Neural Additive Models: Interpretable Machine Learning with Neural Nets

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep neural networks (DNNs) are powerful black-box predictors that have achieved impressive performance on a wide variety of tasks. However, their accuracy comes at the cost of intelligibility: it is usually unclear how they make their decisions. This hinders their applicability to high stakes decision-making domains such as healthcare. We propose Neural Additive Models (NAMs) which combine some of the expressivity of DNNs with the inherent intelligibility of generalized additive models. NAMs learn a linear combination of neural networks that each attend to a single input feature. These networks are trained jointly and can learn arbitrarily complex relationships between their input feature and the output. Our experiments on regression and classification datasets show that NAMs are more accurate than widely used intelligible models such as logistic regression and shallow decision trees. They perform similarly to existing state-of-the-art generalized additive models in accuracy, but are more flexible because they are based on neural nets instead of boosted trees. To demonstrate this, we show how NAMs can be used for multitask learning on synthetic data and on the COMPAS recidivism data due to their composability, and demonstrate that the differentiability of NAMs allows them to train more complex interpretable models for COVID-19.

1 Introduction

Deep neural networks are powerful function approximators that have achieved impressive results on a variety of tasks, such as computer vision [17], language modeling [31] and reinforcement learning [38]. However, it is notoriously difficult to understand how they make their predictions, and they are often considered as black-box models. This hinders their applicability to high-stakes domains such as healthcare, finance and criminal justice.

Various efforts have been made to demystify the predictions of neural networks (NNs). For example, one family of methods, represented by LIME [33], attempt to *explain* individual predictions of a neural network by approximating it locally with interpretable models such as linear models and shallow trees[1]. However, these approaches often fail to provide a global view of the model and their explanations often are not faithful to what the original model computes or do not provide enough detail to understand the model's behavior [35]. In this paper, we make restrictions on the *structure* of neural networks, which yields a family of models called Neural Additive Models (NAMs), that are inherently interpretable while suffering little loss in prediction accuracy when applied to tabular data.

Methodologically, NAMs belong to a larger model family called Generalized Additive Models (GAMs) [14]. GAMs have the form:

$$g(\mathbb{E}[y]) = \beta + f_1(x_1) + f_2(x_2) + \cdots + f_K(x_K) \tag{1}$$

[1]Linear models, shallow decision trees and GAMs are interpretable only if the features they are trained on are interpretable.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

where $\mathbf{x} = (x_1, x_2, \ldots, x_K)$ is the input with K features, y is the target variable, $g(.)$ is the link function (*e.g.,* logistic function) and each f_i is a univariate shape function with $\mathbb{E}[f_i] = 0$. Generalized linear models, such as logistic regression, are a special form of GAMs where each f_i is restricted to be linear.

NAMs learn a linear combination of networks that each attend to a single input feature: each f_i in (1) is parametrized by a neural network. These networks are trained jointly using back-propagation and can learn arbitrarily complex shape functions. Interpreting NAMs is easy as the impact of a feature on the prediction does not rely on the other features and can be understood by visualizing its corresponding shape function (*e.g.,* plotting $f_i(x_i)$ *vs.* x_i).

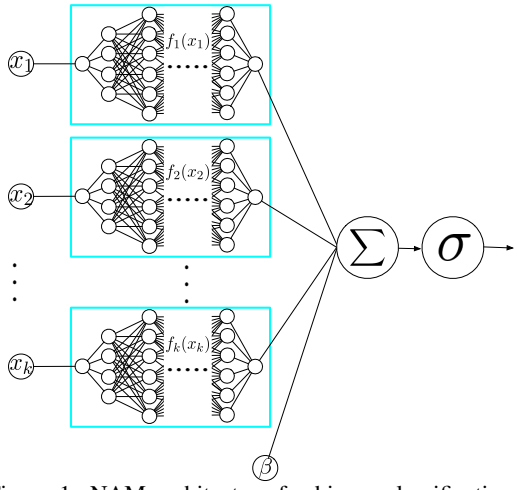


Figure 1: NAM architecture for binary classification. Each input variable is handled by a different neural network. This results in easily interpretable yet highly accurate models.

Traditionally, GAMs were fitted via iterative backfitting using smooth low-order splines, which reduce overfitting and can be fit analytically. More recently, GAMs [5] were fitted with boosted decision trees to improve accuracy and to allow GAMs to learn jumps in the feature shaping functions to better match patterns seen in real data that smooth splines could not easily capture. This paper examines using DNNs to fit generalized additive models (NAMs) which provides the following advantages:

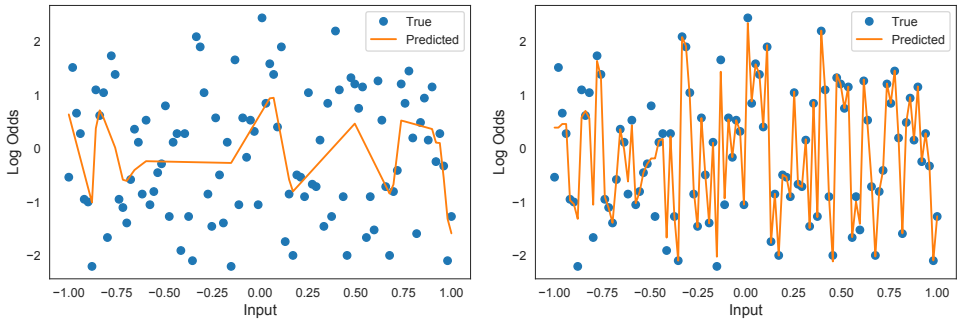
- NAMs[2] introduce an expressive yet intelligible class of models to the deep learning (DL) community, a much larger community than the one using tree-based GAMs.
- NAMs are likely to be combined with other DL methods in ways we don't foresee. This is important because a key drawback of deep learning is interpretability. For example, NAMs have already been employed for survival analysis [45] while CNN-LSTM based extension of NAMs have already been developed for genomics [40].
- NAMs, due to the flexibility of NNs, can be easily extended to various settings problematic for boosted decision trees. For example, extending boosted tree GAMs to multitask, multi-class or multi-label learning requires significant changes to how trees are trained, but is easily accomplished with NAMs without requiring changes to how neural nets are trained due to their composability (Section 4.2). Futhermore, the differentiability of NAMs allows them to train more complex interpretable models for COVID-19 (Section 4.1).
- The graphs learned by NAMs are not just an explanation but an exact description of how NAMs compute a prediction. This would help harness the expressivity of neural nets on high-stakes domains with intelligibility requirements, *e.g.,* NAMs for in-hospital mortality prediction [22].
- NAMs are more scalable as they can be trained on GPUs or other specialized hardware using the same toolkits developed for deep learning over the past decade – GAMs currently cannot.
- Accurate GAMs [5] currently require millions of decision trees to fit each shape function while NAMs only use a small ensemble (2 - 100) of neural nets.

2 Neural Additive Models

2.1 Fitting Jagged Shape Functions

Modeling jagged functions is required to learn accurate additive models as there are often sharp jumps in real-world datasets, *e.g.,* see Figure 4 for jumps in graphs for PFRatio and Bilirubin which correspond to real patterns in the MIMIC-II dataset [37] (Section A.1). Similarly, Caruana et al. [5] observe that GAMs fit using splines tend to over regularize and miss genuine details in real data, yielding less accuracy than tree-based GAMs. Therefore, we require that neural networks (NNs) are able to learn highly non-linear shape functions, to fit these patterns.

[2]Open-source code: `https://anonymous.4open.science/r/NAMs`



(a) (b)

Figure 2: **Accurately Fitting the Toy Dataset**: Training predictions learned by a single hidden layer neural network with 1024 (a) standard ReLU, and (b) ReLU-n with ExU hidden units trained for 10,000 epochs on the binary classification dataset described in Section 2.1. We can see that the ReLU network has learned a fairly smooth function while the ExU network has learned a very jumpy function. We find that a DNN with multiple hidden layers also learned smooth functions (see Figure A.3).

Although NNs can approximate arbitrarily complex functions [18], we find that standard NNs fail to model highly jumpy 1D functions, and demonstrate this failure empirically using a toy dataset. The toy dataset is constructed as follows: For the input x, we sample 100 evenly spaced points in [-1, 1]. For each x, we sample p uniformly random in [0.1, 0.9) and generate 100 labels from a Bernoulli random variable which takes the value 1 with probability p. This creates a binary classification dataset of (x, y) tuples with 10,000 points. Figure 2 shows the log-odds of the empirical probability p (*i.e.,* $\log \frac{p}{1-p}$) of classifying the label of x as 1 for each input x. This dataset tests the NN's ability to "overfit" the data, rather than its ability to generalize.

Over-parameterized NNs with ReLUs [25] and standard initializations such as Kaiming initialization [16] and Xavier initialization [10] struggle to overfit this dataset when trained using mini-batch gradient descent, despite the NN architecture being expressive enough[3](see Figures 2(a) and A.3). This difficulty of learning large local fluctuations with ReLU networks without affecting their global behavior when fitting jagged functions might be due to their bias towards smoothness [2, 32].

We propose *exp-centered* (ExU) hidden units to overcome this neural net failure: we simply learn the weights in the logarithmic space with inputs shifted by a bias. Specifically, for a scalar input x, each hidden unit using an activation function f computes $h(x)$ given by

$$h(x) = f\left(e^{w} * (x - b)\right) \tag{2}$$

where w and b are the weight and bias parameters. The intuition behind ExU units is as follows: For modeling jagged functions, a hidden unit should be able to change its output significantly, with a tiny change in input. This requires the unit to have extremely large weight values depending on the sharpness of the jump. The ExU unit computes a linear function of input where the slope can be very steep with small weights, making it easier to modify the output easily during training. ExU units do not improve the expressivity of neural nets, however they do improve their learnability for fitting jumpy functions. While we use ExU units to train accurate NAMs, they are more generally applicable for approximating jumpy functions with neural nets.

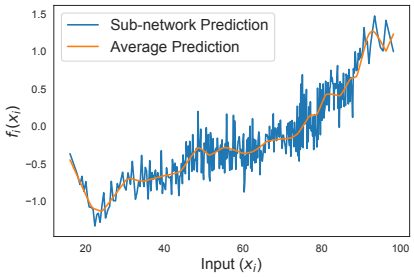


Figure 3: **Regularizing ExU networks.** Output of a ExU feature net trained with dropout = 0.2 for the age feature in the MIMIC-II dataset [37]. Predictions from individual subnets (as a result of dropping out hidden units) are much more jagged than the average predictions using the entire feature net.

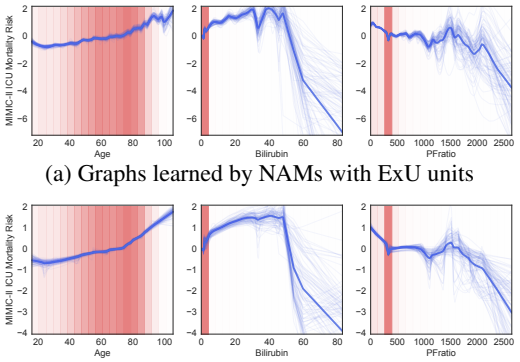
We noticed that ExU units with standard weight initialization also struggle to learn jagged curves; instead initializing the weights using a normal distribution $\mathcal{N}(x, 0.5)$ with $x \in [3, 4]$ works well in practice. This initialization simply ensures that the initial network starts with a jagged (random) function which we empirically find to be crucial for fitting any jumpy function. We also employ various regularization methods when fitting NAMs with ExUs including dropout, weight decay, output penalty, and feature dropout to avoid overfitting (see Section A.4 for an overview).

[3]This problem doesn't occur with full-batch gradient descent.

Furthermore, we use ReLU activations capped at n (ReLU-n) [21] to ensure that each ExU unit is active in a small input range, making it easier to model sharp jumps in a function without significantly affecting the global behavior. ExU-units can be combined with any activation function (*i.e.,* any f can be used in (2)), but ReLU-n performs well in practice. Figure 2(b) shows that NNs with ExU units are able to fit the toy dataset significantly better than standard NNs.

2.2 Intelligibility and Modularity

The intelligibility of NAMs results in part from the ease with which they can be visualized. Because each feature is handled independently by a learned shape function parameterized by a neural net, one can get a full view of the model by simply graphing the individual shape functions. For data with a small number of inputs, it is possible to have an accessible explanation of the model's behavior visualized fully on a single page. Please note these shape function plots are not just an explanation but an *exact* description of how NAMs compute a prediction.



(a) Graphs learned by NAMs with ExU units

(b) Graphs learned by NAMs with standard units

Figure 4: **ExU *vs.* standard** hidden units. On MIMIC-II, NAMs trained with ExU units learn jumpier graphs than with standard units while achieving a similar AUC (≈ 0.829). Ensembling them further improves performance (≈ 0.830). Note that white regions in the plots correspond to regions with low data density (typically a few points) and thus we see much higher variance in the learned shape functions. We present a detailed case study on the MIMIC-II dataset in Section A.1.

We set the average score for each graph (*i.e.,* each feature) averaged across the entire training dataset to zero by subtracting the mean score. To make individual shape functions identifiable and modular, a single bias term is then added to the model so that the average predictions across all data points matches the observed baseline. This makes interpreting the contribution of each term easier: *e.g.,* on binary classification tasks, negative scores decrease probability, and positive scores increase probability compared to the baseline probability of observing that class. This property also allows each graph to be removed from the NAM (zeroed out) without introducing bias to the predictions.

Visualization. We plot each shape function and the corresponding data density on the same graph. Specifically, we plot each learned shape function $f_k(x_k)$ *vs.* x_k for an ensemble of NAMs using a semi transparent blue line, which allows us to see when the models in the ensemble learned the same shape function and when they diverged. This provides a sense of the confidence of the learned shape functions. We also plot on the same graphs the normalized data density, in the form of pink bars. The darker the shade of pink, the more data there is in that region. This allows us to know when the model had adequate training data to learn appropriate shape functions.

3 Evaluating the Accuracy of NAMs

In this section, we evaluate the single-task learning capacity of NAMs against the following baselines on both regression and classification tasks:

- **Logistic / Linear Regression and Decision Trees (CART)**: Prevalent intelligible models. For both methods above we use the `sklearn` implementation [28], and tune the hyper-parameters with grid search.
- **Explainable Boosting Machines (EBMs)**: Current state-of-the-art GAMs [5, 23] which use gradient boosting of millions of shallow bagged trees that cycle one-at-a-time through the features.
- **Deep Neural Networks (DNNs)**: Unrestricted, full-complexity models which can model higher-order interaction between the input features. This gives us a sense of how much accuracy we sacrifice in order to gain interpretability with NAMs.
- **Gradient Boosted Trees (XGBoost)**: Another class of full-complexity models that provides an upper bound on the achievable test accuracy in our experiments. We use the XGBoost implementation [6].

Training and Evaluation. Feature nets in NAMs are selected amongst (1) DNNs containing 3 hidden layers with 64, 64 and 32 units and ReLU activation, and (2) single hidden layer NNs with

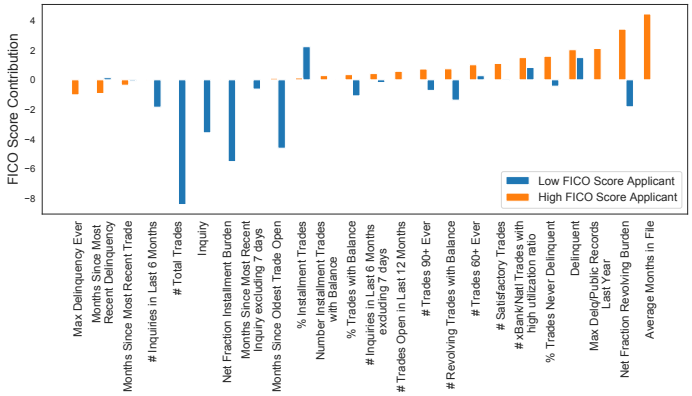


Figure 5: **Understanding individual predictions for credit scores**. Feature contribution using the learned NAMs for predicting scores of two applicants in the FICO dataset [9]. For a given input, each feature net in the NAM acts as a lookup table and returns a contribution term. These contributions are combined in a modular way: they are added up, and passed through a link function for prediction. the longer a person's credit history, the better it is for their credit score The high scoring applicant has a long credit history (Average Months on File), which contributes to their credit score better. On the contrary, the low scoring applicant used their credit quite frequently (Total Number of Trades) and has a large burden (Net Fraction Installment Burden), thus resulting in a low score.

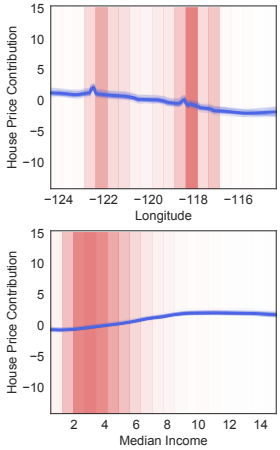


Figure 6: **California Housing**. Graphs learned by NAMs trained to predict house prices [27] for two most important features. As expected, The house prices increase linearly with median income in high data density regions. Furthermore, the graph for longitude shows sharp jumps in price prediction around the location of San Francisco and Los Angeles.

Model	MIMIC-II (AUC)	Credit (AUC)	CA Housing (RMSE)	FICO (RMSE)
Log./Linear Reg.	0.791 ± 0.007	0.975 ± 0.010	0.728 ± 0.015	4.344 ± 0.056
CART	0.768 ± 0.008	0.956 ± 0.004	0.720 ± 0.006	4.900 ± 0.113
NAMs	0.830 ± 0.008	0.980 ± 0.002	0.562 ± 0.007	3.490 ± 0.081
EBMs	0.835 ± 0.007	0.976 ± 0.009	0.557 ± 0.009	3.512 ± 0.095
XGBoost	0.844 ± 0.006	0.981 ± 0.008	0.532 ± 0.014	3.345 ± 0.071
DNNs	0.832 ± 0.009	0.978 ± 0.003	0.492 ± 0.009	3.324 ± 0.092

Table 1: Single-task learning Results. Means and standard deviations are reported from 5-fold cross validation. Higher AUCs and lower RMSEs are better. We report results on two widely used regression datasets, namely California Housing [27] for predicting housing prices and FICO [9] for understanding credit score predictions, as well as two classification datasets, namely Credit [7] for financial fraud detection and MIMIC-II [37] for predicting mortality in ICUs. We present a case study on the MIMIC-II dataset in Section A.1 and discuss the interpretations from NAMs on other datasets in Section A.2.

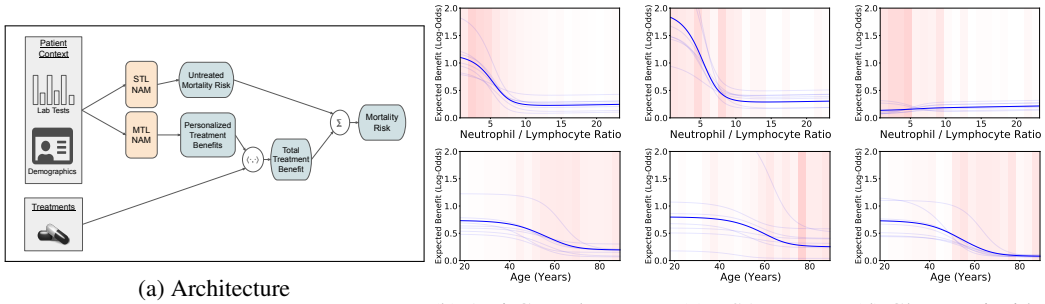
1024 ExU units and ReLU-1 activation. We perform 5-fold cross validation to evaluate the accuracy of the learned models. To measure performance in Table 1, we use area under the precision-recall curve (AUC) for binary classification and root mean-squared error (RMSE) for regression. More details about training and evaluation protocols can be found in Section A.5 in the appendix.

NAMs achieve comparable performance to EBMs on both classification and regression datasets, making them a competitive alternative to EBMs. Given this observation, we next look at some additional capabilities of NAMs that are not available to EBMs or any tree-based learning methods.

4 Unique Capabilities of NAMs

4.1 Intelligible Parameter Generation: Leveraging the Differentiability of NAMs

Medical treatment protocols are designed to deliver treatments to patients who would most benefit from them. To optimize treatment protocols, we would like a model which provides an intelligible map from patient information to an estimate of benefit for each potential treatment. To accomplish this, we use a NAM to generate parameters for personalized models of mortality risk given treatment (Fig. 7). By training to match predicted mortality risk with observed mortality, the NAM encodes expected treatment benefits as a function of patient information. NAMs are the only nonlinear GAM suitable for this application because NAMs are differentiable and can be trained via backpropagation.



(a) Architecture (b) Anti-Coagulants (c) NSAIDs (d) Glucocorticoids

Figure 7: **Estimating personalized treatment benefits for Covid-19 patients.** NAMs provide a unique combination of intelligibility and differentiability which make them suitable as a component in contextual parameter generation (a). By applying NAMs in this way, we are able to estimate and interpret personalized benefits of medical treatments for Covid-19 patients (b-d).

Figure 7 shows a NAM trained to predict treatment benefits for Covid-19 patients. We train the model on deidentified data from over 3000 Covid-19 patients. The model suggests that the benefits of anti-coagulants and NSAIDs decrease with increased Neutrophil / Lymphocyte Ratio (NLR), while the effectiveness of glucocorticoids slightly increases with increasing NLR. NLR is a marker of inflammation and severe Covid-19; it is thus expected that anti-coagulants (which target a distinct biomedical pathway) and NSAIDs (which are weaker) would not be as effective for patients with elevated NLR. In contrast, glucocorticoids become more effective for patients with more inflammation. This example shows the utility of a *differentiable* nonlinear additive model such as NAMs.

4.2 Multitask Learning

One advantage of NAMs is that they are easily extended to multitask learning (MTL) [4], whereas MTL is not available in EBMs or in any major boosted-tree package. In NAMs, the composability of neural nets makes it easy to train multiple subnets per feature. The model can learn task-specific weights over these subnets to allow sharing of subnets (shape functions) across tasks while also allowing subnets to differentiate between tasks as needed. How to implement MTL in EBMs is unclear and possibly involves changes to both the backfitting procedure and the information gain rule in decision trees. Figure 8 shows a multitask NAM architecture that can jointly learn different feature representations for each task while preserving the intelligibility and modularity of NAMs. As we show, this can benefit both accuracy and interpretability. We first demonstrate multitask NAMs on a synthetic dataset before showing their utility on a multitask formulation of the COMPAS recidivism prediction dataset.

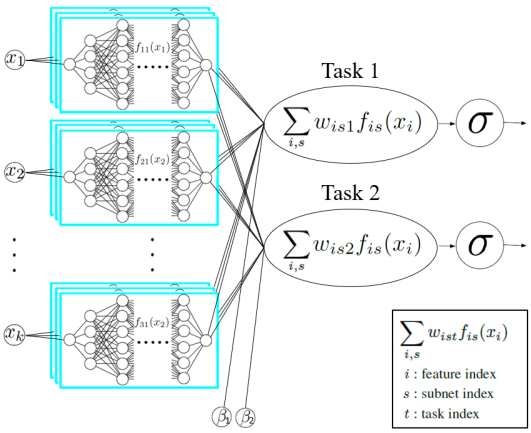


Figure 8: Multitask NAM architecture for binary classification. Multiple subnets are trained on each input feature and weighted sums are learned over the subnets.

Multitask NAM Architecture. The multitask architecture is identical to that of single task NAMs except that each feature is associated with multiple subnets and the model jointly learns a task-specific weighted sum over their outputs that determines the shape function for each feature and task. The outputs corresponding to each task are summed and a bias is added to obtain the final prediction score. The number of subnets does not need to be the same as the number of tasks — the number of subnets can be less than, equal to, or even more than the number of tasks.

4.2.1 Experiments on Synthetic Multitask Data

Multitask models often show improvement over single task learning when tasks are similar to each other and training data is limited. We construct a synthetic dataset that showcases the benefit of multitask learning in NAMs and demonstrates their ability to learn task-specific shape plots when

needed. We define 6 related tasks, each a function of three variables. All 6 tasks are the same function of variables x_0 and x_1, and differ only in the function applied to x_2:

$$Task_0 = f(x_0) + g(x_1) + h(x_2) \qquad Task_1 = f(x_0) + g(x_1) + i(x_2)$$
$$Task_2 = f(x_0) + g(x_1) - h(x_2) \qquad Task_3 = f(x_0) + g(x_1) - i(x_2)$$
$$Task_4 = f(x_0) + g(x_1) + (h(x_2) + i(x_2)) \qquad Task_5 = f(x_0) + g(x_1) - (h(x_2) + i(x_2))$$

Functions $f(x_0)$, $g(x_1)$, $h(x_2)$ and $i(x_2)$ are as follows:

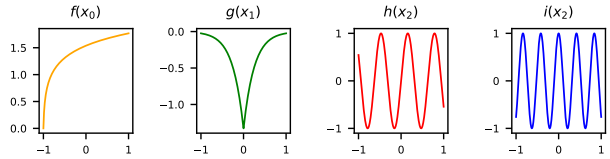


Table 2: MSE for STL and MTL NAMs on synthetic data. Average of 20 runs. Lower MSEs are better.

Model	Task 0	Task 1	Task 2	Task 3	Task 4	Task 5	Mean
Single Task NAM	0.965	1.116	1.347	0.944	1.058	1.066	1.083
Multitask NAM	0.710	0.715	0.709	0.711	0.717	0.709	0.712

A NAM with two subnets per feature can model every function of x_2 by learning two subnets, one for $h(x_2)$ and one for $i(x_2)$ and assigning appropriate weights to the output of each. Because we would not know this in advance with real data, we use 6 subnets so that each of the 6 tasks (outputs) could, if needed, learn independent shape functions. We train models on 2,500 training examples, evaluate them on a test set of 10,000 examples, and average the results over 20 trials.

Table 2 shows that on average across all tasks, multitask NAMs achieve mean squared error 34% lower than single task NAMs, and at least 25% lower on each individual task. In all 120 trials of the 6 tasks combined, MTL achieved a better score than STL on 119 of the 120 trials. Figure 9 shows the shape plots learned by median runs of STL and MTL for the functions of x_2 that vary among tasks.

4.2.2 Single and Multitask COMPAS Recidivism Prediction

COMPAS is a proprietary score developed to predict recidivism risk, which is used to inform bail, sentencing and parole decisions and has been the subject of scrutiny for racial bias [1, 8, 42]. In 2016, ProPublica released recidivism data [30] on defendants in Broward County, Florida.

Single Task Recidivism Prediction: First, we ask whether this dataset is biased using the transparency of single-task NAMs. Figure 10 shows the learned single-task NAM which is as accurate as black-box models on this dataset (see Table 1). The shape function for race indicates that the learned NAM may be racially biased: Black defendants are predicted to be higher risk for reoffending than white or Asian defendants. This suggests that the recidivism data may be racially-biased. The modularity of NAMs makes it easy to correct this bias by simply removing the contributions learned from the race attribute by zeroing out its mean-centered graph in the learned NAM. Although this would drop the AUC score as we are removing a discriminative feature, it may be a more fair model to use for making bail decisions. It is important to keep potentially offending attributes in the model

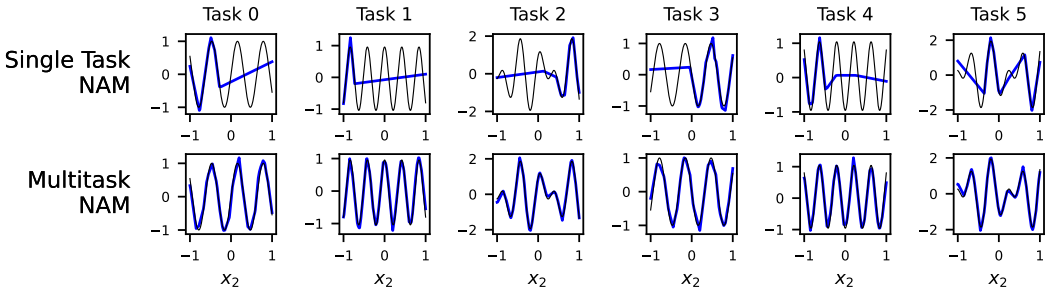


Figure 9: **Single and Multitask NAM shape plots** for x_2 from a typical (median) run of each task. The learned shape function is blue; the generator function is black. See A.9.1 for details of the generator functions.

Table 3: ROC AUC for multitask and single task NAMs on COMPAS dataset, broken down by gender. Each cell contains the mean AUC $\pm$ one standard deviation obtained via 5-fold cross validation. Higher AUCs are better.

Model	COMPAS Women	COMPAS Men	COMPAS Combined
Single Task NAM	0.716 ± 0.026	0.735 ± 0.009	0.737 ± 0.010
Multitask NAM	0.723 ± 0.019	0.737 ± 0.009	0.739 ± 0.010

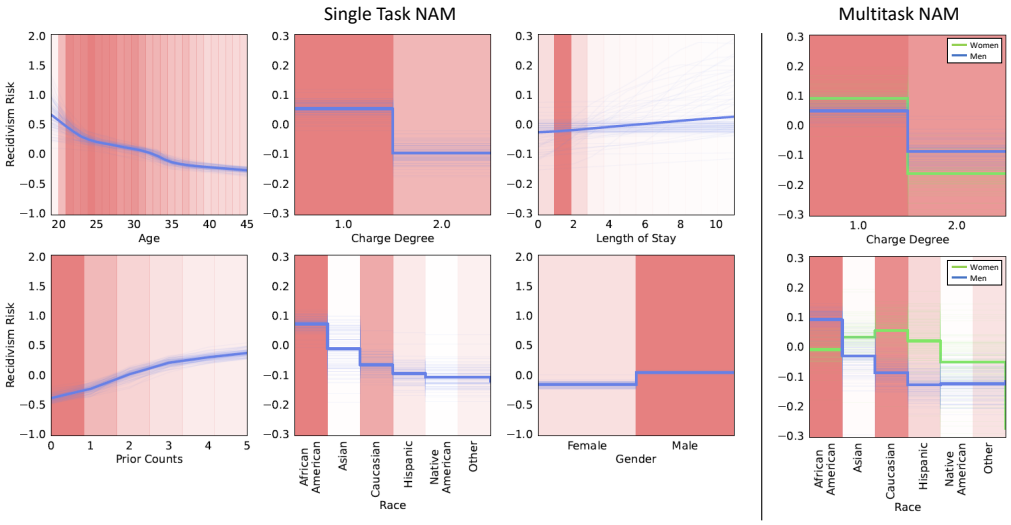


Figure 10: **Single and Multitask COMPAS Recidivism Prediction**. Plots in the left column show the shape functions for each input feature learned by an ensemble of 100 single task NAMs. Thin blue lines represent shape functions for individual members of the ensemble. Pink bars represent the normalized data density for each feature. Plots in the right column show the Race and Charge degree shape plots for an ensemble of 100 multitask NAMS, with the Women task shown in green, and the Men task in blue.

during training so that the bias can be detected and then removed after training. If the offending variables are eliminated before training, it makes debiasing the model more difficult: if the offending attributes are correlated with other training attributes, the bias is likely to spread to those attributes [3]. The transparency and modularity of NAMs allows one to detect unanticipated biases in data and makes it easier to correct the bias in the learned model.

Multitask Recidivism Prediction: In some settings multitask learning can increase accuracy and intelligibility by learning task-specific shape plots that expose task-specific patterns in the data that would not be learned by single task learning. We reformulate COMPAS as a multitask problem where recidivism prediction for men and women are treated as separate tasks on a NAM with two outputs. Indeed, we find that a multitask NAM reveals different relationships between race, charge degree, and recidivism risk for men and women while achieving slightly higher overall accuracy.

The right column of Figure 10 displays a selection of shape plots learned for a multitask NAM trained on the same data as the single task NAM but with Male and Female as separate output tasks. (The remaining MTL shape plots are similar for the two genders, reinforcing that these are strongly related tasks, but we omit them for brevity.) The race shape plot in the multitask NAM shows a different pattern of racial bias for each gender. The curve for men looks similar to that of the single task NAM (which is expected because men make up 81% of the data), but the curve for women suggests that recidivism risk is lower for Black women and higher for Caucasian and Hispanic women than for their male counterparts. The multitask shape plots also reveal that charge degree is almost twice as important for women as it is for men. The straightforward extension of NAMs to the multitask setting offers a useful modelling technique not currently available with tree-based GAMs.

5 Related Work

Generalized Additive Neural Networks (GANNs) [29] are somewhat similar to the NAMs we propose here. Like NAMs, GANNs used a restriction in the neural net architecture to force it to learn additive functions of individual features. GANNs, however, predate deep learning and use a single hidden layer with typically only 1-5 hidden units. Furthermore, GANNs did not use backpropagation [36],

required human-in-the-loop evaluation and were not successful in training accurate or scalable GAMs with neural nets. See Section A.7 for a more detailed overview of GANNs.

In contrast, NAMs in this paper benefit from the advances in deep learning. They use a large number of hidden units and multiple hidden layers per input feature subnet to allow more complex, more accurate shape functions to be learned. Furthermore, NAMs use novel ExU hidden units to allow subnets to learn the more non-linear functions often required for accurate additive models, and then form an ensemble of these nets to provide uncertainty estimates, further improve accuracy and reduce the high-variance that can result from encouraging the model to learn highly non-linear functions.

Prior to NAMs, the state-of-the-art in high-accuracy, interpretable generalized additive models [12, 14] are the GAM [23] and $\mathrm{GA}^2\mathrm{M}$ [24] based on regularized boosted decision trees which were successfully applied to healthcare datasets [5]. We compare the accuracy of NAMs to these models in Section 3. We note that pairwise interactions, similar to $\mathrm{GA}^2\mathrm{M}$s, can be easily added to NAMs – $\mathrm{GA}^2\mathrm{M}$s use a heuristic to compute the importance of each pairwise interaction by fitting residual from first-order terms and select the k (≤ 10) most important interactions. We don't consider such interactions to keep the paper focused on additive modeling with neural nets.

6 Conclusion and Broader Impact

We present Neural Additive Models (NAMs), which combine the inherent interpretability of GAMs with the expressivity of DNNs, opening the door for other advances in interpretability in deep learning. NAMs are competitive in accuracy to GAMs while being more easily extendable than existing GAMs due to their differentiability and composability. Overall, NAMs are likely to broaden the use of inherently interpretable models in the deep learning community.

Interpretability in AI systems might be desirable or necessary for various reasons – see [43] for an overview; we discuss some of them in the context of NAMs below:

- **Safeguarding against bias**: NAMs can check whether training data is used in ways that result in bias or discriminatory outcomes and can be easily corrected for bias to yield possibly more fair models – *e.g.,* Section 4.2.2 demonstrates this utility for recidivism risk prediction.
- **Improving AI system design**: NAMs allow developers to interrogate why it behaved in a certain way (*e.g.,* tracking system malfunctions), and develop improvements – Section A.1 shows that NAMs can explain seemingly anomalous results in healthcare as well as uncover problems that might put some kinds of patients at risk and need correction before deploying the system.
- **Adhering to regulatory standards or policy requirements**: Interpretability of NAMs can be important in enforcing legal rights surrounding a system – *e.g.,* credit scores in the United States, have a well-established "right to explanation". NAMs can also enable individuals to contest model outputs, *e.g.,* challenging an unsuccessful loan application, based on the interpretations provided by NAMs for a specific decision (Figure 5).
- **Assessing risk, robustness, and vulnerability**: This can be particularly important if an AI system is deployed in a new environment, where we cannot be sure of its effectiveness – *e.g.,* NAMs for fraud detection (Section A.2.2) can be analyzed to understand the risks involved or how it might fail before deploying it to unseen customers.
- **Giving users confidence in the system**: Interpretations from NAMs might provide users confidence that it works as intended – *e.g.,* expensive house prices near metropolitan areas such as San Francisco, as predicted by NAMs (Figure 6), is expected for a trustworthy model.
- **Data-driven scientific discovery**: NAMs can be applied in natural sciences – *e.g.,* ecology [12], medicine [13], astronomy [15] *etc.* – to obtain novel scientific insights and discoveries from observational or simulated data [34, 44] while remaining scalable to the ever-increasing data.

There are also pitfalls associated with interpretability methods – NAMs are no exception. Different contexts give rise to different interpretability needs – *e.g.,* public have different expectations of systems used in healthcare *vs.* recruitment [19]. Furthermore, AI system designs often need to balance competing demands – *e.g.,* to optimize the accuracy of a system or ensure fairness (NAMs for making bail decisions with race feature "removed" may be less accurate but more fair). In many critical decision-making areas – *e.g.,* healthcare, justice, and public services – complex processes have developed over time to provide safeguards, audit functions, or other forms of accountability. NAMs may therefore be only the first step in creating trustworthy systems. Those developing NAMs must consider how their use fits in the wider socio-technical context of its deployment

References

[1] Julia Angwin, Jeff Larson, Lauren Kirchner, and Surya Mattu. Machine Bias: There's software used across the country to predict future criminals. And it's biased against blacks, 2016. URL https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing. [Accessed February 1, 2020].

[2] Devansh Arpit, Stanislaw Jastrzkebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. *ICML*, 2017.

[3] Richard Berk, Hoda Heidari, Shahin Jabbari, Michael Kearns, and Aaron Roth. Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research*, 2018.

[4] Rich Caruana. Multitask learning. *Machine Learning*, 1997.

[5] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. *SIGKDD*, 2015.

[6] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. *SIGKDD*, 2016.

[7] Andrea Dal Pozzolo. Adaptive machine learning for credit card fraud detection. *PhD Thesis, Department of Computer Science, Université Libre de Bruxelles*, 2015.

[8] Julia Dressel and Hany Farid. The accuracy, fairness, and limits of predicting recidivism. *Science advances*, 2018.

[9] FICO. FICO Explainable Machine Learning Challenge. https://community.fico.com/s/explainable-machine-learning-challenge, 2018.

[10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *AISTATS*, 2010.

[11] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D Sculley. Google vizier: A service for black-box optimization. *SIGKDD*, 2017.

[12] Antoine Guisan, Thomas C Edwards Jr, and Trevor Hastie. Generalized linear and generalized additive models in studies of species distributions: setting the scene. *Ecological modelling*, 2002.

[13] T Hastie and R Tibshirani. Generalized additive models for medical research. *Statistical methods in medical research*, 1995.

[14] Trevor Hastie and Robert Tibshirani. *Generalized Additive Models*. Chapman and Hall/CRC, 1990.

[15] M W Hattab, R S de Souza, B Ciardi, J-P Paardekooper, S Khochfar, and C Dalla Vecchia. A case study of hurdle and generalized additive models in astronomy: the escape of ionizing radiation. *Monthly Notices of the Royal Astronomical Society*, 2018.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CVPR*, 2015.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2016.

[18] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. Multilayer feedforward networks are universal approximators. *Neural Networks*, 1989.

[19] ICO. Project explain: Interim report. 2019. https://ico.org.uk/media/2615039/project-explain-20190603.pdf.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Alex Krizhevsky. Convolutional deep belief networks on cifar-10. 2010.

[22] Christine K Lee, Muntaha Samad, Ira Hofer, Maxime Cannesson, and Pierre Baldi. Development and validation of an interpretable neural network for prediction of postoperative in-hospital mortality. *NPJ digital medicine*, 2021.

[23] Yin Lou, Rich Caruana, and Johannes Gehrke. Intelligible models for classification and regression. *SIGKDD*, 2012.

[24] Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. *SIGKDD*, 2013.

[25] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. *ICML*, 2010.

[26] Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. Interpretml: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*, 2019.

[27] R Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 1997.

[28] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *JMLR*, 2011.

[29] William JE Potts. Generalized additive neural networks. *SIGKDD*, 1999.

[30] ProPublica. COMPAS Data and analysis for 'Machine Bias'. `https://github.com/propublica/compas-analysis`, 2016.

[31] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2018.

[32] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. *ICML*, 2018.

[33] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. *SIGKDD*, 2016.

[34] Ribana Roscher, Bastian Bohn, Marco F Duarte, and Jochen Garcke. Explainable machine learning for scientific insights and discoveries. *arXiv preprint arXiv:1905.08883*, 2019.

[35] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 2019.

[36] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 1986.

[37] Mohammed Saeed, Mauricio Villarroel, Andrew T Reisner, Gari Clifford, Li-Wei Lehman, George Moody, Thomas Heldt, Tin H Kyaw, Benjamin Moody, and Roger G Mark. Multiparameter intelligent monitoring in intensive care ii (mimic-ii): a public-access intensive care unit database. *Critical care medicine*, 2011.

[38] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 2016.

[39] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *NeurIPS*, 2012.

[40] Divyanshi Srivastava, Begüm Aydin, Esteban O Mazzoni, and Shaun Mahony. An interpretable bimodal neural network characterizes the sequence and preexisting chromatin predictors of induced transcription factor binding. *Genome Biology*, 2021.

[41] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014.

[42] Sarah Tan, Rich Caruana, Giles Hooker, and Yin Lou. Distill-and-compare: Auditing black-box models using transparent model distillation. *AAAI/ACM Conference on AI, Ethics, and Society*, 2018.

[43] The Royal Society. Explainable AI: The Basics - Policy Briefing. 2019. `https://royalsociety.org/-/media/policy/projects/explainable-ai/AI-and-interpretability-policy-briefing.pdf`.

[44] The Royal Society and The Alan Turing Institute. The AI revolution in scientific research. 2019. `https://royalsociety.org/-/media/policy/projects/ai-and-society/AI-revolution-in-science.pdf`.

[45] Lev V Utkin, Egor D Satyukov, and Andrei V Konstantinov. Survnam: The machine learning survival model explanation. *arXiv preprint arXiv:2104.08903*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Supplementary Material for Neural Additive Models

A.1 NAMs on MIMIC-II: Mortality Prediction in ICUs

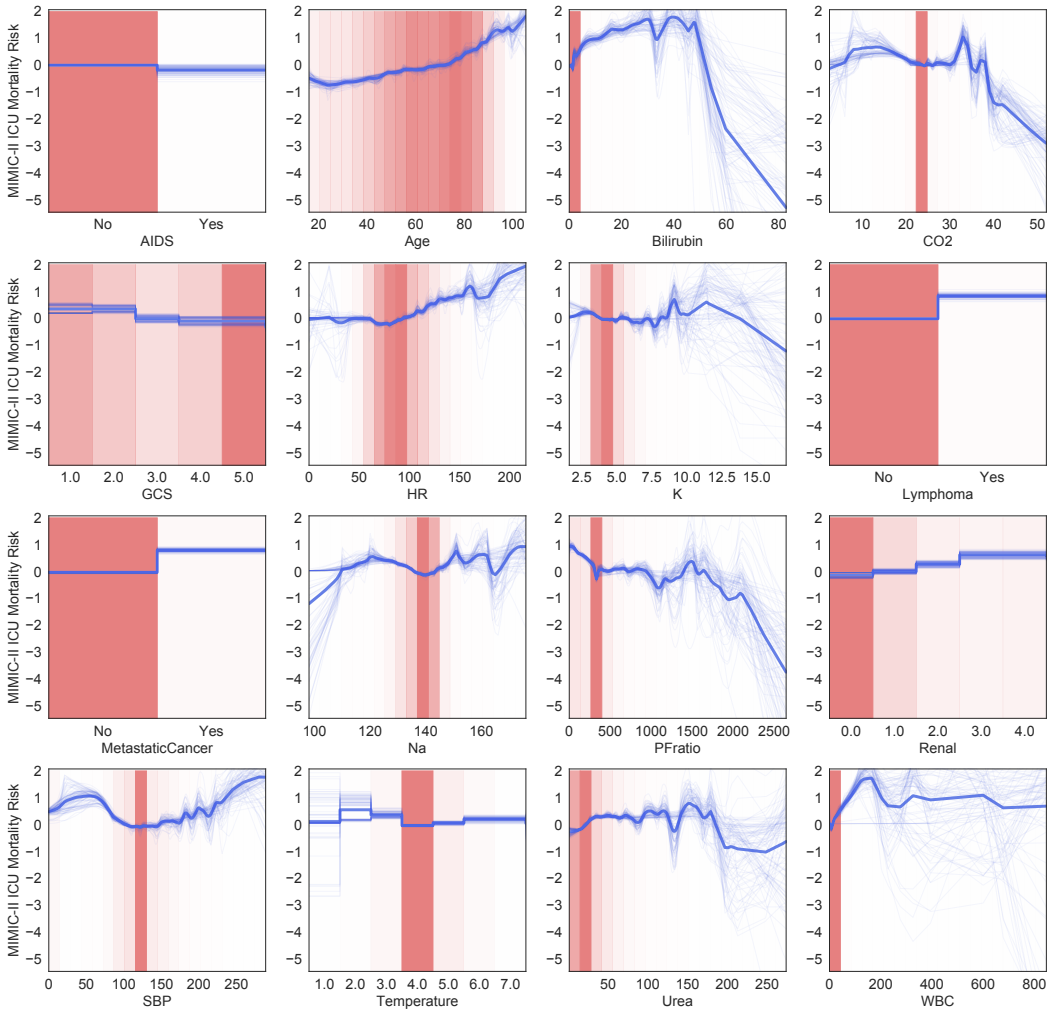


Figure A.1: **MIMIC-II ICU Mortality**. NAM shape functions learned on the MIMIC-II dataset to predict mortality risk using medical features (shown on the x-axis) collected during the stay in the ICU. Low values on the y-axis indicates a low risk of mortality.

Figure A.1 shows 16 of the shape functions learned by the NAM for the MIMIC-II dataset [37] to predict mortality in intensive care unit (ICUs). (The 17th graph for Admission Type is flat and we omit it to save space.) The plot for HIV/AIDS shows that patients with AIDS have less risk of ICU mortality. While this might seem counter-intuitive, we confirmed with doctors that this is probably correct: among the various reasons why one might be admitted to the ICU, AIDS is a relatively treatable illness and is one of the less risky reasons for ICU admission. In other words, being admitted to the ICU for AIDS suggests that the patient was not admitted for another riskier condition, and thus the model correctly predicts that the patient is at less risk than other non-AIDS patients.

The shape plot for Age shows, as expected, that mortality risk tends to increase with age, with the most rapid rise in risk happening above age 80. There is detail in the graph that is interesting and warrants further study such as the small increase in risk at ages 18 and 19, and the bump in risk at age 90 — jumps in risk that happen at round numbers are often due to social effects.

The shape plot for Bilirubin (a by product of the breakdown of red blood cells) shows that risk is low for normal levels below 2-3, and rises significantly for levels above 15-20, until risk drops again above 50. There is also a surprising drop in risk near 35 that requires further investigation. We believe

the drop in risk above 50 is because patients above 50 begin to receive dialysis and other critical care and these treatments are very effective. The drop in risk that occurs for Urea above 175 is also likely due to dialysis.

The plot for the Glasgow Coma Index (GCS) is monotone decreasing as would be expected: higher GCS indicates less severe coma. Note that NAMs are not biased to learn monotone functions such as this and the shape of the plot is driven by the data. The NAM also learns a monotone increasing shape plot for risk as a function of renal function. This, too, is as expected: 0.0 codes for normal renal function and 4.0 indicates severe renal failure.

The NAM has learned that risk is least for normal heart rate (HR) in the range 60-80, and that risk rises as heart rate climbs above 100. Also, as expected, both Lymphoma and Metastatic Cancer increase mortality risk. The CO2 graph shows low risk for the normal range 22-24. There is an interesting drop in risk at CO2 equal to 37 (the dip between the peaks at 33 and 39) that warrants further investigation.

The shape plot for PFratio (a measure of the effectiveness of converting O2 in air to O2 in blood) shows a drop at PFratio = 332 which upon further inspection is due to missing values in PFratio being imputed with the mean: because most patients do not have their PFratio measured, the highest density of patients are actually missing their PFratio which was then imputed with the mean value of 332. One way to detect that imputed missing values are responsible for a dip (or rise) in a shape plot is when risk at the mean value of the attribute suddenly drops (or rises) to a risk level similar to what the model learns for patients who are considered normal/healthy in this dimension: the jump happens at the mean when imputation is done with the mean value, and the level jumps towards the risk of normal healthy patients because often the variable was not measured because the patients were considered normal (for this attribute), a medical assessment which is often correct.

Normal temperature is coded as 4.0 on the temperature plot, and risk rises for abnormal temperature above and below this value. It's not clear if the non-monotone risk for hypothermic patients with temperatures 1 or 2 is due to variance, an unknown problem with the data, or an unexplained but real effect in the training signals and warrants further investigation. Similarly, the shape plot for Systolic Blood Pressure (SBP) shows lowest risk for normal SBP near 120, with risk rising for abnormally elevated or depressed SBP. The jumps in risk that happen at 175, 200, and 225 are probably due to treatments that doctors start to apply at these levels: the risk rises to left of these thresholds as SBP rises to more dangerous levels but before the treatment threshold is reached, and then drops a little to the right of these thresholds when most patients above the treatment threshold are receiving a more aggressive treatment that is effective at lowering their risk.

Discussion. In summary, most of what the NAM has learned appears to be consistent with medical knowledge, though a few details on some of the graphs (e.g., the increase in risk for young patients, and the drop in risk for patients with Bilirubin near 35) require further investigation. NAMs are attractive models because they often are very accurate, while remaining interpretable, and if a detail in some graph is found to be incorrect, the model can be edited by re-drawing the graph. However, NAMs (like all GAMs), are not causal models. Although the shape plots can be informative, and can help uncover problems with the data that might need correction before deploying the models, the plots do not tell us *why* the model learned what it did, or what the impact of intervention (*e.g.,* actively lowering a patient's fever or blood pressure) would be. The shape plots do, however, tell us *exactly* how the model makes its predictions.

A.2 Intelligibility of NAMs on other datasets

A.2.1 FICO Score: Understanding Individual Predictions on Credit Scores

The FICO score is a widely used proprietary credit score to determine credit worthiness for loans in the United States. The FICO dataset [9] is comprised of real-world anonymized credit applications made by customers and their assigned FICO Score, based on their credit report information. We visualize the feature contributions of a NAM trained using the FICO dataset (see Figure A.4 in appendix) for two applicants (Table A.4) with low and high scores respectively.

Figure 5 shows that the most important features for the high scoring applicant are (1) Average Months on File and (2) Net Fraction Revolving Burden (*i.e.,* percentage of credit limit used) which take the value 235 months and 0% respectively. This makes sense, as generally, the longer a person's credit history, the better it is for their credit score. Although there is a strong inverse correlation between

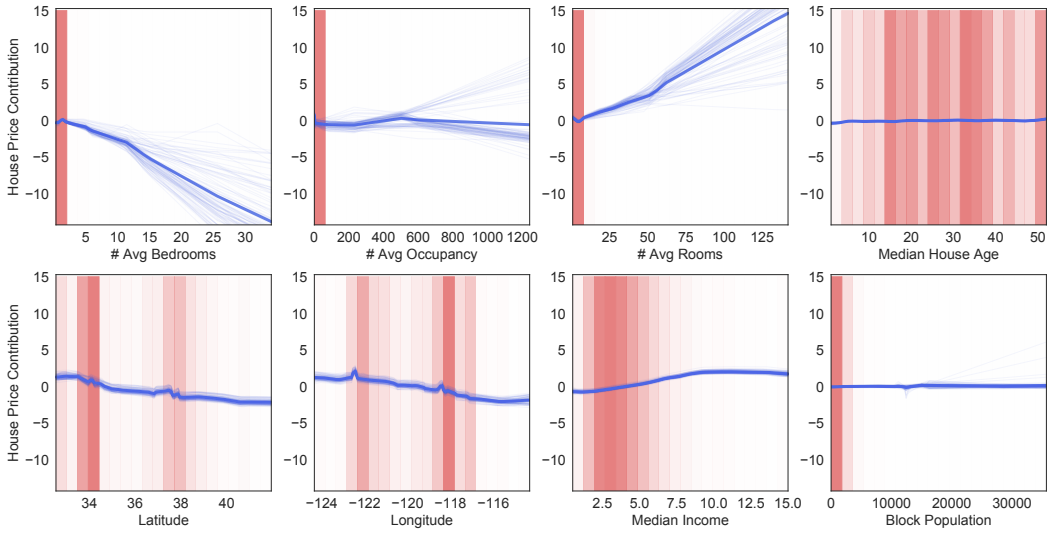


Figure A.2: **California Housing**. Graphs learned by NAMs trained to predict house prices (regression) on the California Housing dataset. These plots show the individual shape functions learned by an ensemble of hundred NAMs for each input feature as well as the data density. The thin blue lines represents different shape functions from the ensemble to show the agreement of the members of the ensemble. The pink bars represent the normalized data density for each feature. The darker the bar the more data there is with that value.

Net Fraction Revolving Burden and the score, it is positively correlated for small values (< 10). This means that making use of some credit increases your credit score, but using too much of it is bad. We are confident in this interpretation because most of the data density is in small values, and each NAM in the ensemble displays a similar shape function (Figure A.4). For the low scoring applicant, the main factors are (1) Total Number of Trades[4] and (2) Net Fraction Installment Burden (Installment balance divided by original loan amount) which take the values 57 and 68% respectively. This applicant used their credit quite frequently and has a large burden, thus resulting in a low score.

A.2.2 Credit Fraud: Financial Fraud Detection [Classification]

This is a large dataset [7] containing 284,807 transactions made by European credit cardholders where the task is to predict whether a given transaction is fraudulent or not. It is highly unbalanced and contains only 492 frauds (0.172% of the entire dataset) of all transactions. Table 1 shows that on this dataset, NAMs outperform EBMs and perform comparably to the XGBoost baseline. This shows the benefit of using NAMs instead of tree-based GAMs and suggests that NAMs can provide highly accurate and intelligible models on large datasets. NAMs using ExU units perform much better compared to NAMs with standard DNNs (AUC ≈ 0.974).

A.2.3 California Housing: Predicting Housing Prices [Regression]

California Housing dataset [27] is a canonical machine learning dataset where the task is to predict the median price of houses (in million dollars) in each California district. The learned NAM considers the median income as well as the house location (latitude, longitude) as the most important features (we omit the other six graphs to save space, see Figure A.2). As shown by Figure 6, the house prices increase linearly with median income in high data density regions. Furthermore, the graph for longitude shows sharp jumps in price prediction around 122.5°W and 118.5°W which roughly correspond to San Francisco and Los Angeles respectively.

A.3 Fitting Jagged Curves

A.4 Regularization and Training

ExU units encourage learning highly jagged curves, however, most realistic shape functions tend to be smooth with large jumps at only a few points. To avoid overfitting, we use the following regularization techniques:

[4]Credit trades refer to any agreement between a lending agency and consumers.

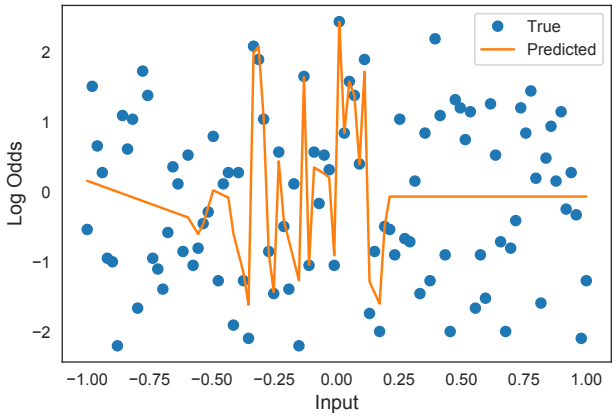


Figure A.3: **Toy classification**: Deep neural network with 3 hidden layers of size 64, 64 and 32 respectively with ReLU activation and Xavier initialization trained for 10,000 epochs on toy classification dataset described in Section 2.1. We use a batch size of 1024 with the Adam optimizer and a learning rate decay of 0.995 every epoch. The learning rate was tuned in [1e-3, 1e-1) and we show the results with the best learning rate.

- **Dropout** [41]: It regularizes ExUs in each feature net, allowing them to learn smooth functions while being able to represent jumps (Figure 3).
- **Weight decay**: This is done by penalizing the L2 norm of weights in each feature net.
- **Output Penalty**: We penalize the L2 norm of the prediction of each feature net, so that its contribution stays close to zero unless evident otherwise from the data.
- **Feature Dropout**: We also drop out individual feature networks during training. When there are correlated input features, an additive model can possibly learn multiple explanations by shifting contributions across these features. This term encourages NAMs to spread out those contributions.

Training. Let $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ be a training dataset of size N, where each input $\mathbf{x} = (x_1,\ x_2,\ \ldots,\ x_K)$ contains K features and y is the target variable. In this work, we train NAMs using the loss $\mathcal{L}(\theta)$ given by

$$\mathcal{L}(\theta) = \mathbb{E}_{x,y\sim\mathcal{D}}\big[l(x, y; \theta) + \lambda_1 \eta(x; \theta)\big] + \lambda_2 \gamma(\theta) \tag{3}$$

where $\eta(x; \theta) = \frac{1}{K}\sum_x \sum_k (f_k^\theta(x_k))^2$ is the output penalty, $\gamma(\theta)$ is the weight decay and f_k^θ is the feature network for the kth feature. Each individual network is also regularized using feature dropout and dropout with coefficients λ_3 and λ_4 respectively. $l(x, y; \theta)$ is the task dependent loss function. We use the cross-entropy loss for binary classification:

$$l(x, y; \theta) = -y \log(p_\theta(x)) - (1 - y) \log(1 - p_\theta(x)),$$

where $p_\theta(x) = \sigma\big(\beta^\theta + \sum_{k=1}^{K} f_k^\theta(x_k)\big)$ and mean squared error (MSE) for regression:

$$l(x, y; \theta) = \big(\beta^\theta + \sum_{k=1}^{K} f_k^\theta(x_k) - y\big)^2$$

A.5 Experimental Details

Training Details. The NAM feature networks (f_k^θ) are trained jointly using the Adam optimizer [20] with a batch size of 1024 for a maximum of 1000 epochs with early stopping using the validation dataset. The learning rate is annealed by a factor of 0.995 every training epoch. For all the tasks, we tune the learning rate, output penalty coefficient (λ_1), weight decay coefficient (λ_2), dropout rate (λ_3) and feature dropout rate (λ_4). For computational efficiency, we tune these hyperparameters using Bayesian optimization [11, 39] based on cross-validation performance with a single train-validation split for each fold. We used TESLA P100 GPUs for all experiments involving neural networks while CPU machines for *sklearn* or XGBoost baselines.

Evaluation. We perform 5-fold cross validation to evaluate the accuracy of the learned models. To measure performance, we use area under the precision-recall curve (AUC) for binary classification (as

the datasets are unbalanced) and root mean-squared error (RMSE) for regression. For NAMs and DNNs, one of the 5 folds (20% data) is used as a held-out test set while the remaining 4 folds are used for training (70% data) and validation (10% data). The training and validation splits are randomly subsampled from the 4 folds and this process is repeated 20 times. For each run, the validation set is used for early stopping. For each fold, we ensemble the NAMs and DNNs trained on the 20 and EBMs on 100 train-validation splits respectively to make the prediction on the held-out test set.

A.6 Hyperparameters

We use a batch size of 1024 with the Adam optimizer and a learning rate decay of 0.995 every epoch in our experiments for NAMs and DNNs.

Linear Models/ Decision Trees. We use the *sklearn* implementation [28], and tune the hyperparameters with grid search.

EBMs. We use the open-source implementation [26] with the parameters specified by prior work [5] for a fair comparison.

NAMs. We tune the dropout coefficient (λ_3) in the discrete set {0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}, weight decay coefficient (λ_2) in the continuous interval $[0.000001, 0.0001)$, learning rate in the interval $[0.001, 0.1)$, feature dropout coefficient (λ_4) in the discrete set {0, 0.05, 0.1, 0.2} and output penalty coefficient (λ_1) in the interval $[0.001, 0.1)$. Note that the weight decay is implemented as the average weight decay over the individual feature networks in NAMs. Refer to Table A.2 and Table A.1 for hyperparameters found on regression and classification datasets.

DNNs. We train DNNs with 10 hidden layers containing 100 units each with ReLU activation using the Adam optimizer. This architecture choice ensures that this network had the capacity to achieve perfect training accuracy on datasets used in our experiments. We use weight decay and dropout to prevent overfitting and tune hyperparameters using a similar protocol as NAMs. We tune the dropout coefficient (λ_3) in {0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5}, weight decay coefficient (λ_2) in the continuous interval $[0.0000001, 0.1)$ and learning rate in the interval $[0.001, 0.1)$.

Table A.1: Optimal hyperparameters found for NAMs on regression datasets. "Hidden units" shows the number of hidden layers as well as the number of neurons used in each layer for each feature network.

Hyperparameter	FICO	Housing
Learning Rate	0.0161	0.00674
Output Penalty (λ_1)	0.0205	0.001
Weight Decay (λ_2)	1.07 x 10^{-5}	10^{-6}
Dropout	0.0	0.0
Feature Dropout	0.0	0.0
Num units	64, 64, 32	64, 64, 32
Activation	ReLU	ReLU
Hidden unit	Standard	Standard

Table A.2: Optimal hyperparameters found for NAMs on classification datasets. "Hidden units" shows the number of hidden layers as well as the number of units used in each hidden layer for each feature network.

Hyperparameter	COMPAS	MIMIC-II	Credit Fraud
Learning Rate	0.02082	0.005	0.0157
Output Penalty	0.2078	0.3	0.0
Weight Decay	0.0	9.6 x 10^{-5}	4.95 x 10^{-6}
Dropout	0.1	0.2	0.8
Feature Dropout	0.05	0.0	0.0
Num units	64, 64, 32	1024	1024
Activation	ReLU	ReLU-1	ReLU-1
Hidden unit	Standard	ExU	ExU

A.7 GANNs

GANNs begin with subnets containing a single hidden unit for each input feature, and use a human-in-the-loop process to add (or subtract) hidden units to the architecture based on human evaluation of plotted partial residuals. This means that the training procedure cannot be automated. In practice, the laborious manual effort required to evaluate all of the partial residual plots to decide what to do next, and then retrain the model after adding or subtracting hidden units from the architecture meant that GANN nets remained very small and simple — typically only one hidden unit per feature.

A.8 Additional Graphs and Tables

Table A.3: **FICO Score**. Meaning of the different attributes of the feature "Max Delq/Public Records Last Year".

Value	Meaning
0	Derogatory comment
1	120+ days delinquent
2	90 days delinquent
3	60 days delinquent
4	30 days delinquent
5,6	Unknown delinquent
7	Current and never delinquent
8,9	All other

A.9 Multitask Learning

A.9.1 Synthetic Data Generation

We used the following generator functions to produce our synthetic dataset:

$$
\begin{aligned}
f(x_0) &= \frac{1}{3}\log 100x_0 + 101 \\
g(x_1) &= -\frac{4}{3}e^{-4|x_1|} \\
h(x_2) &= \sin(10x_2) \\
i(x_2) &= \cos(15x_2)
\end{aligned}
\tag{4}
$$

Noise sampled from $N(0, \frac{5}{6})$ was added to the target for each task. We will provide our generation code for others who are interested in using this dataset.

A.9.2 Shape Plots for All Synthetic Features

We include here shape plots for all features in the synthetic data for both single and multitask NAMs. The MTL results represent a single model trained on all 6 tasks. In each case, it models the shape functions for every feature and the target with high accuracy. By contrast, the STL for $Task0$, struggles to fit the data for x_2 and achieves low accuracy on the target in this regime of noise and training set size.

Table A.4: Feature attributes for the two individuals shown in Figure 5

Feature	High Score Applicant	Low Score applicant
Months Since Oldest Trade Open	417.0	174.0
Months Since Most Recent Trade	25.0	1.0
Average Months in File	235.0	66.0
# Satisfactory Trades	9.0	44.0
# Trades 60+ Ever	0.0	11.0
# Trades 90+ Ever	0.0	8.0
% Trades Never Delinquent	100.0	70.0
Months Since Most Recent Delinquency	0.0	3.0
Max Delq/Public Records Last Year	6.0	0.0
Max Delinquency Ever	6.0	0.0
# Total Trades	9.0	57.0
# Trades Open in Last 12 Months	0.0	5.0
% Installment Trades	22.0	66.0
Months Since Most Recent Inquiry excluding 7 days	0.0	0.0
# Inquiries in Last 6 Months	1.0	7.0
# Inquiries in Last 6 Months excluding 7 days	0.0	6.0
Net Fraction Revolving Burden	0.0	23.0
Net Fraction Installment Burden	0.0	68.0
# Revolving Trades with Balance	1.0	2.0
Number Installment Trades with Balance	0.0	5.0
# Bank/Natl Trades with high utilization ratio	0.0	0.0
% Trades with Balance	40.0	64.0
Delinquent	0.0	1.0
Inquiry	1.0	1.0

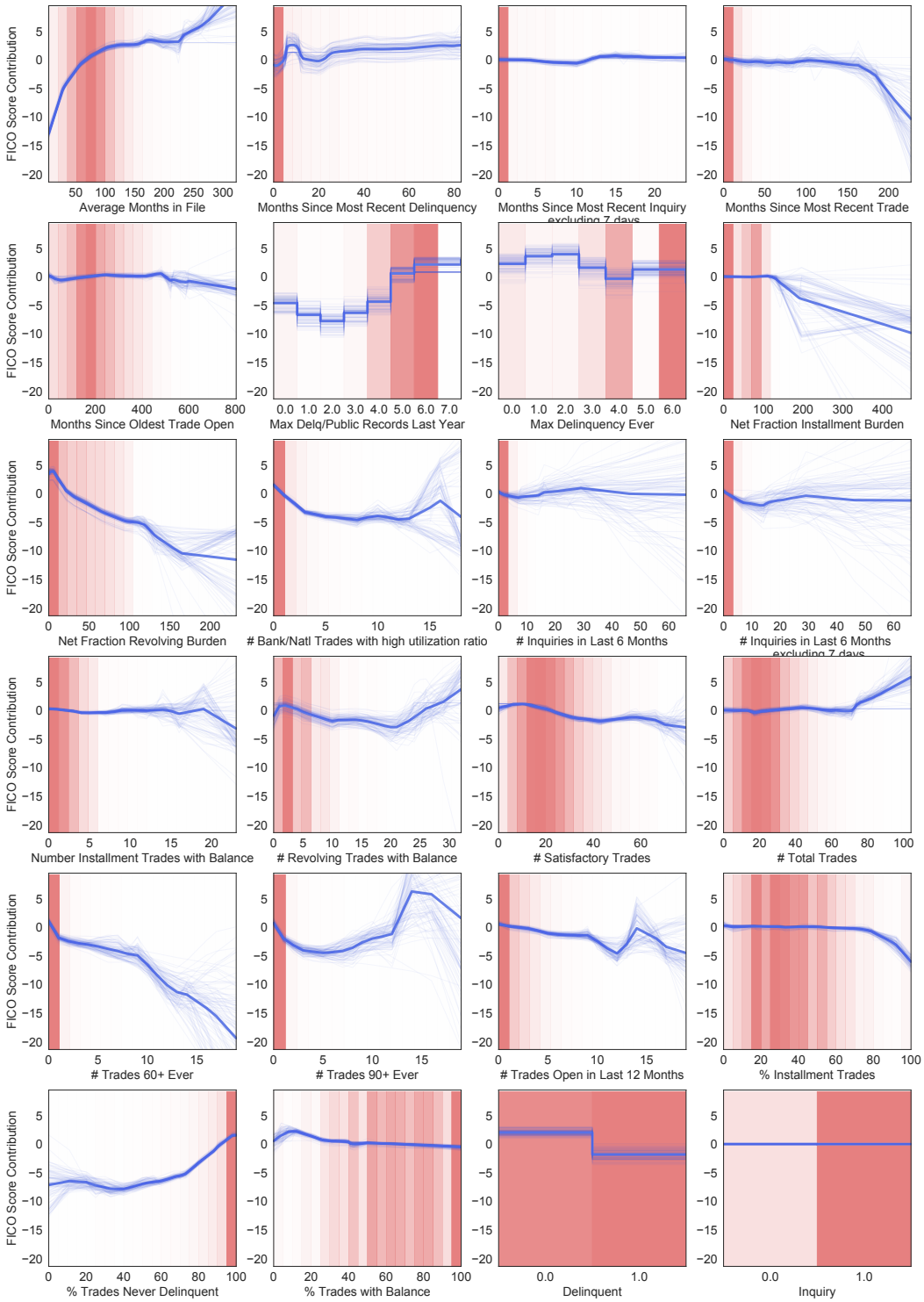


Figure A.4: **FICO Score Prediction**. Graphs learned by NAMs trained to predict FICO scores (regression) based on their credit report information. These graphs can be interpreted easily, *e.g.,* the second last graph in the bottom row shows that being delinquent on your payments decreases your credit score.

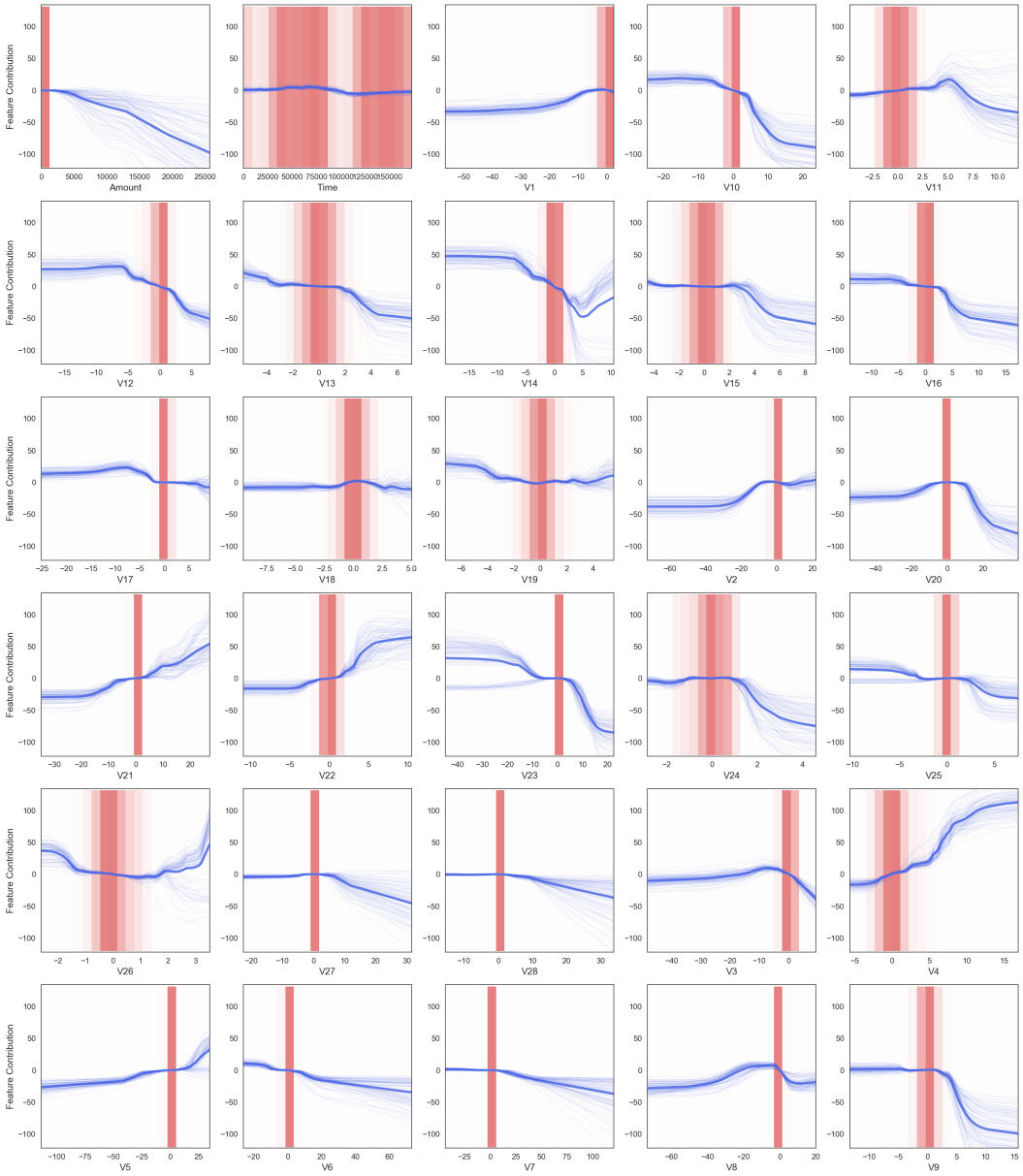


Figure A.5: **Credit Fraud Detection**: Graphs learned by NAMs with ExU units on this large classification dataset. The task is to predict credit fraud where the class variable takes value 1 in case of fraud and 0 otherwise using a large dataset of credit card transactions. The dataset only contains only numerical input variables which are the result of a PCA transformation except the features 'Time' and 'Amount'. Unfortunately, due to confidentiality issues, the original features are not provided in the dataset.

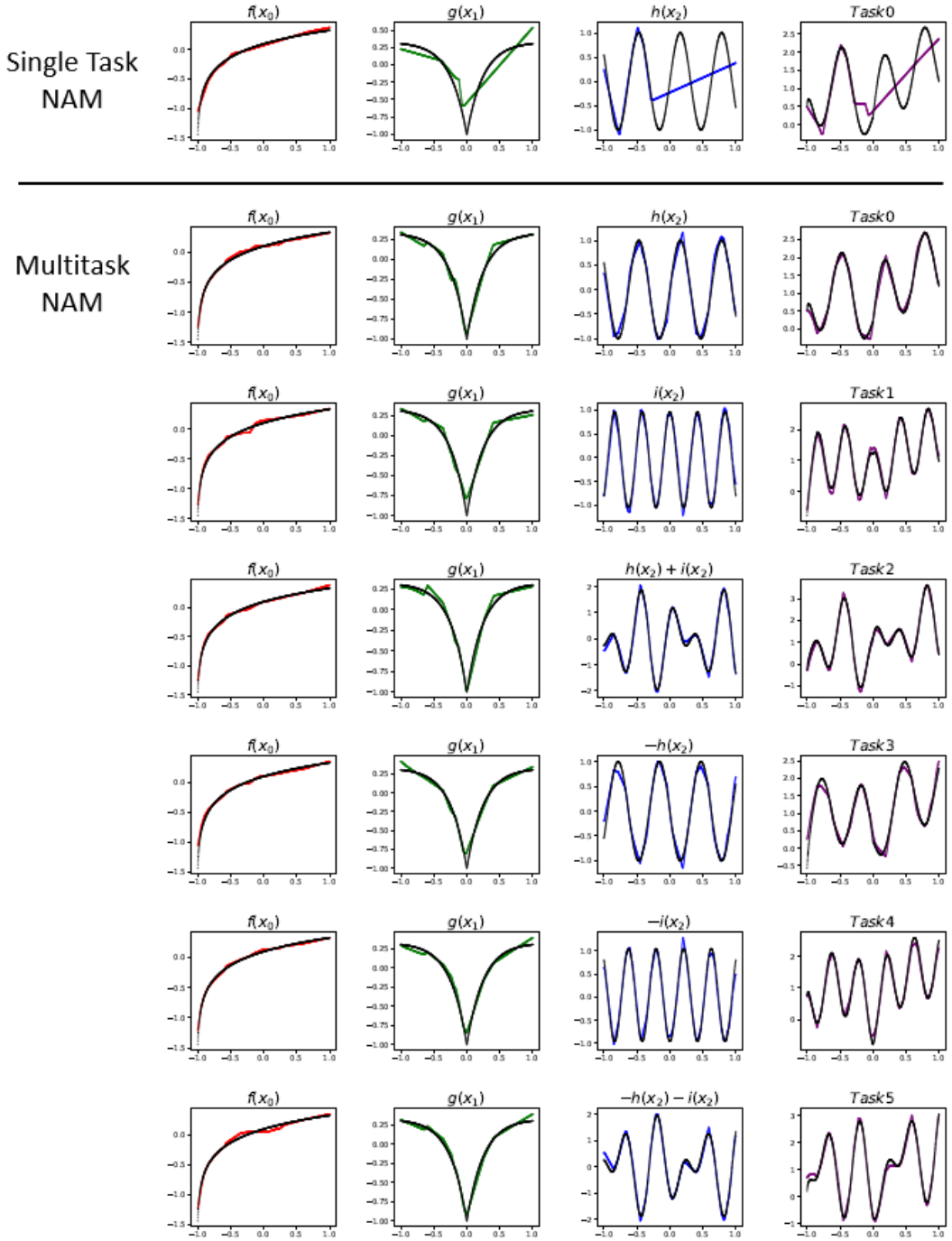


Figure A.6: **Single and Multitask NAMs trained on synthetic data**: Shape plots for all synthetic features for a typical (median) run of single and multitask NAMs. The colored lines represent learned shape functions for each feature and the black line represents the generator function.