of Göttingen

Benjamin Säfken
Clausthal University of Technology

Abstract

Deep neural networks (DNNs) have proven to be highly effective in a variety of tasks, making them the go-to method for problems requiring high-level predictive power. Despite this success, the inner workings of DNNs are often not transparent, making them difficult to interpret or understand. This lack of interpretability has led to increased research on inherently interpretable neural networks in recent years. Models such as Neural Additive Models (NAMs) achieve visual interpretability through the combination of classical statistical methods with DNNs. However, these approaches only concentrate on mean response predictions, leaving out other properties of the response distribution of the underlying data. We propose Neural Additive Models for Location Scale and Shape (NAMLSS), a modelling framework that combines the predictive power of classical deep learning models with the inherent advantages of distributional regression while maintaining the interpretability of additive models. The code is available at the following link: `https://github.com/AnFreTh/NAMpy`

1 Introduction

Deep learning models have shown impressive performances on a variety of predictive tasks. These models represent the forefront of technology for handling unstructured data tasks, including but not limited to image classification (Yu et al., 2022; Dosovitskiy et al., 2020), text classification (Huang et al., 2021; Lin et al., 2021), audio classification (Nagrani et al., 2021), time-series forecasting (Zhou et al., 2022; Zeng et al., 2022) and numerous other applications. However, the predictive performance comes not only at the price of computational demands. The black-box nature of deep neural networks poses hard challenges to interpretability. To achieve sample-level interpretability, existing methods resort to model-agnostic methods. Locally Interpretable Model Explanations (LIME) (Ribeiro et al., 2016) or Shapley values (Shapley, 1953) and their extensions (Sundararajan and Najmi, 2020) try to explain model predictions via local approximation and feature importance. Sensitivity-based approaches (Horel and Giesecke, 2020), exploiting significance statistics, can only be applied to single-layer feed-forward neural networks and can hence not be used to model difficult non-linear effects, requiring more complex model structures.

Subsequently, high-risk domains, such as medical applications often cannot exploit the advantages of complex neural networks due to their lack of innate interpretability. The creation of these innately interpretable models hence remains an important challenge. Achieving the interpretability from flexible statistical models as Generalized Linear Models (GLMs) (Nelder and Wedderburn, 1972) or Generalized Additive Models (GAMs) (Hastie, 2017), in deep neural networks, however, is inherently difficult. Recently, Agarwal et al. (2021) introduced Neural Additive Models (NAMs), a framework that models all features individually and thus creates visual interpretability of the single features. While this is an important step towards interpretable deep neural networks, any insightfulness of aspects beyond the mean is lost in the model structure. To

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

counter that, we propose the neural counterpart to Generalized Additive Models for Location, Scale and Shape (GAMLSS) (Rigby and Stasinopoulos, 2005), the **N**eural **A**dditive **M**odel for **L**ocation, **S**cale and **S**hape (NAMLSS). NAMLSS adopts and iterates on the model class of GAMLSS, in the same scope as NAMs (Agarwal et al., 2021) on GAMs.

The GAMLSS framework relaxes the exponential family assumption and replaces it with a general distribution family. The systematic part of the model is expanded to allow not only the mean (location) but all the parameters of the conditional distribution of the dependent variable to be modelled as additive nonparametric functions of the features, resulting in the following model notation:

$$\theta^{(k)} = {g^{(k)}}^{-1}\left(\beta^{(k)} + \sum_{j=1}^{J_k} f_j^{(k)}(x_j^{(k)})\right) = \eta_{\theta^{(k)}},$$

with the superscript $k = 1, \ldots, K$ denoting the k-th parameter and $j = 1, \ldots, J$ denoting the features.

The model assumes that the underlying response observations y_i for $i = 1, 2, \ldots, n$ are conditionally independent given the covariates. The assumed conditional density can depend on up to K different distributional parameters[1]. Each of these distribution parameters $\theta^{(k)}$ can be modelled using its additive predictor $\eta_{\theta^{(k)}}$ for $k = 1, \ldots, K$, allowing for complex relationships between the response and predictor variables, as well as the flexibility to choose different distributions for different parts of the response variable. An additional important component of the GAMLSS model is the link function $g^{(k)}(\cdot)$, which allows each parameter of the distribution vector to be conditional on different sets of covariates. In the case that the distribution under consideration features only one distribution parameter, the model simplifies to an ordinary GAM model. Therefore, GAMLSS is to be seen as a conceptual extension of the GAM idea and is suitable for the extension and generalisation of approaches such as NAMs which are themselves built upon the GAM idea. For an overview of the current state of regression models that focus on the full response distribution approaches, see Kneib et al. (2021).

While NAMs learn linear combinations of different input features to learn arbitrary complex functions and at the same time provide improved interpretability, these models, like their statistical counterparts GAMs, focus exclusively on modelling mean and dispersion. In contrast, the GAMLSS, and later the proposed NAMLSS, significantly broaden the scope by allowing all underlying parameters of the response distribution to potentially depend on the information in the covariates.

Contributions The contributions of the paper hence can be summarized as follows:

- We present a novel architecture for Neural Additive Models for Location, Scale and Shape.
- Compared to state-of-the-art GAM, GAMLSS and DNNs our NAMLSS demonstrates superior performance on benchmark datasets.
- We demonstrate that NAMLSS effectively captures the information underlying the data. Especially NAMLSS allows for prediction beyond point estimates, for instance prediction intervals.
- Lastly, we show that the NAMLSS approach allows to go beyond the mean prediction of the response and to model the entire response distribution.

2 Literature Review

The idea of generating feature-level interpretability in deep neural networks by translating GAMs into a neural framework was already introduced by Potts (1999) and expanded by de Waal and du Toit (2007). While the framework was remarkably parameter-sparse, it did not use backpropagation and hence did not achieve as good predictive results as GAMs, while remaining less interpretable. More recently, Agarwal et al. (2021) introduced NAMs, a more flexible approach than the Generalized Additive Neural Networks (GANNs) introduced by de Waal and du Toit (2007) that leverages the recent advances in the field of Deep Learning.

NAMs are a class of flexible and powerful machine learning models that combine the strengths of neural networks and GAMs. These models can be used to model complex, non-linear relationships between response and predictor variables, and can be applied to a wide range of tasks including regression, classification, and time series forecasting. The basic structure of a NAM consists of a sum of multiple components, each representing a different aspect of the relationship between the response and predictor variables. These components can be linear, non-linear, or a combination of both, and can be learned using a variety of optimization algorithms. One of the key advantages of NAMs is their inherent ability to learn the interactions between different predictor variables and the response without the need for manual feature engineering. This allows NAMs to capture complex relationships in the data that may not be easily apparent to the human eye.

[1] In practice most application focus on up to four $\boldsymbol{\theta}_i = \left(\theta_i^{(1)}, \theta_i^{(2)}, \theta_i^{(3)}, \theta_i^{(4)}\right)$.

The general form of a NAM can be written as:

$$\mathbb{E}(y) = h\left(\beta + \sum_{j=1}^{J} f_j(x_j)\right), \qquad (1)$$

where $h(\cdot)$ is the activation function used in the output layer, $x \in \mathbb{R}^j$ are the input features, β is the global intercept term, and $f_j : \mathbb{R} \to \mathbb{R}$ represents the Multi-Layer Perceptron (MLP) corresponding to the j-th feature. The similarity to GAMs is apparent, as the two frameworks mostly distinguish in the form the individual features are modelled. $h(\cdot)$ is comparable to the link function $g(\cdot)$.

Several extensions to the NAM framework have already been introduced. It is possible to take into account pairwise or higher order interaction effects (Yang et al., 2021; Enouen and Liu, 2022; Wang et al., 2021; Dubey et al., 2022). Chang et al. (2021) introduced NODE-GAM, a differentiable model based on oblivious neural decision trees developed for high-risk domains. All these models follow the additive framework from GAMs and learn the nonlinear additive features with separate networks, one for each feature or feature interaction, either leveraging MLPs (Potts, 1999; de Waal and du Toit, 2007; Agarwal et al., 2021; Yang et al., 2021; Radenovic et al., 2022), using decision trees (Chang et al., 2021) or using Splines (Rügamer et al., 2020; Seifert et al., 2022; Luber et al., 2023).

The applications of such models range from nowcasting (Jo and Kim, 2022), financial applications (Chen and Ye, 2022), to survival analysis (Peroni et al., 2022). While the linear combination of neural subnetworks provides a visual interpretation of the results, any interpretability beyond the feature-level representation of the model predictions is lost in their black-box subnetworks.

3 Beyond the Mean

Obviously, the mean (or the arithmetic mean as its empirical counterpart) provides only a rather incomplete description of a probability distribution (the empirical distribution of corresponding observations, in case of the arithmetic mean). While this fact is widely acknowledged when it comes to exploratory data analysis, it is also widely ignored in the context of prediction models where the focus is typically on predicting expected outcomes. This narrow focus reflects an interest in common or average observations, but is misleading when phenomena such as risk, extremes, or uncertainty are central to an analysis. With the GAMLSS-based framework considered in this paper, we are able to quantify effects of covariates not only on the mean, but

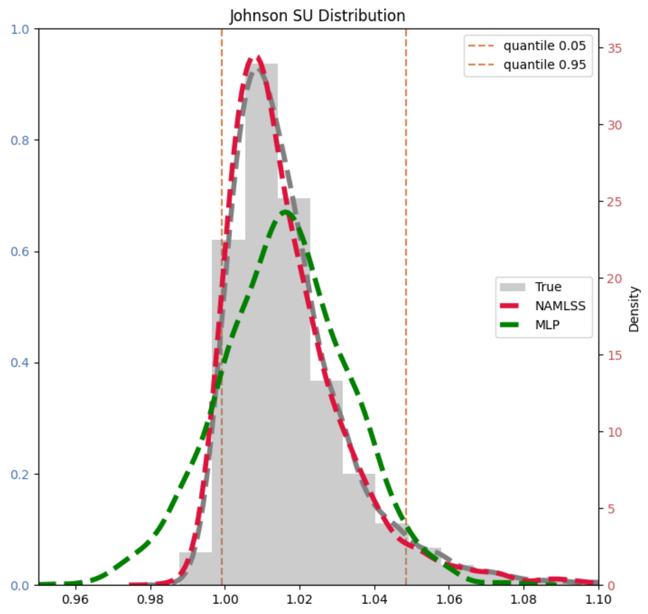

Figure 1: **Johnson's $\mathbf{S}_U$ Distribution:** Simulated Johnson's S_U distribution and the fit of a simple NAMLSS (see Figure 6) and a MLP. While the MLP achieves an impressive fit concerning the quadratic loss, it clearly cannot capture the underlying distribution adequately.

on any parameter of a potentially complex distribution assumed for the responses. As major advantage, the resulting models can determine changes in all aspects of the response distribution, such as variance, skewness or tail probabilities. This also contributes to properly disentangling aleatoric from epistemic uncertainty.

Changing the focus from regression models for the mean to regression for distributions also requires changes in the evaluation metric that is used to compare rivalling model specifications. More precisely, the evaluation metric should be proper (Gneiting and Raftery, 2007), i.e. enforce the analyst to report their true beliefs in terms of a predictive distribution. While the MSE that is commonly employed in mean-based modelling is proper for the mean, it is not for general distributions. We therefore will rely on the negative log-likelihood (also refereed to as the log-score) as a proper score for comparing distributional regression models (see Supplemental Material for details). Additionally to the negative log-likelihood, we use the Continuous Ranked Probability Score (Gneiting and Raftery, 2007) for model evaluation, given by:

$$CRPS(F, x) = -\int_{-\infty}^{\infty} (F(y) - \mathbf{1}_{y \geq x})^2 \, dy.$$

Note that the CRPS is also defined for models not specifically predicting all distributional parameters and thus allows a fair distributional comparison for all tested models:

$$CRPS(p,x) = \frac{1}{2}\left(\mathbb{E}_{p|X}\left[|X - \mathbf{X}| - \mathbb{E}_{p|X}\left[|X - x|\right]\right)\right. .$$

See Gneiting and Raftery (2007) for more details.

While predicting all parameters from a distribution may not always improve predictive power, understanding the underlying data distribution is crucial in high-risk domains and can provide valuable insights about feature effects. As an example, Figure 1 illustrates the fit of our approach on data following a Johnson's S_U distribution, including 3 features, compared to the fit of a MLP that minimizes the Mean Squared Error (MSE). The MLP has a better predictive performance with an MSE of 0.0002, however, NAMLSS is able to reflect the underlying data distribution much more accurately (as shown in Figure 1), even though it has an MSE of 0.0005.

The idea of focusing on more than the underlying mean prediction is thus certainly relevant and has been an important part, especially of the statistical literature in recent years. There has been a strong focus on the GAMLSS (Rigby and Stasinopoulos, 2005) framework, conditional transformation models (Hothorn et al., 2014), density regression (Wang et al., 1996) or quantile and expectile regression frameworks. However, these methods are inferior to machine and deep learning techniques in terms of pure predictive power; the disadvantage of not being able to deal with unstructured data forms such as images, text or audio files; or the inherent problems of statistical models in dealing with extremely large and complex data sets. One resulting development to deal with these drawbacks is frameworks that utilize statistical modelling methods and combine them with machine learning techniques such as boosting to create new types of distributional regression models such as boosted generalized additive model for location, scale and shape as presented by Hofner et al. (2014). More recently, there has been an increasing trend in incorporating distributional strategies within black-box boosting frameworks, often by modifying the model's objective function (e.g. (Duan et al., 2020; März and Kneib, 2022), through ensemble methods (Malinin et al., 2020) or by leveraging normalizing flows (Wielopolski and Zieba, 2022). However, the models leveraging boosting techniques, while successfully modelling all distributional parameters, lack the inherent interpretability from GAMLSS or even the visual interpretability from NAMs. Consequently, the benefits of analyzing individual feature contributions to distributional parameters, such as the variance in a normal distribution, may not be fully realized.

4 Methodology

While NAMs incorporate some feature-level interpretability and hence entail easy interpretability of the estimated regression effects, they are unable to capture skewness, heteroskedasticity or kurtosis in the underlying data distribution due to their focus on mean prediction. Therefore, the presented method is the neural counterpart to GAMLSS, offering the flexibility and predictive performance of neural networks while maintaining feature-level interpretability and which allows estimation of the underlying total response distribution.

Let $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$ be the training dataset of size n. Each input $x = (x_1, x_2, \ldots, x_J)$ contains J features. y denotes the target variable and can be arbitrarily distributed. NAMLSS are trained by minimizing the negative log-likelihood as the loss function, $-\log\left(\mathcal{L}(\theta|y)\right)$ by optimally approximating the distributional parameters, $\theta^{(k)}$. Each parameter, $\theta^{(k)}$, is defined as:

$$\theta^{(k)} = h^{(k)}\left(\beta^{(k)} + \sum_{j=1}^{J} f_j^{(k)}(x_j)\right), \qquad (2)$$

where $h^{(k)}(\cdot)$ denotes the output layer activation functions dependent on the underlying distributional parameter, $\beta^{(k)}$ denotes the parameter-specific intercept and $f_j^{(k)} : \mathbb{R} \to \mathbb{R}$ represents the feature network for parameter k for the j-th feature, subsequently called the *parameter-feature network*.

Just as in GAMLSS, $\theta^{(k)}$ can be derived from a subset of the J features, however, due to the inherent flexibility of the neural networks, defining each $\theta^{(k)}$ over all J is sufficient, as the individual feature importance for each parameter, $\theta^{(k)}$, is learned automatically. Each parameter-feature network, $f_j^{(k)}$, can be regularized employing regular dropout coefficients in conjunction with feature dropout coefficients, $\lambda_{1j}^{(k)}$ and $\lambda_{2j}^{(k)}$ respectively, as also implemented by Agarwal et al. (2021).

We propose two different network architectures that can both flexibly model all distributional parameters. The first model architecture, possible due to the flexibility of neural networks is depicted in Figure 2 and creates J subnetworks, with each subnetwork having a K-dimensional output layer. This architecture thus creates the same number of subnetworks as a common GAM and differs from the classical GAMLSS architecture as it comprises less feature networks/shape functions. Each distributional parameter, $\theta^{(k)}$, is subsequently obtained by summing over the k-th output of the J subnetworks. Every dimension in the output

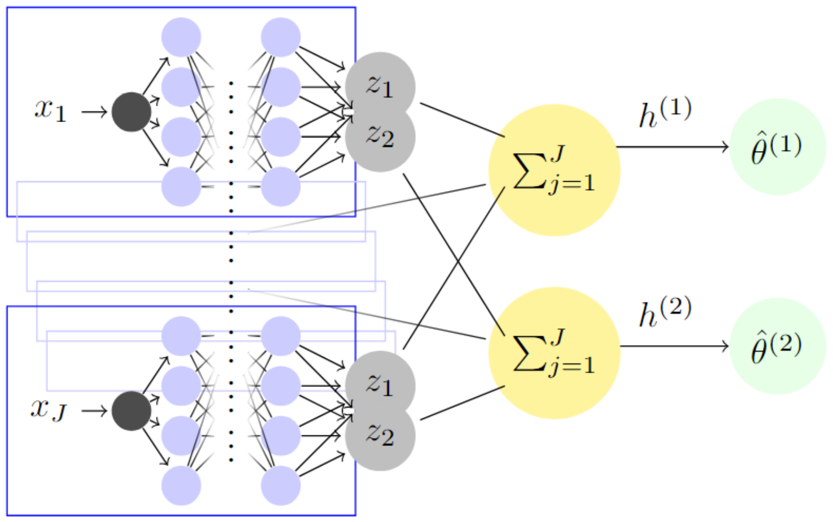

Figure 2: The network structure of a NAMLSS model. Each input variable is handled by a different neural network with k outputs for each distributional parameter. $h^{(k)}$ are different activation functions depending on the distributional parameter that is modelled. E.g. a quadratic transformation for modelling the variance in a normally distributed variable to ensure the non-negativity constraint. The presented structure demonstrates a NAMLSS modelling a distribution with two parameters, e.g. a normal distribution.

layer can be activated using different activation functions, according to parameter restrictions. This allows the capture of interaction effects between the distributional parameters in each of the subnetworks. Equation 2 would only slightly be adjusted, to account for the subnetwork f_j now mapping to $\mathbb{R}^k$, $f_j : \mathbb{R} \rightarrow \mathbb{R}^k$:

$$\theta^{(k)} = h^{(k)} \left(\beta^{(k)} + \sum_{j=1}^{J} f_j(x_j) \, [:, k] \right), \tag{3}$$

with $[:, k]$ denoting an index and representing the $k-th$ index of $f_j : \mathbb{R} \rightarrow \mathbb{R}^k$. Note, that the superscript $^{(k)}$ is missing from the subnetwork f_j, as only J subnetworks are trained. For instance, consider a simple example of a normal distribution: $\hat{\mu} = h^{(0)} + \beta^{(0)} \sum_{j=1}^{J} f_j(x_j)[:, 0]$ (with Pythonic indexing where 0 is the first index), and $\hat{\sigma} = g\left(h^{(1)} + \beta^{(1)} \sum_{j=1}^{J} f_j(x_j)[:, 1]\right)$, where $g(\cdot)$ describes the softplus activation function, $g(x) = \log(1 + e^x)$ since $\sigma \in \mathbb{R}^+$. For μ, $g(\cdot)$ is simply the linear activation and thus neglectable as $\mu \in \mathbb{R}$. These predictions are directly utilized in the loss function and thus serve as parameters for the negative log-likelihood, which, for the normal distribution, is given by:

$$-1 \cdot \log\left(\mathcal{L}(\hat{\mu}, \hat{\sigma}^2 | y)\right) = \frac{n}{2} \log(2\pi\hat{\sigma}^2) + \frac{1}{2\hat{\sigma}^2} \sum_{i=1}^{n} (y_i - \hat{\mu})^2.$$

The negative log-likelihood is then minimized using gradient descent, similar to standard neural networks. The additivity constraint serves two important purposes: First, as an additional regulator against overfitting, and second, to ensure interpretability. This constraint allows to visualize all feature effects on all distributional parameters. In terms of interpretability, accounting for uncertainty can be highly beneficial, particularly in domains such as healthcare and biology, where the impact of feature variance on outcomes is critical.

The second proposed architecture is depicted in Figure 6 in the Supplemental Material and creates J subnetworks for each of the K distributional parameters and thus much more resembles the classical GAMLSS architecture[2]. Each distributional subnetwork is comprised of the sum of the parameter-feature networks $f_j^{(k)}$. Hence we create $K \times J$ parameter-feature networks as denoted in equation 2. To account for distributional restrictions, each distributional subnetwork is again specified with possibly differing activation functions in the output layer.

Figure 3 demonstrates the accurate parameter fit of NAMLSS for a Johnson's S_U distribution. Each parameter is depicted in a different row. The black line shows the data generating functions. The simulated feature effects (columns) are accurately captured over all of the 150 runs for all 4 distributional parameters.

[2]Note, that for distributions where only one parameter is modelled, the two proposed NAMLSS structures are identical.

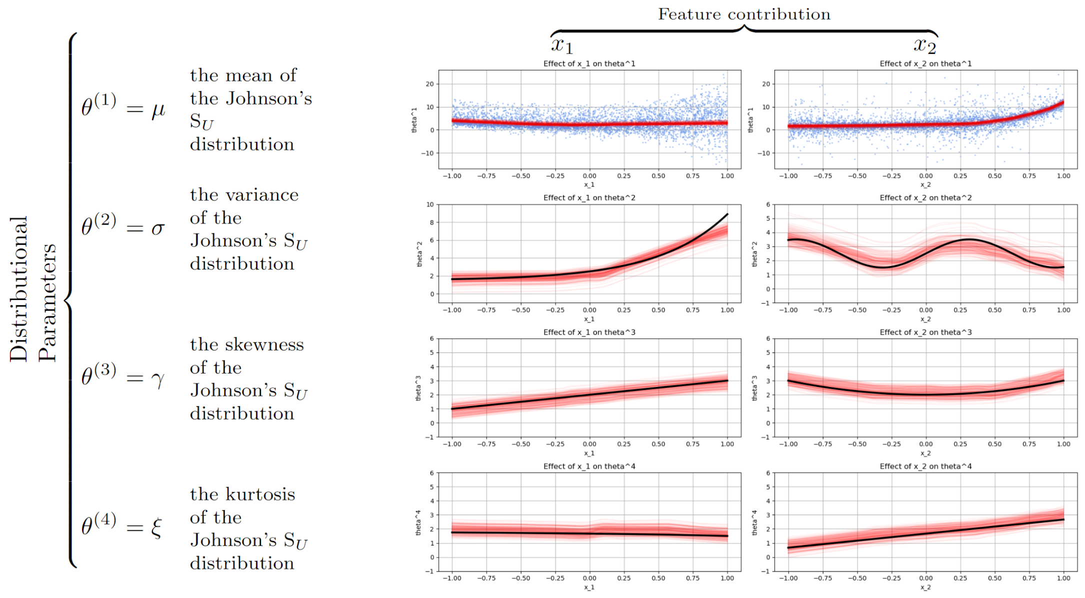

Figure 3: NAMLSS distributional parameter prediction for a Johnson's S$_U$ Distribution over 150 runs. NAMLSS accurately detects the feature effects of x_1 and x_2 on all distributional parameters, $\theta^{(1)}$ the mean, $\theta^{(2)}$ the standard deviation, $\theta^{(3)}$ the skewness and $\theta^{(4)}$ the tailweight.

5 Benchmarking

To demonstrate the competitiveness of the presented method, we perform several analyses. First, we compare NAMLSS with the most common statistical distributional regression approach GAMLSS (Stasinopoulos et al., 2000).

Synthetic data comparison study The synthetic data used for this task is generated from the same underlying processes. Five features are included in each application. The data-generating functions used to generate the true underlying distributional parameters can be found in the Supplemental Material 8.3.1. Each of the five input vectors x_j is sampled from a uniform distribution $\mathcal{U}(0, 1)$, with a total of $n = 3000$ observations per data set. The remaining parameters are generated based on the input vectors and the chosen distribution. We selected distributions that are widely used, popular in science, or relatively complex to reflect a diverse range of scenarios. The results can be found in Table 1.

Table 1: **Results for Synthetic data:** We compare NAMLSS with the baseline of additive distributional models, GAMLSS.

Distribution	GAMLSS	NAMLSS
	Neg Log-Likelihood $\downarrow$	
Binomial	**397**	274
Poisson	800	**802**
Normal	**600**	589
Inv. Gaussian	**385**	377
Weibull	**625**	621
Johnson's S$_U$	**370**	326
Gamma	**426**	410
Logistic	**731**	682

We find that the presented NAMLSS outperforms GAMLSS for all distributions except the Poisson distribution. This can be attributed to the fact that the Poisson distribution only involves a single distributional parameter.

Experiments with Real World Data We compare the performance of NAMLSS with several state-of-the-art models including neural as well as non-neural approaches and orientate on the benchmarks performed by Agarwal et al. (2021).

Additionally, we compare related methods of distribution-focused data analysis approaches that overcome the focus on relating the conditional mean of the response to features and instead target the complete conditional response distribution. Note, that we include the mean-centric models as a method to demonstrate the advantages of distributional approaches, particularly in terms of proper scoring metrics (negative log-likelihood, CRPS). Where no closed-form solution for the CRPS exists (Inverse Gamma distribution), we report the Kullback-Leibler Divergence. We have selected the following baselines for the comparisons:

The Multilayer Perceptron (MLP). Gradient Boosted Trees (XGBoost), based on decision tree-based gradient

Table 2: Average Rank table: The average rank over all models over all real world datasets. Both NAMLSS architectures outperform the mean reducing models by a large margin. We find the second NAMLSS architecture that captures parameter interaction effects to outperform the vanilla additive structure.

Model	Avg. Rank LL	Avg. Rank CRPS/KL
MLP	7.4	6.8
XGBoost	8.2	7.4
NAM	7.4	7.5
EBM	6.5	6.1
NodeGAM	7.8	7.5
DDNN	4.0	2.1
GAMLSS	4.5	6.0
gamboostLSS	4.2	5.3
NAMLSS[1]	1.9	3.1
NAMLSS[2]	**1.6**	**1.9**

[1] With $J \times K$ subnetworks
[2] With J subnetworks

boosting using the implementation provided by (Chen and Guestrin, 2016). Neural Additive Models (NAMs), represented as a linear combination of DNNs as described in equation (1) and presented by (Agarwal et al., 2021). Explainable Boosting Machines (EBMs), which are state-of-the-art Generalized Additive Models leveraging shallow boosted trees (Nori et al., 2019). Neural Generalized Additive Model (NodeGAM), leveraging Neural Oblivious Decision Trees (Chang et al., 2021). The Deep Distributional Neural Network (DDNN), a fully connected neural network trained to minimize the negative log-likelihood of the specified distribution. GAMLSS, employing standard GAMLSS models using the R implementation from (Rigby and Stasinopoulos, 2005). gamboost for Location Scale and Shape (gamboostLSS), fitting GAMLSS by employing boosting techniques as proposed by (Hofner et al., 2014).

These baseline models provide a diverse set of techniques for our comparative analysis.

We preprocess all used datasets exactly as done by Agarwal et al. (2021). We perform 5-fold cross-validation for all datasets and report the average performances over all folds as well as the standard deviations. For reproducability, we have only chosen publicly available datasets. The datasets, as well as the preprocessing and the seeds set for obtaining the folds, are described in detail in the Supplemental Material, 8.3.1. We fit all models without an intercept and explicitly do not model feature interaction effects.

For datasets following a Gaussian distribution we use the California Housing (CA Housing) dataset (Pace and Barry, 1997) from sklearn (Pedregosa et al., 2011), the Insurance dataset Lantz (2019), the Abalone dataset (Dua and Graff, 2017) and standard normalize the response variables. Thus, a normal distribution $\mathcal{N}\left(\mu, \sigma^2 \mathcal{I}\right)$ of the underlying response variables is assumed. This notably serves the objective of illustrating that, even in scenarios where mean-centric models designed to minimize the MSE should theoretically excel, they fall short in comparison to distributional approaches[3]. For a (binary) classification benchmark we use the FICO dataset (FICO, 2018), the Shrutime dataset and the Telco dataset. A logistic distribution, $\mathcal{LO}\left(\mu, s\right)$, of the underlying response variable was assumed (see equation (8.2.1) for the log-likelihood). Again, we use the true standard deviation of the underlying data for the models only resulting in a mean prediction. For the Melbourne and Munich datasets, also analyzed by Rügamer et al. (2020), we assume an Inverse Gamma distribution $\mathcal{IG}(\alpha, \beta)$ as the underlying data distribution (see equation (8.2.1) for the log-likelihood)[4].

The NAMLSS approach achieves the lowest negative log-likelihood and CRPS values for all of the datasets as shown in table 2. The architecture that implicitly captures the distributional parameter interactions slightly outperforms the architecture more closely related to GAMLSS.

One of the strengths of the NAMLSS, in contrast to the DNNs, is its interpretability at the feature level. Similar to NAMs, we can plot and visually analyze the results (see Figures 4 and 5). Additionally, we are able to accurately depict shifts in variance in the underlying data. It is, for example, clearly distinguishable, that with a larger median income, the house prices tend to vary much stronger than with a smaller median income (see Figure 4). A piece of information, that is lost in the models focusing solely on mean predictions Additionally, we are capable of accurately representing sharp price jumps around the location of San Francisco, depicted by the jumps in the graphs for longitude and latitude (see Figure 5) as compared to GAMLSS, NAMLSS are additionally capable of representing jagged shape functions.

[3]As the (negative) log-likelihood of a normal distribution (see equation (8.2.1)) is dependent on two parameters, but models as an MLP or XGBoost only predict a single parameter, we adjust the computation accordingly and use the true standard deviation calculated from the underlying data for XGBoost, EBM, NAM and MLP.

[4]See Supplemental Material for further details on activation functions.

Table 3: **Benchmark results:** For models not explicitly modelling a shape parameter, the shape is approximated with a constant as the true standard deviation of the dependent variable. Lower negative log-likelihoods (ℓ) are better. We report results on 6 commonly used datasets (see Supplemental Material for further results). The California Housing dataset for predicting house prices (Pace and Barry, 1997), an Insurance dataset for predicting billed medical expenses (Lantz, 2019), the Abalone dataset for predicting number of rings in trees (Dua and Graff, 2017), two AirBnb datasets and the FICO dataset for predicting *Risk Performance*.

	Negative Log-Likelihood ℓ (↓)					
	Normal			**Inv. Gamma**		**Logistic**
Model	CA Housing	Insurance	Abalone	Munich	Melbourne	FICO
MLP	4191 ±(42)	266.8 ± (11)	966.2 ±(27)	6827 ± (178)	22999 ± (232)	1813 ±(6)
XGBoost	4219 ±(40)	266.8 ± (9)	982.0 ±(33)	5618 ± (152)	20471 ± (242)	1976 ±(13)
NAM	4251 ±(43)	474.7 ± (73)	956.8 ±(22)	5892 ± (37)	25375 ± (844)	1809 ± (8)
EBM	4202 ±(42)	263.8 ± (10)	965.1 ±(22)	5474 ± (56)	20361 ± (207)	1944 ±(21)
NodeGAM	4206 ±(89)	279.1 ± (11)	958.3 ±(23)	5984 ± (135)	21896 ± (261)	1942 ±(21)
DDNN	2681 ±(1279)	178.2 ± (30)	897.2 ±(159)	5555 ± (34)	20790 ± (29)	1230 ± (48)
GAMLSS	3512 ±(67)	175.5 ± (28)	870.8 ±(16)	5419 ± (61)	26353 ± (45)	1321 ± (30)
gamboostLSS	3812 ±(52)	173.0 ± (28)	815.1 ± (29)	5421 ± (33)	26436 ± (48)	1191 ± (30)
NAMLSS[1]	2667 ± (91)	172.7 ± (23)	869.8 ±(118)	**5383** ± (24)	**19517** ± (68)	1201 ± (41)
NAMLSS[2]	**2329** ± (176)	**172.6** ± (20)	**802.3** ±(41)	5422 ± (22)	19675 ± (67)	**1160** ± (49)

[1] With $J \times K$ subnetworks. See Table 6 for an exemplary network structure.
[2] With J subnetworks and each subnetwork returning a parameter for the location and shape respectively. See Table 2 for an exemplary network structure.

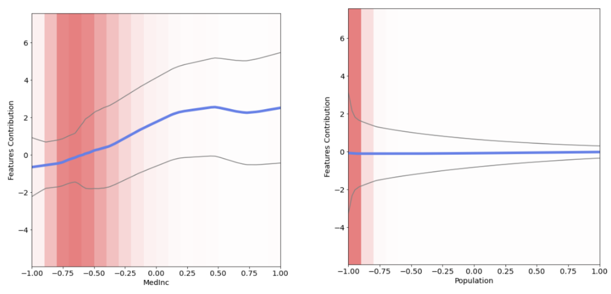

Figure 4: **California Housing:** Graphs for median income and population respectively learned by the NAMLSS model. We see an increase in housing prices with a larger median income. Additionally, we find a larger variance in housing prices in less densely populated areas.

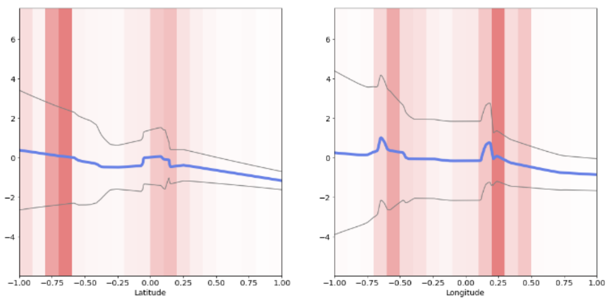

Figure 5: **California Housing:** Graphs for longitude and latitude respectively learned by the NAMLSS model. NAMLSS captures changes in mean as well as variance. Therefore, the plotted standard deviations change in dependence of the longitude and latitude. The house price jumps around the location of Los Angeles are depictable. Additionally, we find a decrease in variance for areas further away from the large cities.

6 Conclusion & Future Work

We have presented Neural Additive Models for Location, Scale and Shape and their theoretical foundation as the neural counterpart to GAMLSS. NAMLSS can model an arbitrary number of parameters of the underlying data distribution while preserving the predictive quality of NAMs. The visual intelligibility achieved by NAMs is also maintained by NAMLSS, with the added benefit of gaining further insights from knowledge of additional distribution characteristics. Hence, NAMLSS are a further step in the direction of fully interpretable neural networks and already offer interpretability that may make them suitable for high-risk domains.

The extensibility of NAMLSS offers many different further applied and theoretical research directions. One important point is the extension of the modelling of the distribution of the response variable. Many empirical works focus on modelling not just one, but several responses conditionally on covariates. One way to do this is to use copula methods, which are a valuable extension of our approach, hence including a copula-based approach for NAMLSS models would greatly improve the overall general usefulness. Another possible extension would be the adaptation to mixture density networks, as e.g. done by Seifert et al. (2022). Another possible focus is to switch our approach to a Bayesian-based training approach. Bayesian approaches are particularly well suited to deal with epistemic uncertainty and to incorporate it into the modelling. Another advantage is that Bayesian approaches are particularly suitable in cases where insufficiently small training datasets have to be dealt with and have been shown to have better prediction performance in these cases.

NAMLSS, akin to NAM, EBM and Node-GAM initially centers on tabular data. However, the model seamlessly extends its capability to encompass multimodal data by incorporating components such as a CNN for images as one of the feature networks. In this context, each distributional parameter is modelled by: $\theta^{(k)} = g\left(h^{(k)} + \beta^{(k)} \sum_{j=1}^{J} f_j(x_j)[:, k] + f_{img}(Z)[:, k]\right)$, where Z represents an image input and $f_{img}(Z)$ denotes e.g. a CNN. Further identifiability constraints such a orthogonalization (Rügamer, 2023) could be added to account for identifiability of the image effects.

7 Limitations

Although the presented method of NAMLSS takes advantage of the interpretation capabilities of the NAM framework and thus offers a better and easier interpretation of the results compared to pure deep learning approaches, it is still beholden to classical statistical models with their inherent interpretability and explainability. A critical point in the application of our proposed method, as well as comparable distributional statistical methods, is the choice of the correct distributional assumptions. The choice of the assumed distribution can strongly influence the results of the model. Our approach requires some basic mathematical-statistical knowledge from the user. Also, the understanding that the presented approach focuses on (log)-likelihood and thus deviates from the classical approach of simply minimising an error measure may require some users to rethink their understanding of the model results.

Acknowledgements

Funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) within project 450330162 is gratefully acknowledged.

References

Agarwal, R., Melnick, L., Frosst, N., Zhang, X., Lengerich, B., Caruana, R., and Hinton, G. E. (2021). Neural additive models: Interpretable machine learning with neural nets. *Advances in Neural Information Processing Systems*, 34.

Chang, C.-H., Caruana, R., and Goldenberg, A. (2021). Node-gam: Neural generalized additive model for interpretable deep learning. *arXiv preprint arXiv:2106.01613*.

Chen, D. and Ye, W. (2022). Generalized gloves of neural additive models: Pursuing transparent and accurate machine learning models in finance. *arXiv preprint arXiv:2209.10082*.

Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM.

Cole, T. J. and Green, P. J. (1992). Smoothing reference centile curves: the lms method and penalized likelihood. *Statistics in medicine*, 11(10):1305–1319.

de Waal, D. A. and du Toit, J. V. (2007). Generalized additive models from a neural network perspective. In *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*, pages 265–270. IEEE.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Duan, T., Anand, A., Ding, D. Y., Thai, K. K., Basu, S., Ng, A., and Schuler, A. (2020). Ngboost: Natural gradient boosting for probabilistic prediction. In *International conference on machine learning*, pages 2690–2700. PMLR.

Dubey, A., Radenovic, F., and Mahajan, D. (2022). Scalable interpretability via polynomials. *arXiv preprint arXiv:2205.14108*.

Dürr, O., Sick, B., and Murina, E. (2020). *Probabilistic Deep Learning: With Python, Keras and TensorFlow Probability*. Manning Publications.

Enouen, J. and Liu, Y. (2022). Sparse interaction additive networks via feature interaction detection and sparse selection. *arXiv preprint arXiv:2209.09326*.

FICO (2018). Fico explainable machine learning challenge.

Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102:359–378.

Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. (2021). Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34.

Hastie, T. J. (2017). Generalized additive models. In *Statistical models in S*, pages 249–307. Routledge.

Hofner, B., Mayr, A., and Schmid, M. (2014). gamboostlss: An r package for model building and variable selection in the gamlss framework. *arXiv preprint arXiv:1407.1774*.

Horel, E. and Giesecke, K. (2020). Significance tests for neural networks. *Journal of Machine Learning Research*, 21(227):1–29.

Hothorn, T., Kneib, T., and Bühlmann, P. (2014). Conditional transformation models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(1):3–27.

Huang, Y., Giledereli, B., Köksal, A., Özgür, A., and Ozkirimli, E. (2021). Balancing methods for multi-label text classification with long-tailed class distribution. *arXiv preprint arXiv:2109.04712*.

IBM (2019). Telco customer churn.

Jo, W. and Kim, D. (2022). Neural additive models for nowcasting. *arXiv preprint arXiv:2205.10020*.

Kaggle (2019). Churn modelling.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kneib, T., Silbersdorff, A., and Säfken, B. (2021). Rage against the mean–a review of distributional regression approaches. *Econometrics and Statistics*.

Lantz, B. (2019). *Machine learning with R: expert techniques for predictive modeling*. Packt publishing ltd.

Lin, Y., Meng, Y., Sun, X., Han, Q., Kuang, K., Li, J., and Wu, F. (2021). Bertgcn: Transductive text classification by combining gcn and bert. *arXiv preprint arXiv:2105.05727*.

Luber, M., Thielmann, A., and Säfken, B. (2023). Structural neural additive models: Enhanced interpretable machine learning. *arXiv preprint arXiv:2302.09275*.

Malinin, A., Prokhorenkova, L., and Ustimenko, A. (2020). Uncertainty in gradient boosting via ensembles. *arXiv preprint arXiv:2006.10562*.

März, A. and Kneib, T. (2022). Distributional gradient boosting machines. *arXiv preprint arXiv:2204.00778*.

Nagrani, A., Yang, S., Arnab, A., Jansen, A., Schmid, C., and Sun, C. (2021). Attention bottlenecks for multimodal fusion. *Advances in Neural Information Processing Systems*, 34:14200–14213.

Nash, W. J., Sellers, T. L., Talbot, S. R., Cawthorn, A. J., and Ford, W. B. (1994). The population biology of abalone (haliotis species) in tasmania. i. blacklip abalone (h. rubra) from the north coast and islands of bass strait. *Sea Fisheries Division, Technical Report*, 48:p411.

Nelder, J. A. and Wedderburn, R. W. (1972). Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384.

Nori, H., Jenkins, S., Koch, P., and Caruana, R. (2019). Interpretml: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*.

Pace, R. K. and Barry, R. (1997). Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Peroni, M., Kurban, M., Yang, S. Y., Kim, Y. S., Kang, H. Y., and Song, J. H. (2022). Extending the neural additive model for survival analysis with ehr data. *arXiv preprint arXiv:2211.07814*.

Potts, W. J. (1999). Generalized additive neural networks. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 194–200.

Radenovic, F., Dubey, A., and Mahajan, D. (2022). Neural basis models for interpretability. *arXiv preprint arXiv:2205.14120*.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Rigby, R. A. and Stasinopoulos, D. M. (2005). Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 54(3):507–554.

Rügamer, D. (2023). A new pho-rmula for improved performance of semi-structured networks. *arXiv preprint arXiv:2306.00522*.

Rügamer, D., Kolb, C., and Klein, N. (2020). Semi-structured deep distributional regression: Combining structured additive models and deep learning. *arXiv preprint arXiv:2002.05777*.

Seifert, Q. E., Thielmann, A., Bergherr, E., Säfken, B., Zierk, J., Rauh, M., and Hepp, T. (2022). Penalized regression splines in mixture density networks.

Shapley, L. (1953). Quota solutions op n-person games1. *Edited by Emil Artin and Marston Morse*, page 343.

Stasinopoulos, D., Rigby, R., and Fahrmeir, L. (2000). Modelling rental guide data using mean and dispersion additive models. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 49(4):479–493.

Sundararajan, M. and Najmi, A. (2020). The many shapley values for model explanation. In *International conference on machine learning*, pages 9269–9278. PMLR.

Wang, P., Puterman, M. L., Cockburn, I., and Le, N. (1996). Mixed poisson regression models with covariate dependent rates. *Biometrics*, pages 381–400.

Wang, T., Yang, J., Li, Y., and Wang, B. (2021). Partially interpretable estimators (pie): black-box-refined interpretable machine learning. *arXiv preprint arXiv:2105.02410*.

Wielopolski, P. and Zieba, M. (2022). Treeflow: Going beyond tree-based gaussian probabilistic regression. *arXiv preprint arXiv:2206.04140*.

Yang, Z., Zhang, A., and Sudjianto, A. (2021). Gaminet: An explainable neural network based on generalized additive models with structured interactions. *Pattern Recognition*, 120:108192.

Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M., and Wu, Y. (2022). Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. (2022). Are transformers effective for time series forecasting? *arXiv preprint arXiv:2205.13504*.

Zhou, T., Ma, Z., Wen, Q., Sun, L., Yao, T., Jin, R., et al. (2022). Film: Frequency improved legendre memory model for long-term time series forecasting. *arXiv preprint arXiv:2205.08897*.

8 Supplemental Material for NAMLSS

8.1 Network architecture

We propose two different network architectures that can both flexibly model all distributional parameters. One is depicted in Figure 6 and creates J subnetworks for each distributional parameter. Each distributional subnetwork is comprised of the sum of $f_j^{(k)}$. Hence we create $K \times J$ subnetworks. To account for distributional restrictions, each distributional subnetwork is specified with possibly differing activation functions in the output layer.

The second model architecture is depicted in Figure 2. Here we only create J subnetworks and hence have the same amount of subnetworks as a common NAM. Each subnetwork then has a k-dimensional output layer. Each distributional Parameter, $\theta^{(k)}$, is subsequently obtained by summing over the k-th output of the J subnetworks. Each dimension in the output layer can be activated using different activation functions, adjusting to parameter restrictions.

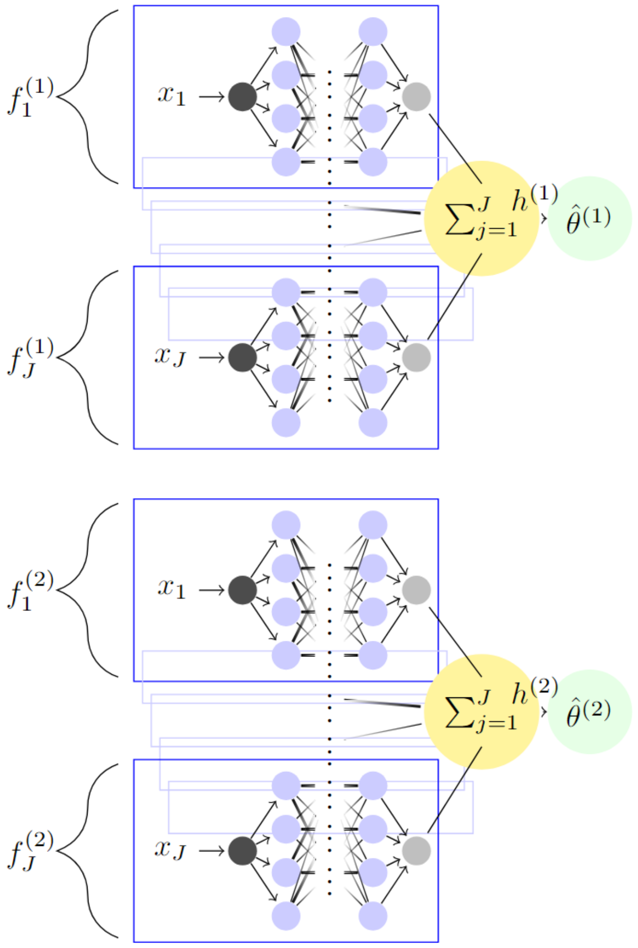

Figure 6: The network structure of a simple NAMLSS model. Each input variable as well as each distributional parameter is handled by a different neural network. h_k are different activation functions depending on the distributional parameter that is modelled. E.g. a quadratic transformation for modelling the variance in a normally distributed variable to ensure the non-negativity constraint.

8.2 Scoring

We use negative-log likelihoods as well as the CRPS for evaluation metrics. All used log-likelihoods are given in the following.

8.2.1 Log-Likelihoods

As the presented method minimizes negative log-likelihoods, we created a comprehensive list of all the log-likelihoods of the distributions used in the paper. When we reference the results of NAMLSS these are the log-likelihoods we used for fitting the models as well as evaluating them.

(Bernoulli) Logistic Distribution The log-likelihood function for a logistic distribution is given by:

$$\log\left(\mathcal{L}(\mu,\sigma|y)\right) = \sum_{i=1}^{n}\left[y_i \log\left(\frac{1}{1+e^{-\left(\frac{y_i-\mu}{\sigma}\right)}}\right) + (1-y_i)\log\left(1-\frac{1}{1+e^{-\left(\frac{y_i-\mu}{\sigma}\right)}}\right)\right],$$

with n is as the number of observations and the parameters location $\mu \in \mathbb{R}$, scale $\sigma \in \mathbb{R}^+$ and $x \in \mathbb{R}$.

Binomial Distribution The log-likelihood function for a binomial distribution is given by:

$$\log\left(\mathcal{L}(k|n,p)\right) = k\log(p) + (n-k)\log(1-p) + \log\left(\binom{n}{k}\right),$$

where n is the number of trials, the parameters success probability is given by $p \in [0,1]$ and the number of successes is denoted as $k \in \mathbb{N}_0$.

Inverse Gamma Distribution The log-likelihood function of the inverse gamma distribution is defined as:

$$\log\left(\mathcal{L}(\alpha,\beta|y)\right) = -n\,(\alpha+1)\,\overline{\log \boldsymbol{y}} - n\log\Gamma(\alpha) + n\alpha\log\beta - \sum_{i=1}^{n}\beta y_i^{-1}.$$

with $\alpha > 0$ and $\beta > 0$ and where the upper bar operand indicates the arithmetic mean

Normal Distribution The log-likelihood function for a normal distribution is given by:

$$log\left(\mathcal{L}(\mu,\sigma^2|y)\right) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\mu)^2,$$

where n is the underlying number of observations and parameters $y \in \mathbb{R}$, location $\mu \in \mathbb{R}$ and scale $\sigma \in \mathbb{R}^+$.

Inverse Gaussian Distribution The log-likelihood function of the inverse Gaussian distribution is given by:

$$\log\left(\mathcal{L}(\mu,\sigma|x)\right) = \frac{n}{2}\ln(\sigma) - \sum_{i=1}^{n}\frac{\sigma(x_i-\mu)^2}{2\mu^2 x_i},$$

with n is as the number of observations and the parameters location $\mu \in \mathbb{R}^+$, scale $\sigma \in \mathbb{R}^+$ and $x \in \mathbb{R}^+$.

Poisson Distribution The log-likelihood function for a Poisson distribution with parameter λ is given by:

$$\log\left(\mathcal{L}(\lambda|x)\right) = \sum_{i=1}^{n}[x_i\log(\lambda) - \lambda - \log(x_i!)]$$

where $x = (x_1, x_2, ..., x_n)$ is the sample, n is the number of observations and x_i are non-negative integers.

Johnson's S_U The log-likelihood function of the Johnson's S_U distribution is defined as:

$$\log\left(\mathcal{L}(\beta,\omega,\mu,\sigma|y)\right) = n\log\left[\frac{\beta}{\omega\sqrt{2\pi}}\right] - \frac{\beta^2}{2\omega^2}\sum_{i=1}^{n}\left[\frac{(y_i-\mu)^2}{\sigma^2} + \ln\left(1+\frac{(y_i-\mu)^2}{\omega^2\sigma^2}\right)\right],$$

with n is as the number of observations and the parameters location $\mu \in \mathbb{R}$, scale $\sigma \in \mathbb{R}^+$, shape $\omega \in \mathbb{R}^+$, skewness $\beta \in \mathbb{R}$ and $y \in \mathbb{R}$.

Weibull The log-likelihood function of the Weibull distribution is defined as:

$$\log\left(\mathcal{L}(\lambda,\beta,|y)\right) = n\ln\beta - n\beta\ln\lambda - \sum_{i=1}^{n}\left(\frac{y_i}{\lambda}\right)^{\beta} + (\beta-1)\sum_{i=1}^{n}\ln y_i,$$

with n is the number of observations and with the location $\lambda \in \mathbb{R}^+$, the shape $\beta \in \mathbb{R}^+$ and $y \in \mathbb{R}^+$.

8.2.2 Deviance Measures

We use several deviance measures, to evaluate the model

Mean Squared Error The mean squared error is defined as :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

Mean Gamma Deviance The mean gamma deviance used for the AirBnB dataset is defined as:

$$D = \frac{2}{n}\sum_{i=1}^{n}\log\left(\frac{\hat{y}_i}{y_i}\right).$$

Area Under the Curve We use the Riemannian formula for the AUC. Hence the area of rectangles is defined as:

$$AR = \sum_{i} = 1^{n-1} f(x_i)\Delta x,$$

and hence with larger n, the definite integral of f from a to b is defined as:

$$\int_a^b f(x)dx = \lim_{n\to\infty}\sum_{i=0}^{n-1} f(x_i)\Delta x.$$

8.3 Further Benchmark Results

All Benchmark results are given below. Additionally to the negative log-likelihood and the CRPS we include the presented deviance measures. As expected, mean centric models such as NAM, EBM perform well for the mean centric metrics. Additionally, similar to figure 3 we present the same simulation for a normal distribution: Furthermore, we present that NAMLSS accurately captures the mean effects as well as the variance effects for the same features as used in the original NAM paper (Agarwal et al., 2021). The single feature longitude and latitude effect for the California Housing dataset are accurately captured.

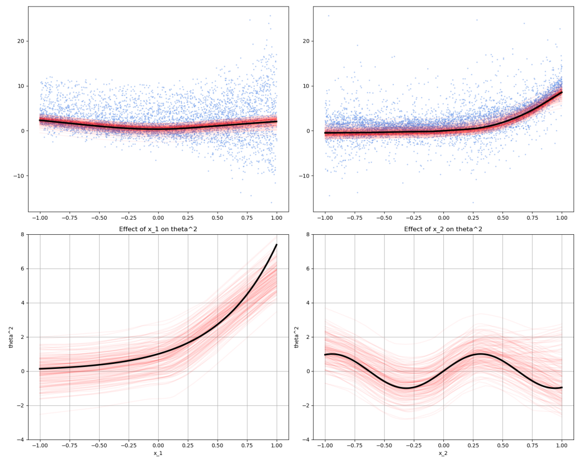

Figure 7: NAMLSS distributional parameter prediction for a Normal Distribution over 150 runs. NAMLSS accurately detects the feature effects of x1 and x2 on all distributional parameters.

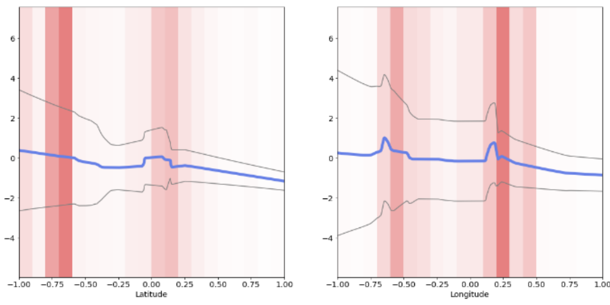

Figure 8: **California Housing:** Graphs for longitude and latitude respectively learned by the NAMLSS model. NAMLSS captures changes in mean as well as variance. Therefore, the plotted standard deviations change in dependence of the longitude and latitude. The house price jumps around the location of Los Angeles are depictable. Additionally, we find a decrease in variance for areas further away from the large cities.

All benchmark results are given below.

Table 4: Benchmark results for the FICO and the Shrutime dataset. We report continuous ranked probability score (CRPS) for the logistic distribution as there exists a close form solution as well as the AUC score.

Model	**Logistic**					
		FICO			Shru Time	
	LL ↓	AUC ↑	CRPS ↓	LL ↓	AUC ↑	CRPS ↓
MLP	1813 ± (6)	**0.79** ± (0.007)	0.386 ± (0.014)	1240 ± (22)	0.73 ± (0.011)	0.296 ± (0.010)
XGBoost	1976 ± (13)	0.73 ± (0.010)	0.515 ± (0.013)	1314 ± (18)	0.79 ± (0.010)	0.227 ± (0.011)
NAM	1809 ± (8)	0.73 ± (0.010)	0.372 ± (0.014)	1247 ± (26)	0.81 ± (0.012)	0.231 ± (0.007)
EBM	1944 ± (21)	0.73 ± (0.010)	0.511 ± (0.013)	1290 ± (26)	0.83 ± (0.012)	0.201 ± (0.016)
NodeGAM	1942 ± (21)	0.72 ± (0.006)	0.513 ± (0.027)	1308 ± (29)	0.81 ± (0.012)	0.205 ± (0.015)
GAMLSS	1321 ± (30)	0.78 ± (0.009)	0.392 ± (0.005)	391 ± (126)	0.77 ± (0.014)	0.119 ± (0.006)
gamboostLSS	1191 ± (30)	*0.79* ± (0.008)	0.370 ± (0.007)	*	*	*
DDNN	1230 ± (48)	0.73 ± (0.002)	*0.342* ± (0.012)	-211 ± (364)	0.81 ± (0.011)	0.145 ± (0.012)
NAMLSS[1]	1201 ± (41)	0.73 ± (0.010)	0.347 ± (0.019)	*-220* ± (210)	**0.85** ± (0.040)	*0.111* ± (0.015)
NAMLSS[2]	**1160** ± (49)	0.72 ± (0.008)	**0.328** ± (0.013)	**-237** ± (219)	*0.84* ± (0.020)	**0.107** ± (0.003)

* gamboostLSS was not able to execute.

Table 5: Benchmark results for the California Housing and the Abalone dataset. We report continuous ranked probability score (CRPS) as there exists a close form solution. For metrics detecting the accuracy of point predictions, we find the models that specifically minimize the MSE to excel.

Model		CA Housing			Abalone	
	LL↓	MSE ↓	CRPS ↓	LL ↓	MSE ↓	CRPS ↓
MLP	4191 ± (42)	**0.197** ± (0.005)	0.264 ± (0.004)	966.2 ± (27)	0.475 ± (0.044)	0.472 ± (0.013)
XGBoost	4219 ± (40)	0.211 ±(0.005)	0.271 ± (0.002)	982.0 ± (33)	0.515 ± (0.028)	0.504 ± (0.015)
NAM	4251 ± (43)	0.273 ±(0.037)	0.370 ± (0.045)	956.8 ± (22)	*0.454* ± (0.024)	0.484 ± (0.015)
EBM	4202 ± (42)	0.203 ±(0.004)	0.297 ± (0.002)	965.1 ± (22)	0.474 ± (0.025)	0.491 ± (0.012)
NodeGAM	4206 ± (89)	0.242 ± (0.007)	0.350 ± (0.004)	958.3 ± (23)	0.461± (0.033)	0.486 ± (0.017)
GAMLSS	3512 ± (67)	0.398 ± (0.040)	0.315 ± (0.005)	870.8 ± (16)	0.497 ± (0.032)	0.353 ± (0.007)
gamboostLSS	3812 ± (52)	0.415 ± (0.024)	0.336 ± (0.004)	815.1 ± (29)	0.524 ± (0.039)	0.370 ± (0.011)
DDNN	2681 ± (1279)	*0.197* ± (0.005)	**0.193** ± (0.002)	897.2 ± (159)	**0.444** ± (0.026)	**0.338** ± (0.009)
NAMLSS[1]	*2667* ± (91)	0.245 ± (0.004)	0.287 ± (0.019)	*869.8* ± (118)	0.496 ± (0.042)	0.357 ± (0.013)
NAMLSS[2]	**2329** ± (176)	0.265 ± (0.005)	*0.264* ± (0.009)	**802.3** ± (41)	0.486± (0.043)	*0.353* ± (0.014)

8.3.1 Synthetic Benchmarks

The benchmark study for used real-world datasets was performed under similar conditions. All datasets are publicly available and we describe every preprocessing step as well as all model specifications in detail in the following.

Synthetic Data Generation For the simulation of the data, respectively their underlying distribution parameters $\theta = \left(\theta^{(1)}, \theta^{(2)}, \theta^{(3)}, \theta^{(4)}\right)$, the following assumptions are made:

$$\begin{aligned}
\theta^{(1)} &= \frac{30}{13} x_1 \left((3x_2 + 1.5) - 2\sin\left(\frac{x_3}{2}\right)\right)^{-1} + \frac{113}{115} x_4 + 0.1 x_5, \\
\theta^{(2)} &= \exp\left(-0.0035 x_1 + (x_2 - 0.23)^2 - 1.42 x_3\right) + 0.0001 x_4, \\
\theta^{(3)} &= \frac{1}{42}(4x_1 - 90x_2), \\
\theta^{(4)} &= exp\left(0.0323 * x_2 + 0.0123 - 0.0234 * x_4\right),
\end{aligned}$$

where each of the five input vectors x_j is sampled from a uniform distribution $\mathcal{U}(0,1)$, with a total of $n = 3000$ observations per data set.

Table 6: Benchmark results for the Insurance dataset and munich dataset.

Model	Inv. Gamma					
		Melbourne			Munich	
	LL ↓	Gamma dev. ↓	KL-divergence ↓	LL ↓	Gamma dev. ↓	KL-divergence ↓
MLP	22999 ± (232)	1.090 ± (0.52)	5.23 ± (1.09)	6827 ± (178)	0.55 ± (0.04)	0.119 ± (0.034)
XGBoost	20471 ± (242)	1.088 ± (0.498)	5.23 ± (1.17)	5618 ± (152)	**0.48** ± (0.09)	0.125 ± (0.015)
NAM	25375± (844)	0.871± (0.209)	4.70± (1.93)	5892 ± (37)	0.72 ± (0.10)	0.169 ± (0.048)
EBM	20361± (207)	1.206± (0.651)	5.59 ± (0.64)	5474 ± (56)	*0.49* ± (0.09)	0.117 ± (0.021)
NodeGAM	20790± (29)	1.110 ± (0.466)	5.66± (0.59)	5984 ± (135)	0.57 ± (0.05)	0.112 ± (0.059)
GAMLSS	26353 ± (45)	**0.737** ± (0.069)	4.30 ± (0.01)	5419 ± (61)	0.53 ± (0.06)	0.148 ± (0.056)
gamboostLSS	26436 ± (48)	*0.761* ± (0.044)	4.29 ± (0.02)	5421± (33)	0.58 ± (0.10)	0.148 ± (0.056)
DDNN	26353± (45)	0.987± (0.304)	4.21± (2.02)	5555 ± (34)	0.69 ± (0.04)	*0.011* ± (0.015)
NAMLSS[1]	**19517**± (68)	0.962± (0.272)	*4.23*± (2.09)	**5383** ± (24)	0.59 ± (0.09)	0.014 ± (0.018)
NAMLSS[2]	*19675*± (67)	0.966± (0.285)	**4.18**± (2.06)	*5422* ± (22)	0.59 ± (0.10)	**0.011** ± (0.015)

Table 7: Benchmark results for the Telco and the -. We report continuous ranked probability score (CRPS) for the logistic distribution as there exists a close form solution as well as the AUC score.

Model	Logistic			Normal		
		Telco			Insurance	
	LL ↓	AUC ↑	CRPS ↓	LL ↓	MSE ↓	CRPS ↓
MLP	1027 ± (19)	0.69 ± (0.012)	0.307 ± (0.012)	266.8 ± (11)	*0.17* ± (0.027)	0.276 ± (0.019)
XGBoost	1123 ± (22)	0.73 ± (0.005)	0.331 ± (0.014)	266.8 ± (9)	0.19 ±(0.024)	0.248 ± (0.012)
NAM	1023 ± (28)	0.76 ± (0.011)	0.279 ± (0.008)	474.4 ± (73)	0.26 ±(0.031)	0.381 ± (0.022)
EBM	1094 ± (22)	*0.76* ± (0.007)	0.304 ± (0.012)	263.8 ± (10)	**0.14** ±(0.018)	**0.204** ± (0.009)
NodeGAM	1097 ± (27)	0.76 ± (0.020)	0.303 ± (0.019)	279.1 ± (11)	0.26 ± (0.031)	0.366 ± (0.026)
GAMLSS	85± (173)	**0.83** ± (0.011)	3.38 ± (6.369)	175.5 ± (28)	0.241 ± (0.033)	0.263 ± (0.020)
gamboostLSS	*	*	*	173.0 ± (28)	0.268 ± (0.044)	0.248 ± (0.019)
DDNN	27 ± (314)	0.73 ± (0.015)	0.188 ± (0.015)	178.2 ± (30)	0.17 ± (0.028)	*0.206* ± (0.016)
NAMLSS[1]	**-22** ± (137)	0.64 ± (0.018)	**0.133** ± (0.013)	*172.7* ± (23)	0.27 ± (0.044)	0.257 ± (0.020)
NAMLSS[2]	*-11* ± (114)	0.65 ± (0.017)	*0.135* ± (0.008)	**172.6** ± (20)	0.27 ± (0.049)	0.257 ± (0.022)

* gamboostLSS was not able to execute.

8.4 Activation Functions

For DDNN and NAMLSS, independent of the implementation, we use a Softplus activation for the scale parameter σ^2 to ensure non-negativity and a linear activation for the mean μ.

For the AirBnB datasets, also analyzed by Rügamer et al. (2020), we assume an Inverse Gamma distribution $\mathcal{IG}(\alpha, \beta)$ as the underlying data distribution (see equation (8.2.1) for the log-likelihood). For NAMLSS as well as DDNN we have to adjust the activation functions, as both models minimize the log-likelihood via the parameters α and β. However, the mean prediction resulting from these parameters is defined via:

$$\mu = \frac{\beta}{\alpha - 1}$$

and is hence only defined for $\alpha > 1$. The activation functions thus need to ensure an α prediction that is larger than 1 and a β prediction that is larger than 0. Hence we again use a Softplus activation for the β output layer [5]. For the α prediction, we use the following activation function element-wise:

$$h(x) = \begin{cases} \log(1 + \exp(x)), & \text{if } \log(1 + \exp(x)) > 1, \\ \frac{1}{\log(1+\exp(x))}, & \text{else.} \end{cases} \tag{4}$$

To compute the log-likelihood for the models resulting in a mean prediction we compute the parameters α and β as follows:

$$\alpha = \frac{\mu^2}{\sigma^2 + 2},$$
$$\beta = \mu \frac{\mu^2}{\sigma^2 + 1},$$

with σ^2 denoting the variance of the mean predictions. For XGBoost and EBM we use a simple transformation of the target variable to ensure that $\mu > 0$. Hence we fit the model on $\log(y)$ and re-transform the predictions accordingly with $\exp(\hat{y})$.

For a (binary) classification benchmark we use the FICO dataset (FICO, 2018), the Shrutime dataset and the Telco dataset. A logistic distribution, $\mathcal{LO}(\mu, s)$, of the underlying response variable was assumed (see equation (8.2.1) for the log-likelihood). Again, we use the true standard deviation of the underlying data for the models only resulting in a mean prediction. The models resulting in a mean prediction use binary cross-entropy as the loss function and hence a sigmoid activation function on the output layer.

8.4.1 Preprocessing

We implement the same preprocessing for all used datasets and only slightly adapt the preprocessing of the target variable for the two regression problems, California Housing and Insurance. We closely follow Gorishniy et al. (2021) in their preprocessing steps and use the preprocessing also implemented by Agarwal et al. (2021). Hence all numerical variables are scaled between -1 and 1, all categorical features are one-hot encoded. In contrast to Gorishniy et al. (2021) we do not implement quantile smoothing, as one of the biggest advantages of neural models is the capability to model jagged shape functions. We use 5-fold cross-validation and report mean results as well as the standard deviations over all datasets. For reproducibility, we use the sklearn (Pedregosa et al., 2011) Kfold function with a random state of 101 and shuffle equal to true for all datasets. For the two regression datasets, we implement a standard normal transformation of the target variable. This results in better performances in terms of log-likelihood for all models only predicting a mean and is hence even disadvantageous for the presented NAMLSS framework.

8.4.2 Datasets

[5] Interestingly, the NAM did not converge using the Softplus activation function as the MLP did. Using the Softplus activation resulted in tremendously large mean gamma deviances and log-likelihoods, as the model kept predicting values that were nearly zero. Hence, we were only able to achieve good results for the NAM using the activation function given by formula (4).

Table 8: Statistics of the benchmarking datasets.

Dataset	No. Samples	No. Features	Distribution	Task
California Housing	20640	8	Normal $\mathcal{N}(\mu, \sigma)$	Regression
Insurance	1338	6	Normal $\mathcal{N}(\mu, \sigma)$	Regression
Abalone	4177	10	Normal $\mathcal{N}(\mu, \sigma)$	Regression
Munich	4568	9	Inverse Gamma $\mathcal{IG}(\alpha, \beta)$	Regression
Melbourne	16868	11	Inverse Gamma $\mathcal{IG}(\alpha, \beta)$	Regression
Fico	10459	23	Logistic $\mathcal{LO}(\mu, s)$	Classification
Shrutime	10000	10	Logistic $\mathcal{LO}(\mu, s)$	Classification
Telco	7032	19	Logistic $\mathcal{LO}(\mu, s)$	Classification

California Housing The California Housing (CA Housing) dataset Pace and Barry (1997) is a popular publicly available dataset and was obtained from sklearn Pedregosa et al. (2011). It is also used as a benchmark in Agarwal et al. (2021) and Gorishniy et al. (2021) and we achieve similar results concerning the MSE for the models which were used in both publications. The dataset contains the house prices for California homes from the U.S. census in 1990. The dataset is comprised of 20640 observations and besides the logarithmic median house price of the blockwise areas as the target variable contains eight predictors. As described above, we additionally standard normalize the target variable. All other variables are preprocessed as described above.

Insurance The Insurance dataset is another regression type dataset for predicting billed medical expenses (Lantz, 2019). The dataset is publicly available in the book *Machine Learning with R* by Lantz (2019). Additionally, the data is freely available on https://github.com/stedy/Machine-Learning-with-R-datasetsGithub and https://www.kaggle.com/code/gloriousc/insurance-forecast-by-using-linear-regression/dataKaggle. It is a small dataset with only 1338 observations. The target variable is *charges*, which represents the *Individual medical costs billed by health insurance*. Similar to the California Housing regression we standard normalize the response. Additionally, the dataset includes 6 feature variables. They are preprocessed as described above, which, due to one-hot encoding leads to a feature matrix with 9 columns.

Abalone The Abalone dataset contains information for the prediction of the age of abalone, a type of sea snail, based on their physical measurements. The data set is taken from the original publication (Nash et al., 1994) and today is a part of https://archive.ics.uci.edu/ml/datasets/abaloneUCI Machine Learning Repository. A dataset of 4177 observations, 10 features and one response variable is obtained after processing the data.

Munich For the AirBnB data, we orientate on Rügamer et al. (2020) and used the data for the city of Munich. The dataset is also publicly available and was taken from http://insideairbnb.com/get-the-data/Inside AirBnB on January 15, 2023. After excluding the variables *ID*, *Name*, *Host ID*, *Host Name*, *Last Review* and after removing rows with missing values the dataset contains 4568 observations. Additionally, we drop the *Neighbourhood* variable as firstly the predictive power of that variable is limited at best and secondly not to create too large feature matrices for GAMLSS. Hence, in addition to the target variable, the dataset contains 9 variables. All preprocessing steps are subsequently performed as described above and the target variable, *Price*, is not preprocessed at all.

Melbourne The dataset is also publicly available and was taken from http://insideairbnb.com/get-the-data/Inside AirBnB. The second Airbnb dataset (Melbourne) originates from the same source as the Munich Airbnb dataset. The data processing follows the same procedure as described in the Munich section. All preprocessing steps are then performed as described above and the target variable *Price* is not preprocessed at all.

FICO Similar to Agarwal et al. (2021) we also use the FICO dataset for our benchmarking study. However, we use it as described on the website and hence use the *Risk Performance* as the target variable. A detailed description of the features and their meaning is available at the https://community.fico.com/s/explainable-machine-learning-challengeExplainable Machine Learning Challenge. The dataset is comprised of 10459 observations. We did not implement any preprocessing steps for the target variable.

Shrutime This dataset contains information on the customers of a bank and the target variable is a binary variable reflecting whether the customer has left the bank (closed his account) or remains a customer. The corresponding data set can be found at https://www.kaggle.com/datasets/shrutimechlearn/churn-modellingKaggle and is introduced by Kaggle (2019). After the processing described above, the set consists of 10000 observations, each with 10 features.

Telco The Telco customer churn data contains information about a fictitious telco company that provided home phone and internet services to 7043 customers. It details which customers left, stayed or signed up for their service. Several key demographics are included for each customer, as well as a satisfaction score, a churn score and a customer lifetime value (CLTV) index and was introduced by IBM (2019). After the processing described above, the set consists of 7043 observations, each with 19 features.

8.4.3 Model Architectures & Hyperparameters

As we do not implement extensive hyperparameter tuning for the presented NAMLSS framework, we do not perform hyperparameter tuning for the comparison models. We fit all models without an intercept. However, we try to achieve the highest comparability by choosing similar modelling frameworks, network architectures and hyperparameters where possible. All neural models are hence fit with identical learning rates, batch sizes, hidden layer sizes, activation functions and regularization techniques. Through all neural models and all datasets, we use the ADAM optimizer (Kingma and Ba, 2014) with a starting learning rate of 1e-04. For the larger datasets, *California Housing*, *Abalone*, *FICO*, *Telco* and *Shrutime* we orient on Agarwal et al. (2021) and use larger batch sizes of 1024. For the smaller dataset, *Insurance*, we use a smaller batch size of 256 and for the *Munich* and *Melbourne* dataset we use a batch size of 512. For every dataset and for every neural model, the maximum number of epochs is set to 2000. However, we implement early stopping with a patience of 150 epochs and no model over no fold and no dataset ever trained for the full 2000 epochs. Additionally, we reduce the learning rate with a factor of 0.95 with patience of 10 epochs for all models for all datasets. We use the Rectified Linear Unit (ReLU) activation function for all hidden layers for all models:

$$h(x) = \begin{cases} 0, & x < 0 \\ x, & \text{else.} \end{cases}$$

We also experimented with the Exponential centred hidden Unit (ExU) activation function presented by Agarwal et al. (2021) but found no improvement in model performance and even a slight deterioration for most models.

For the statistical models used from the GAMLSS and gamboostLSS frameworks, we do not optimize the model hyperparameters, as with neural networks. We use the respective default settings unless otherwise stated in the modelling descriptions included in the Appendix. We try to keep the model settings equal between all models, if applicable. All GAMLSS models use the same RS solver proposed by Rigby and Stasinopoulos (2005), in cases where this approach does not lead to convergence, the alternative CG solver presented by Cole and Green (1992) is employed. To exclude possible numerical differences, the same distributions from the GAMLSS R package are used for modelling the response distribution and calculating the log-likelihoods. gamboostLSS allows the use of different boosting approaches. Here we use the implemented boosting methods based on GAMs and GLMs and choose the model that performs better in terms of log-likelihood and the assumed loss.

Table 9: Hyperparameters for the neural models for the California Housing and the Abalone dataset

Hyperparameter	NAMLSS[1]	NAMLSS[2]	DNN	MLP	NAM
Learning Rate	1e-04	1e-04	1e-04	1e-04	1e-04
Dropout	0.25	0.25	0.25	0.25	0.25
Hidden Layers	$[1000, 500, 100, 50, 25]$	$[1000, 500, 100, 50, 25]$	$[1000, 500, 100, 50, 25]$	$[1000, 500, 100, 50, 25]$	$[1000, 500, 100, 50, 25]$
LR Decay, Patience	0.95 - 10	0.95 - 10	0.95 - 10	0.95 - 10	0.95 - 10
Activation	ReLU	ReLU	ReLU	ReLU	ReLU
Output Activation	Linear, Softplus	Linear, Softplus	Linear, Softplus	Linear	Linear

[1] With 2 $\times$8 subnetworks. See Table 6 for an exemplary network structure.
[2] With 8 subnetworks and each subnetwork returning a parameter for the location and shape respectively. See Table 2 for an exemplary network structure.

California Housing and Abalone We orient again on Agarwal et al. (2021) and use the following hidden layer sizes for all networks: $[1000, 500, 100, 50, 25]$. The second hidden layer is followed by a 0.25 dropout layer.

While subsequently the NAM and NAMLSS have much more trainable parameters than the MLP and the DNN, we find that the MLP and DNN outperform the NAM and NAMLSS in terms of mean prediction. Additionally, we encountered severe overfitting when using the same number of parameters in an MLP as in the NAM and NAMLSS implementation. For the mean predicting models, we use a one-dimensional output layer with a linear activation. For the DNN and both NAMLSS implementations, we use a linear activation over the mean prediction and a Softplus activation for the variance prediction with:

$$h(x) = \log(1 + \exp(x)).$$

For the NAMLSS implementation depicted in Figure 6 we use a smaller network structure for predicting the variance with two hidden layers of sizes 50 and 25 without any form of regularization as Dürr et al. (2020) found that using smaller networks for predicting the scale parameters is sufficient. For XGBoost we use the default parameters from the Python implementation. For the Explainable Boosting machines, we increased the number of maximum epochs to the default value of 5000 but set the early stopping patience considerably lower to 10, as otherwise, the model reached far worse results compared to the other models. We additionally increased the learning rate to 0.005 compared to the learning rate used in the neural approaches as a too small learning rate resulted in bad results. Otherwise, we kept all other hyperparameters as the default values. The GAMLSS and gamboostLSS models assume a normal distribution, with a location estimator μ employing an identity link and a scale estimator σ with a log-link function. Due to numerical instabilities, we choose to use the GLM-based boosting method instead of the default GAM-based version.

Table 10: Hyperparameters for the neural models for the Insurance dataset

Hyperparameter	NAMLSS[1]	NAMLSS[2]	DNN	MLP	NAM
Learning Rate	1e-04	1e-04	1e-04	1e-04	1e-04
Dropout	0.5	0.5	0.5	0.5	0.5
Hidden Layers	$[250, 50, 25]$	$[250, 50, 25]$	$[250, 50, 25]$	$[250, 50, 25]$	$[250, 50, 25]$
LR Decay, Patience	0.95 - 10	0.95 - 10	0.95 - 10	0.95 - 10	0.95 - 10
Activation	ReLU	ReLU	ReLU	ReLU	ReLU
Output Activation	Linear, Softplus	Linear, Softplus	Linear, Softplus	Linear	Linear

[1] With 2 $\times 9$ subnetworks. See Table 6 for an exemplary network structure.
[2] With 9 subnetworks and each subnetwork returning a parameter for the location and shape respectively. See Table 2 for an exemplary network structure.

Insurance As the insurance dataset is considerably smaller than all other datasets we use slightly different model structures, as the model structure used for the California Housing and Abalone datasets led to worse results. Hence, for all neural models, we use hidden layers of sizes $[250, 50, 25]$. The first layer is followed by a 0.5 dropout layer. Again, we use a simple linear activation for the models only predicting the mean and a linear and a Softplus activation for the models predicting the mean and the variance respectively. For the first NAMLSS implementation (see Figure 6) we again use a smaller network for predicting the variance with just one hidden layer with 50 neurons.

For XGBoost and EBM we use the same hyperparameter specifications as for the California Housing and Abalone datasets.

The GAMLSS and gamboostLSS models assume a normal distribution, with a location estimator μ employing an identity link and a scale estimator σ with a log-link function. The boosting for location, scale and shape method employed uses the GLM based, instead of the GAM, based version.

FICO, Telco and Shrutime For the logistic datasets, we use the exact same model structure as for the Insurance dataset, as the model structures implemented for the California Housing dataset resulted in worse results. However, as it is a binary classification problem we use a Sigmoid activation for the MLP as well as the NAM. For the DNN and both NAMLSS implementations, we use a Sigmoid activation for the location and a Softplus activation for the scale. To generate the log-likelihoods for the models only predicting a mean, we again use the true standard deviation of the underlying data.

For XGBoost and EBM we had to adjust the hyperparameters in order to get results comparable to the MLP, NAM or NAMLSS. Hence, for EBM we use 10 as the maximum number of leaves, 100 early stopping rounds and again the same learning rate of 0.005.

For XGboost we use 500 estimators with a maximum depth of 15. η is set to 0.05.

For the GAMLSS and gamboost models we use a logistic distribution to model the response distribution, where μ estimator uses identity and the σ estimator uses a log-link function.

Table 11: Hyperparameters for the neural models for the FICO, Telco and Shrutime datasets

Hyperparameter	NAMLSS[1]	NAMLSS[2]	DNN	MLP	NAM
Learning Rate	1e-04	1e-04	1e-04	1e-04	1e-04
Dropout	0.5	0.5	0.5	0.5	0.5
Hidden Layers	$[250, 50, 25]$	$[250, 50, 25]$	$[250, 50, 25]$	$[250, 50, 25]$	$[250, 50, 25]$
LR Decay, Patience	0.95 - 10	0.95 - 10	0.95 - 10	0.95 - 10	0.95 - 10
Activation	ReLU	ReLU	ReLU	ReLU	ReLU
Output Activation	Sigmoid, Softplus	Sigmoid, Softplus	Sigmoid, Softplus	Sigmoid	Sigmoid

[1] With 2 ×23 subnetworks. See Table 6 for an exemplary network structure.
[2] With 23 subnetworks and each subnetwork returning a parameter for the location and shape respectively. See Table 2 for an exemplary network structure.

Munich and Melbourne We fit the AirBnB datasets, with an Inverse Gamma distribution where applicable. However, we train the models that only predict the mean with the squared error loss function. While one might suspect worse performances due to that, we find that using the squared error actually leads to much smaller gamma deviances compared to the models leveraging the Inverse Gamma distribution. Additionally, we use slightly smaller model structures than for the California Housing dataset. For all neural models, we use hidden layers of sizes $[512, 256, 50]$. The first hidden layer is followed by a 0.5 dropout layer. Throughout the hidden layers, we use ReLU activation functions. However, we deviate from that for the output layer activation functions. For the MLP we use a Softplus activation function for the output layer, ensuring that strictly positive values are predicted. For NAMLSS as well as the DNN we have to adjust the activation functions, as both models minimize the log-likelihood via the parameters α and β. However, the mean prediction resulting from these parameters is defined via:

$$\mu = \frac{\beta}{\alpha - 1}$$

and is hence only defined for $\alpha > 1$. The activation functions thus need to ensure a α prediction that is larger than 1 and a β prediction that is larger than 0. Hence we again use a Softplus activation for the β output layer. For the α prediction, we use the following activation function element-wise:

$$h(x) = \begin{cases} \log(1 + \exp(x)), & \text{if } \log(1 + \exp(x)) > 1 \\ \frac{1}{\log(1+\exp(x))}, & \text{else.} \end{cases}$$

To compute the log-likelihood for the models resulting in a mean prediction we compute the parameters α and β as follows:

$$\alpha = \frac{\mu^2}{\sigma^2 + 2},$$
$$\beta = \mu \frac{\mu^2}{\sigma^2 + 1},$$

with σ^2 denoting the variance of the mean predictions.

For XGBoost and EBM we use a simple transformation of the target variable in order to ensure that $\mu > 0$. Hence we fit the model on $\log(y)$ and re-transform the predictions accordingly with $\exp(\hat{y})$. Otherwise, we use the same hyperparameters as for the California Housing dataset.

Interestingly, the NAM did not converge using the Softplus activation function as the MLP did. Using the Softplus activation resulted in tremendously large mean gamma deviances and log-likelihoods, as the model kept predicting values that were nearly zero. Hence, we were only able to achieve good results for the NAM using the activation function given by formula (4).

The presented GAMLSS and gamboostLSS models assume an Inverse Gamma distribution with both μ and σ utilizing the log-link function. It should be noted that the RS algorithm does not converge with GAMLSS, which is why CG is used.

Table 12: Hyperparameters for the neural models for the Munich and Melbourne datasets

Hyperparameter	NAMLSS[1]	NAMLSS[2]	DNN	MLP	NAM
Learning Rate	1e-04	1e-04	1e-04	1e-04	1e-04
Dropout	0.5	0.5	0.5	0.5	0.5
Hidden Layers	$[512, 256, 50]$	$[512, 256, 50]$	$[512, 256, 50]$	$[512, 256, 50]$	$[512, 256, 50]$
LR Decay, Patience	0.95 - 10	0.95 - 10	0.95 - 10	0.95 - 10	0.95 - 10
Activation	ReLU	ReLU	ReLU	ReLU	ReLU
Output Activation	Gamma*, Softplus	Gamma*, Softplus	Gamma*, Softplus	Linear	Linear

[1] With 2 $\times$23 subnetworks. See Table 6 for an exemplary network structure.

[2] With 23 subnetworks and each subnetwork returning a parameter for the location and shape respectively. See Table 2 for an exemplary network structure.

* See formula (4) for the detailed element-wise activation function.