Explainable Neural Networks with Guarantee: A Sparse Estimation Approach

Antoine Ledent and Peng Liu*

Singapore Management University
aledent@smu.edu.sg, liupeng@smu.edu.sg

Abstract

Balancing predictive power and interpretability has long been a challenging research area, particularly in powerful yet complex models like neural networks, where nonlinearity obstructs direct interpretation. This paper introduces a novel approach to constructing an explainable neural network that harmonizes predictiveness and explainability. Our model is designed as a linear combination of a sparse set of jointly learned features, each derived from a different trainable function applied to a single 1-dimensional input feature. Leveraging the ability to learn arbitrarily complex relationships, our neural network architecture enables automatic selection of a sparse set of important features, with the final prediction being a sum of rescaled versions of these features. We demonstrate the ability to select significant features while maintaining comparable predictive performance and direct interpretability through extensive experiments on synthetic and real-world datasets. We also provide theoretical analysis on the generalization bounds of our framework, which is favorably linear in the number of selected features and only logarithmic in the number of input features. We further lift any dependence of sample complexity on the number of parameters or the architectural details under very mild conditions. Our research paves the way for further research on sparse and explainable neural networks with guarantees.

Extended version — https://arxiv.org/pdf/2501.02010

1 Introduction

Neural networks have achieved state-of-the-art performance in multiple domains, from image and speech recognition (He et al. 2016; He, Liu, and Tao 2020; Lecun et al. 1998; Szegedy et al. 2015) to natural language processing (Devlin et al. 2018; Church 2017; TAN et al. 2022). However, their complex architectures and the vast number of parameters mean that they can only be understood as 'black box' models, where the decision-making process is not interpretable. Furthermore, neural networks typically involve millions of trainable parameters or even more, leading to poor understanding of their generalization behavior (Bartlett, Foster, and Telgarsky 2017). This lack of transparency and explainability is bound to raise concerns, especially in high-stakes domains like healthcare, finance, and autonomous driving, where understanding the reasoning behind predictions is paramount.

Whilst many works attempt to analyze the predictions made by neural networks to make them more interpretable (Zhou et al. 2018, 2014; Dhurandhar et al. 2018; Goyal et al. 2019), the explanations produced are still far from the aim of providing an easily computable decision function which humans can understand and manipulate. For instance, in the computer vision literature, most works focus on visualizing concepts learned by individual neurons in intermediary layers. This doesn't fully explain how such concepts are learned from the pixel data or how these concepts are aggregated to produce a final prediction. Similarly, in natural language processing, an interpretable method attempts to identify which words were given greater importance in generating the prediction; it is still difficult to fully explain how the model's prediction has utilized grammatical concepts or subtle clues such as irony.

Although many of the issues above are arguably tied to the intrinsic complexity of typical machine learning problems such as computer vision and natural language processing, there is a lack of trustworthy and interpretable models in other domains, such as healthcare and finance, where the data are sometimes much lower-dimensional and where each feature corresponds to a concrete concept. These domains often require identifying a small set of relevant features, from a possibly high-dimensional space, that contribute most significantly to the outcome of interest. Reducing the dimensionality of input data also helps mitigate the curse of dimensionality and reduces the risk of overfitting. However, many feature selection methods often involve heuristics or statistical measures on feature ranking that are decoupled from the model training process. Other methods, such as involve L^1-norm regularization (Roth 2004; Zou 2006; Tibshirani 2013), performs simultaneous feature selection and model estimation, but the resulting function typically depends on all selected features in an untractable way, making the final model less interpretable in complex models such as neural networks.

In this paper, we propose to tackle these issues by introducing an extremely parsimonious class of functions represented by neural networks. Our model SparXnet automatically selects a small number of important input features by ap-

*Both authors contributed equally.

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

plying a neuron-wise softmax transformation to all weights W_u^k between the first and second layers for $u \in \{1, \cdots, d\}$ and $k \in \{1, \cdots, K\}$, where d and K denote the number of neurons in the first and second layers, respectively. Upon saturation, each neuron k in the second layer will have one incoming weight close to one and the rest close to zero, thus selecting a single feature of the input data. In the next layer, SparXnet learns a separate one-dimensional function $f_k : \mathbb{R} \to \mathbb{R}$ for each selected feature. The final predictions (or logit scores in classification problems) are a linear combination of the outputs of each f_k. We represent the functions $\{f_k\}$ as deep neural networks, which are trained jointly in the entire model.

The specific architecture of SparXnet has two immediate consequences: (1) predictions are highly interpretable: indeed, not only can we recover the selected features from the learned weights of the first layer and understand how much importance is given to each of them by looking at the weights of the last layer, but we can also plot the one-dimensional functions $\{f_k\}$ to investigate the individual effects each feature has on the final prediction, the same mechanism as the linear regression framework. For instance, suppose we are trying to predict the likelihood of heart disease in patients based on several factors such as BMI, number of cigarettes smoked per day, and blood pressure. If the model selects systolic blood pressure as one of the relevant features and the corresponding function f_k exhibits a sudden sharp increase at 120 mm Hg, it indicates a threshold phenomenon around that value, thus providing a clear, interpretable threshold for assessing heart disease risk (2) Our model's function class capacity is drastically reduced compared to a traditional neural network. This is achieved in two ways: on the one hand, the feature selection forces the model to focus on a small number of features, which reduces the complexity of the model. Indeed, we provide generalization bounds for SparXnet, which indicate that the sample complexity is linear in the number of selected features, but only logarithmic in the number of features present in the data. In addition, with the very mild assumption that the functions $\{f_k\}$ are Lipschitz continuous, the sample complexity can completely avoid any dependence on the number of parameters or the architectural details of the models representing each of those functions, since the class of Lipschitz continuous functions itself has low sample complexity. Note that this would not be possible in higher dimensions, since the sample complexity of the space of Lipschitz functions is exponential in the input dimension.

Our contributions can be summarized as follows:

- We propose an explainable neural network architecture that performs feature selection and model estimation simultaneously. Our model is parsimonious in that it only applies one-dimensional functions to each chosen feature before combining them with a linear layer.
- We prove generalization bounds for SparXnet, which show a favorable sample complexity of $O\left(KL^2 \log^3(d + L + 1)\right)$, where K is the number of selected features, L is the Lipschitz constant of the learned one-dimensional functions, and d is the original number of input features. In particular, the sample complexity of SparXnet doesn't involve the number of parameters of the networks used to represent the feature transformations $\{f_k\}$, or any other architectural details, and only depends logarithmically on the number of input features in the data.
- To validate our method, we evaluate it on synthetic datasets with several noisy features and one informative feature. SparXnet exhibits superior performance compared to the standard neural network and successfully recovers the true feature and underlying one-dimensional function. In addition, the performance is relatively stable as we add more noisy features, whilst that of the standard neural network baseline deteriorates very fast.
- Finally, we evaluate SparXnet on six real-life datasets, including adult income, breast cancer, credit risk, customer churn, heart disease, and recidivism. We achieve comparable or even superior results compared to feed-forward neural networks and other benchmark models, while preserving a much more interpretable and parsimonious model. We plot the feature transformation functions to further discuss the explainability of our model.

The rest of this paper is organized as follows. In Section 2, we discuss the related works and the improvements we seek to add to existing research. In Section 3, we introduce our notation and describe SparXnet in detail. In Section 4, we prove the sample complexity guarantees for SparXnet. In Section 5, we discuss the results of our experiments on synthetic and real-world datasets. Finally, we conclude in Section 6

2 Related Works

Making neural networks interpretable is a central concern in many machine learning applications (Zhang et al. 2021). For instance, in computer vision, many works attempt to interpret the specific type of features learned by each individual neuron by visualizing the associated representations (Simonyan, Vedaldi, and Zisserman 2013; Zeiler and Fergus 2013). This can then be compared with subjective human-understandable concepts (Zhou et al. 2018, 2014; Zeiler and Fergus 2013). Similarly, in natural language processing, some works attempt to interpret neural networks' representations in terms of the words they use (Dalvi et al. 2019). In biological applications such as DNA sequence analysis, many authors attempt to interpret hidden representations as matching the search for certain explicit amino acid sequences (Stormo et al. 1982). Other works focus on explaining neural network predictions through logic rules such as the presence or absence of certain specific features (Dhurandhar et al. 2018; Goyal et al. 2019; Wachter, Mittelstadt, and Russell 2017). However, although several of those works focus on feature selection, none apply a trainable transformation function.

The explanation of the performance of neural networks, despite their extremely high number of parameters, is a well-developed and active area of research. Earlier works focused on bounding the function class capacity of neural networks in terms of architectural parameters such as the

number of parameters or the norms of the weights (Long and Sedghi 2020; Bartlett, Foster, and Telgarsky 2017; Graf et al. 2022; Ledent et al. 2021). Since then, much of the literature has instead focused on the implicit regularization imposed by the gradient descent procedure and by the underlying structure in the data distribution (Du et al. 2019; Jacot, Gabriel, and Hongler 2018; Arora et al. 2019; Wei and Ma 2019; Nagarajan and Kolter 2019). However, although many of these works rely on the Lipschitz constants of the network to bound complexity, none leverage the especially simple form of one-dimensional functions to sidestep the need for any other contributing terms (such as the norms of the weights or the complexity of the data distribution). It is worth noting that a particularly interesting new line of work (Jacot 2023) has identified the phenomenon that neural networks with standard weight decay regularization may naturally restrict their 'bottleneck rank': irrespective of the number of neurons present in each layer, a lower-dimensional representation of the input is naturally learned in intermediary layers. In particular, this indicates that in the case of a single function, standard neural networks may naturally strive to achieve a similar type of function as the ones learned by SparXnet. However, SparXnet involves several distinct one-dimensional feature-transformation functions rather than one single function whose input space has a small dimension that is still larger than one. In addition, (Jacot 2023) focuses on the training dynamics, while our work focuses on interpretability and generalization.

The idea of using continuous one-dimensional feature transformations on several features goes back to early work in the statistics community on generalized additive models (Hastie and Tibshirani 1986; Sardy and Tseng 2004). In particular, in projection pursuit regression (Friedman and Stuetzle 1981), a linear combination of features is fed through a nonlinear map, though unlike our work, no softmax is used to encourage the selection of a specific feature. Several distinguishing characteristics of our work compared to such early works are (1) the inclusion of generalization bounds from the point of view of modern statistical learning theory, (2) the use of neural networks to model the nonlinear functions and (3) the introduction of feature selection with our softmax operator.

A closely related line of research in modern literature is the Neural Additive Model (NAM) proposed by (Agarwal et al. 2021), which also addresses the challenge of achieving high predictive power while maintaining interpretability through generalized linear models. Our model differentiates itself from NAM in several key aspects. Firstly, we incorporate sparse estimation, which automatically selects the relevant set of features, thereby enhancing generalization performance. Furthermore, our work provides theoretical generalization bounds for the interpretable neural network, a contribution that is absent in previous studies.

3 Methodology

Suppose our input data consists of i.i.d. samples (x^1, y^1), $(x^2, y^2), \ldots, (x^N, y^N)$, where y^i is the label and our inputs $x^i \in \mathbb{R}^d$ contains d interpretable features. For instance, in our real data experiments on heart disease, examples of individual features include the resting heart rate (in beats per minute) and the average blood pressure. We assume that the individual features are suitably normalised so that $\|x^i\|_{\max} \leq \chi$ for some constant χ. Our aim is to simultaneously select a small number $K \ll d$ of the input features and learn K transformation functions $f^1, f^2, \ldots, f^K : \mathbb{R} \to \mathbb{R}$. Each function is represented as a deep neural network for ease of training and will be used to generate target prediction via a linear combination in the final layer. Thus SparXnet's prediction takes the following form:

$$F(x) = \beta + \sum_{k=1}^{K} \theta_k f_k \left(\sum_{u=1}^{d} W_u^k x_u \right). \tag{1}$$

where we assume an upper bound Γ on $\sum_k |\theta_k|$. $F(x)$ is a scalar prediction in regression and logit score in classification, where the predicted probability for the positive class is $\frac{\exp(F(x))}{1+\exp(F(x))}$. β is the bias term, $\{\theta_k\}_{k=1}^K$ denote parameters of the last linear layer, and $\{f_k\}_{k=1}^K$ are K trainable functions represented by separate neural networks. Each W_u^k is determined by the following formula:

$$W_\bullet^k = \text{softmax}\left(w_\bullet^k\right) \qquad \text{i.e.}$$
$$W_u^k = \frac{\exp(w_u^k/\tau)}{\sum_{v=1}^{d} \exp(w_v^k/\tau)} \qquad (\forall u \leq d), \tag{2}$$

where $\{w_\bullet^k\}$ are trainable parameters for the k^{th} sub neural network and τ is a tunable temperature hyperparameter.

Thus, after softmax transformation, each row of the first-layer weight matrix W represents a probability distribution reflecting the relative importance of each input feature for the corresponding neuron, thus prioritizing certain input features over others. As the model is trained, it learns the optimal weight distribution that minimizes the loss function, effectively performing feature selection and model estimation at the same time. In addition, the temperature parameter τ controls the 'sharpness' of the softmax output. The higher the temperature, the more uniform the output distribution will be. Conversely, a lower temperature will make the distribution more sharply peaked.

Figure 1 illustrates the model architecture with two sub networks ($K = 2$,) with x_1 and x_3 being the selected input features[1]. This involves two distinct processing pathways in a feed-forward neural network. The network enforces a softmax operation applied to the weights of the first hidden layer, where softmax serves as a soft form of 'routing' mechanism, allowing the network to learn and distribute the representation of different data characteristics across the two pathways, thus achieving adaptive feature selection. For instance, the first input feature x_1 gets selected via a linear combination $\sum_u W_u^1 x_u$ of the six input features with $W_1^1 \gg W_u^1$ (for $u \neq 1$), where the dominating weight W_1^1 is denoted by the solid line (close to saturation) and the rest

[1]Our model allows the user to specify the number of features to remain, which offers a more precise control over model sparsity as opposed to indirectly configuring the penalty coefficient in Lasso regression.

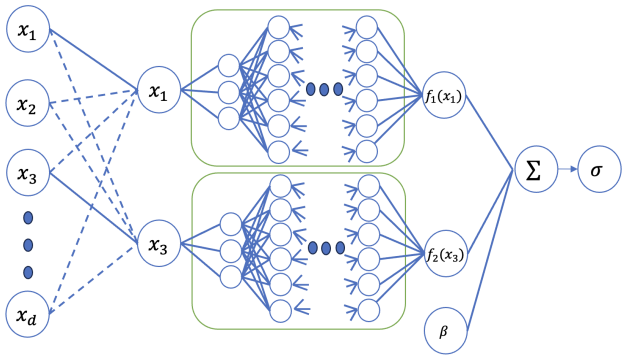

Figure 1: Schematic overview of the proposed model in the case of two selected features ($K = 2$). The neural network has two distinct processing pathways, which are based on a softmax operation applied to the weights of the first hidden layer. The softmax operation serves as a soft form of "routing" mechanism, allowing the network to learn and distribute the representation of different data characteristics across the two pathways, thus achieving adaptive feature selection upon saturation (as indicated by the two solid lines in the first layer). The two fully-connected mapping functions $f_1(x_1)$ and $f_2(x_3)$ are learned automatically and linearly combined to generate the final prediction.

as dashed line. After softmax, the two nonlinear mapping functions $f_1(x_1)$ and $f_2(x_3)$ are automatically learned and linearly combined to generate the final prediction. Overall, training parameters include $\{w_u^k\}$ ($k \leq K$ and $u \leq d$) for the first two layers, β and $\{\theta_k\}$ for the last two layers, and parameters of each individual fully connected network (FCN) $\{f_k\}$.

4 Theoretical Analysis

In this section, we study the sample complexity of SparXnet. Our main result is Theorem 1, whose proof is left to the Appendix (cf. ArXiv version).

Theorem 1. *Consider the function class $\mathcal{F}$ defined above. Suppose we are given N i.i.d. samples $\{(x^i, y^i)\}_{i=1}^N$ with $y^i \in \mathbb{R}$ for all $i \leq N$ and a loss function $\ell : \mathbb{R}^2 \to \mathbb{R}$ which is bounded by B and has a Lipschitz constant at most $\mathcal{L}$. Let*

$$\hat{f} := \arg\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} \ell\left(f(x^i), y^i\right) \tag{3}$$

and

$$f^* := \arg\min_{f \in \mathcal{F}} \mathbb{E}_{x,y} \ell\left(f(x), y\right). \tag{4}$$

We have, with probability $\geq 1 - \delta$ over the draw of the training set:

$$\frac{1}{N} \sum_{i=1}^{N} \ell\left(\hat{f}(x^i), y^i\right) - \mathbb{E}_{x,y} \ell\left(f^*(x), y\right) - 6B\sqrt{\frac{\log(2/\delta)}{2N}}$$
$$\leq \frac{24\mathcal{L}}{\sqrt{N}} \left[15\chi L\Gamma\sqrt{K} + 3\right] \times \tag{5}$$
$$\sqrt{\log_2\left(12dN^2\left[\chi L\Gamma + 1\right]\right)} \log(N).$$

We also obtain the following immediate corollary expressed in terms of excess risk ϵ.

Corollary 2. *Assume the assumptions of Theorem 1 hold with $\mathcal{L}, \chi, \Gamma, B = O(1)$. The number of required samples to reach an excess risk of ϵ is $O\left(\frac{KL^2}{\epsilon^2} \log^3\left(\frac{KL^2 \log(d+L+1)}{\epsilon^2}\right)\right)$.*

Thus, for fixed $\chi, \mathcal{L}, B$, the sample complexity is $\tilde{O}(L^2K)$: up to logarithmic terms, the number of samples required to train the model effectively is proportional to the product of the number of chosen factors K and the square of the bound on the Lipschitzness constant. In particular, the dependency on the original number of features d is only logarithmic. The proof strategy relies on the fact that the set of Lipschitz functions with a *low-dimensional* input (1 or 2 d) has a very mild complexity (von Luxburg and Bousquet 2004; Tikhomirov 1993). This fact was previously exploited in the case of one dimensional inputs in several other contexts such as Matrix Completion (Ledent and Alves 2024) and density estimation (Vandermeulen and Ledent 2021).

In summary, our bound has two particularities, which makes it non-vacuous compared to standard generalization bounds for neural networks:

- There is no dependency on the number of parameters, since the complexity of the classes of 1 to 1 transformation functions is computed purely based on the Lipschitz constant. This contrasts most of the literature on generalization bounds for neural networks, which almost always depend on the number of parameters of the model, whether the dependence is explicit (Long and Sedghi 2020) or implicit (Bartlett, Foster, and Telgarsky 2017; Graf et al. 2022; Ledent et al. 2021).
- The only non-negligible dependency on architectural parameters is on K, not d (although the bound on the norms of the original features is $\|x\|_{\max}$), indicating that the choice of a small number of important features from a multitude of features present in the original data has negligible cost in terms of sample complexity. This indicates that selecting a small number of important features from a high-dimensional input space does not significantly increase the sample complexity, making the model efficient and practical even in high-dimensional settings. This advantage is similar to the characteristic of generalization bounds for multi-class or multi-label prediction scenarios where the dependence is only logarithmic in the total number of possible labels, whilst maintaining nonlogarithmic dependence only on the number labels present (Lei et al. 2019; Mustafa et al. 2021; Wu et al. 2021).

Note that Corollary 2 applies to both the regression and classification settings, with simple modifications of the loss function ℓ. In the regression setting, we can use the truncated square loss:

$$\ell(\hat{y}, y) = \min(|y - \hat{y}|^2, B). \tag{6}$$

In the binary classification setting, we can use the cross entropy loss (with the softmax in the prediction step absorbed):

$$\ell(s, 0) = \log\left(1 + \exp(s)\right) \quad \ell(s, 1) = \log\left(1 + \exp(-s)\right).$$

For simplicity, our theoretical results are provided for the binary classification or the regression setting. However, it is straightforward to extend the results to the multi-class case.

5 Experimental Results

In this section, we present the experimental results on both synthetic and real data. We allocate 20% of the data to the test set in all experiments, with the hyperparameters tuned by cross-validation. Our evaluation of the proposed algorithm serves a dual purpose: first, to gauge its predictive power, and second, to assess its capability to retrieve the sparse set of true features accurately. Through comprehensive analysis, we illustrate that our method establishes a superior equilibrium between predictive accuracy and sparsity compared with prevalent benchmark techniques, thus positioning our approach as an innovative and effective solution for explainable neural networks with guarantee.

Synthetic Data Experiments

A single-variable case In our first synthetic data experiment, we are interested in assessing whether the proposed method can recover the true functional form of the underlying relationship in the presence of different noise levels in both the input features and the observation model. Assuming a ground truth function $y_* = x^2 + 2sin(x) + 3$, we generate 1000 observations uniformly sampled within the interval of [-1,1]. We inject additive Gaussian noise $\epsilon \in \mathcal{N}(0, 0.05)$ into the observations and add a total of J noisy features $x_i \in \mathcal{N}(0, 1)$ for $i \in \{1, ..., J\}$ into the feature space. This allows us to assess the model's capability to discern and recover the true signal from noise.

We use one pathway with six fully connected layers to learn the underlying data-generating process and identify the true feature. Each hidden layer consists of 128 nodes, followed by a dropout layer. We use Bayesian optimization to optimize three hyperparameters: dropout rate (between 0.1 and 0.5), learning rate (between 0.001 and 0.01), and temperature (between 0.1 and 100). We used a high temperature to promote early exploration, as a premature selection of an incorrect input feature during the early training phase may hurt the training performance. The temperature is then slowly reduced to 1% of its initial value throughout a total training budget of 2000 iterations.

Figure 2 illustrates the learned function for this synthetic regression problem that includes one true feature and two noisy ones. We intentionally position the true feature in the middle of the design matrix to circumvent the possible default choice of selecting the first feature upon saturation. The figure suggests that SparXnet can recover the true shape of the underlying function, despite adversarial perturbation in both the observational and feature spaces. Specifically, SparXnet correctly selects the second input feature in the first layer, as evidenced by the learned weights of 3.76221635e-07, 9.99999642e-01, and 3.97033251e-09 in the first hidden layer.

To further assess our method under different signal-to-noise ratios, we progressively increase the number of noisy features from 2 to 5 while keeping the true feature in the central position of the design matrix. We use two benchmark methods for comparison: a fully connected neural network with the same number of layers and a Lasso regression model. As shown in Table 1 across five runs, SparXnet has a much lower test set MSE than others. This advantage is even more pronounced as we increase the number of noisy features in the data. We also observe that SparXnet can identify and recover the true feature in all experiments. This suggests that SparXnet can achieve good predictive performance while recovering the true feature simultaneously, when the ground truth is indeed sparse. The sparse solution also offers direct interpretability of the learned features, which will be further discussed in our next experiments.

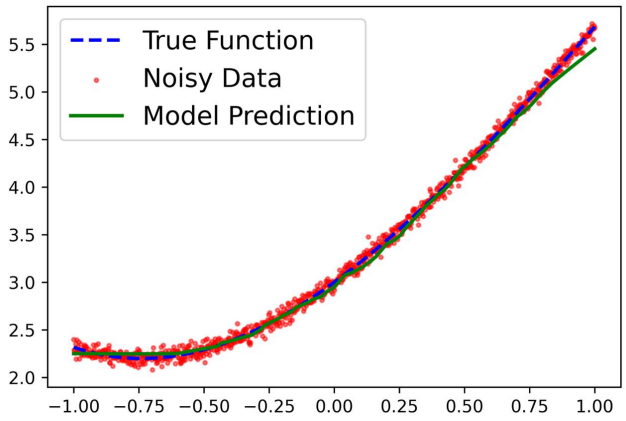

Figure 2: Visualizing the learned model predictions using 2000 noisy observations and three features, including one true feature and two random noise features. Our model can correctly identify the second feature as the true input feature (based on the learned weights of 4.8417e-07, 9.9999e-01 and 3.6648e-09 in the first hidden layer) and recover the original form of the underlying data-generating function.

A multi-variable case We now extend the results to a more challenging case with multiple variables to answer the following question: How will the recovery rate of the true features and the predictive power change as we vary the level of sparsity (controlled by the number of pathways) in SparXnet? This problem has a high practical relevance when one intends to select a subset of features for easy interpretation but is unsure of the exact number of true features present in the underlying model.

To this front, we assume a highly non-linear underlying function with five true features: $y_{\text{true}} = \sin(x_0) + 2x_1^2 - 3x_2^2 + 4e^{x_3} - 5e^{x_4}$. We also include five standard normal features, which are randomly arranged in a design matrix comprising 1000 observations. We track the true feature recovery rate and test-set MSE at different sparsity levels to evaluate the trade-off between these two objectives. Intuitively, one would expect a lower test-set MSE and a higher true feature recovery rate as more input features are present in the model (corresponding to a low sparsity level). However, having excessive noisy features will increase test-set MSE despite a high recovery rate of true features.

Data	Model	Number of noisy features			
		2	3	4	5
Train	SparXnet	**0.0047** (0.0017)	**0.0038** (0.0009)	**0.0047** (0.0013)	**0.0052** (0.0023)
	FCN	0.0313 (0.0145)	0.2839 (0.5547)	0.0253 (0.0024)	0.4415 (0.5792)
	Lasso	0.1248 (0.0012)	0.1245 (0.0019)	0.1251 (0.0023)	0.1244 (0.0015)
Test	SparXnet	**0.0048** (0.0014)	**0.0039** (0.0009)	**0.0048** (0.0013)	**0.0054** (0.0022)
	FCN	0.0327 (0.0159)	0.2681 (0.5204)	0.0269 (0.0034)	0.4207 (0.5393)
	Lasso	0.1182 (0.0137)	0.1204 (0.0146)	0.1175 (0.0141)	0.1191 (0.0146)

Table 1: Comparing the mean and standard deviation of MSE for training and test sets across five runs. SparXnet has a much lower training and test set MSE than alternative models. This advantage is even more pronounced as we increase the number of noisy features in the data.

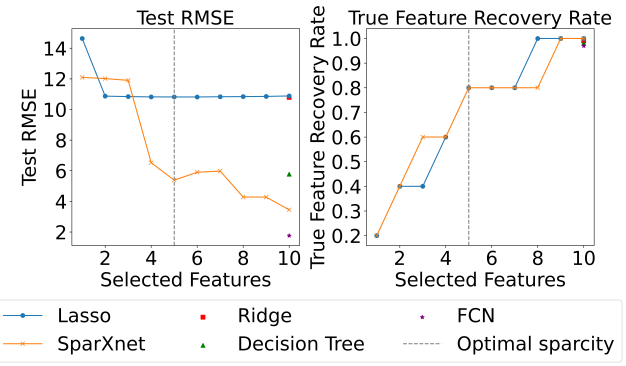

Figure 3: The dynamics between predictive accuracy and recovery rate of true features as the sparsity level varies across different models, including SparXnet, Lasso, Ridge regression, decision tree, and FCN. At high sparsity levels (fewer than four features), SparXnet has a higher test RMSE than Lasso. However, as the number of features increases, SparXnet exhibits enhanced performance, eventually surpassing Ridge regression and decision tree models, and approaching the performance of FCN. SparXnet also demonstrates superior or comparable recovery rates of true features relative to Lasso, except when eight features are selected, where potential overlapping coverage of features chosen by SparXnet slightly reduces its recovery rate.

Figure 3 illustrates the relationship between predictive accuracy and recovery rate at different sparsity levels in all models, including SparXnet, Lasso regression, Ridge regression, decision tree, and FCN. For test-set MSE, SparXnet performs comparably to Lasso when fewer than four features are retained and significantly outperforms Lasso as more features are added to the final model by adjusting the sparsity level. When using all available input features, SparXnet performs inferior to FCN, highlighting SparXnet's efficiency in extracting the most informative features for prediction.

Moreover, SparXnet demonstrates excellent performance in recovering the true features, often matching or exceeding the recovery rates of Lasso. An exception occurs when eight features are selected, where SparXnet slightly lags behind Lasso. This can be attributed to a potential overlapping coverage of features chosen by SparXnet, a complexity absent in Lasso due to its non-overlapping selection mechanism. It is also important to note that all other benchmark methods rely on the full set of features and, therefore, offer no sparsity.

Real Data Experiments

The sparse estimation approach shines in applications where one prefers a small subset of interpretable features. For example, a credit officer needs to rely on a small set of important features to make a credit lending decision, which must also be fully explainable when making pass/fail decisions. Similarly, a doctor often attributes disease to a small number of causes when explaining it to patients. Therefore, our experiments with real data are designed to cover these real-world scenarios that emphasize both predictiveness and explainability in a classification setting. We also assess a wider pool of benchmark models, including logistic regression, FCN, NAM, decision tree, and XGBoost.

We implemented a consistent network architecture for all six datasets, including adult income, breast cancer, credit risk, customer churn, heart disease, and recidivism. We varied the number of nodes K in the first hidden layer, corresponding to the number of features the model would select. In each set of experiments, we optimized the hyperparameters for SparXnet, including the learning rate, batch size, architecture of hidden layers, dropout rate, and the seed for the train-test split using Bayesian optimization based on the validation set. We then evaluated the optimal model configurations on the test set. Table 2 displays the mean and standard deviation of the test set AUC from the top 5 (lowest validation loss) out of 30 repeated experiments. In particular, SparXnet consistently ranks among the top three models in all datasets, demonstrating robust predictive performance in various contexts.

To assess the impact of the number of pathways in the first hidden layer on model performance, we analyzed its relationship with the average test set MSE. Specifically, we selected the top five models with the lowest validation errors for each number of pathways, while also ensuring that the number of pathways did not exceed the total number of features available for each dataset. See the Appendix for further analysis on the relationship between remaining number of features and out-of-sample MSE.

To further highlight the uniqueness of our approach in simultaneous model estimation and feature selection, we provide an example of the inference procedure for two sample applicants from the credit risk dataset. Figure 4 demonstrates the inference examples corresponding to high-risk and low-risk applicants, respectively, showcasing the distribution of individual features and the predicted probability output. We selected six pathways with the highest marginal increase in generalization performance, as shown in the appendix, and plot the density plot of each pathway

Dataset	FCN	NAM	Logistic Regression	Decision Tree	XGBoost	SparXnet
Adult	0.887 (0.015)	0.900 (0.003)	0.854 (0.000)	0.897 (0.002)	0.924 (0.002)	0.899 (0.008)
Breast	0.988 (0.005)	0.645 (0.088)	0.998 (0.000)	0.953 (0.004)	0.995 (0.001)	0.989 (0.010)
Credit Risk	0.893 (0.011)	0.849 (0.004)	0.851 (0.000)	0.899 (0.012)	0.949 (0.002)	0.910 (0.003)
Customer Churn	0.754 (0.016)	0.784 (0.056)	0.837 (0.000)	0.757 (0.012)	0.838 (0.003)	0.832 (0.009)
Heart Disease	0.874 (0.027)	0.546 (0.076)	0.945 (0.001)	0.789 (0.044)	0.950 (0.004)	0.853 (0.030)
Recidivism	0.668 (0.012)	0.669 (0.024)	0.716 (0.000)	0.625 (0.004)	0.715 (0.005)	0.703 (0.010)

Table 2: Average out-of-sample AUC, with standard deviations in parentheses, for various predictive models across different datasets. Notably, SparXnet consistently achieves top-three performance across all evaluated datasets.

before and after their respective transformations.

In this experiment, SparXnet identified three significant features: personal age, personal employment length, and loan percentage income. The final layer weights for these features were 0.1294, 0.1144, and 0.0883, respectively. The signs of the estimated weights for all selected features align with intuitive expectations. For instance, applicants within a certain age group display a consistent risk profile; however, the risk escalates markedly beyond a specific age threshold. The other three features can be excluded during inference, demonstrating the efficacy of automatic feature selection.

Note that the model was trained with an additional temperature parameter to promote the saturation of the weights learned in the first layer, thus accelerating feature selection. Indeed, unimportant features such as loan in rate show zero or near-zero weights, demonstrating the efficacy of this feature selection mechanism. The model, trained with the same hyperparameters as the best model with six pathways and the lowest validation loss, achieves an AUC of 0.82, which is comparable to alternative models. See more details in the Appendix on weight satisfaction in the softmax layer.

Our framework also facilitates transparent tracking of the decision-making process. For instance, in Figure 4, the high-risk applicant, indicated in red, exhibits large output values for loan grade and loan intent, each exceeding critical thresholds that increase the probability of predicted default. This probability score is then contrasted with those of the broader population, facilitating a comprehensive credit risk assessment (Liu 2023). This prediction results from a linear combination of transformed pathways and a softmax operation in the last layer. In contrast, the low-risk applicant, indicated in blue, remains within safe ranges for all features, resulting in a low default probability.

This ability to assess individual feature-level critical regions during inference showcases the strength of SparXnet in explainable sparse estimation. Importantly, this inferential framework enables the interpretation of factors driving high-risk applicants. One possible downside to the approach is that the softmax may not always saturate, leading to some of the feature selection nodes incorporating multiple features. Whilst this can improve representation power and accuracy (Liu et al. 2024), this is done at the cost of some of the interpretability gains that define SparXnet. Mitigating this phenomenon through a more effective tuning of the temperature parameter is an interesting avenue of research for future work. In addition, the smoothness of some of the predictive feature transformations could be further improved

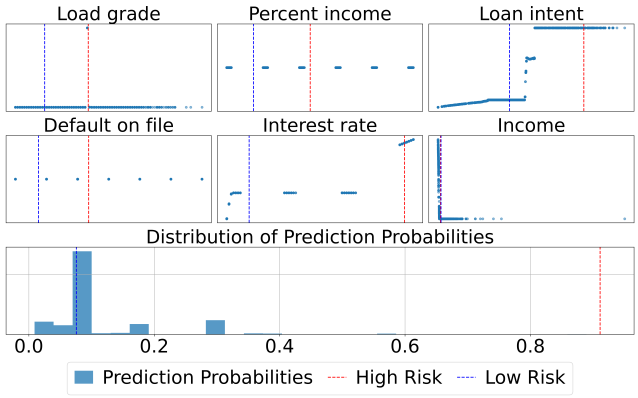

Figure 4: Illustrating the model inference for two credit applicants in the credit risk dataset.

with gradient regularization. Lastly, we acknowledge that the arbitrary linear combination involved at the last layer means the level of interpretability still falls short of purely rules-based interpretable methods (Rudin 2019). Replacing the last linear combination layer with an even more interpretable strategy is left to future work.

6 Conclusion

Our proposed SparXnet provides a theoretically motivated and empirically effective approach in the field of sparse estimation. We have introduced a parsimonious neural network architecture and a training procedure for feature selection in applications with small datasets and high interpretability requirements. Our architecture comprises a softmax layer that selects individual features, trainable 1-dimensional Lipschitz functions, and a final linear layer. The Lipschitz functions are learned by neural networks. We show that the sample complexity of SparXnet only scales like the number of chosen features, with a logarithmic dependence on the total number of features involved in the inputs. In addition, the sample complexity is independent of the number of parameters involved in each transformation function. Synthetic data experiment demonstrate that SparXnet can successfully select the right feature and recover the ground truth function in controlled settings. In real-life datasets, we show that SparXnet can equal or surpass alternatives, despite reduced model complexity and improved interpretability.

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. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 4699–4711. Curran Associates, Inc.

Arora, S.; Du, S.; Hu, W.; Li, Z.; and Wang, R. 2019. Fine-Grained Analysis of Optimization and Generalization for Overparameterized Two-Layer Neural Networks. In *ICML*.

Bartlett, P. L.; Foster, D. J.; and Telgarsky, M. J. 2017. Spectrally-normalized margin bounds for neural networks. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*, 6240–6249. Curran Associates, Inc.

Church, K. W. 2017. Word2Vec. *Natural Language Engineering*, 23(1): 155–162.

Dalvi, F.; Durrani, N.; Sajjad, H.; Belinkov, Y.; Bau, A.; and Glass, J. 2019. What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 6309–6317.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dhurandhar, A.; Chen, P.-Y.; Luss, R.; Tu, C.-C.; Ting, P.; Shanmugam, K.; and Das, P. 2018. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. *Advances in neural information processing systems*, 31.

Du, S. S.; Zhai, X.; Poczos, B.; and Singh, A. 2019. Gradient Descent Provably Optimizes Over-parameterized Neural Networks. In *International Conference on Learning Representations*.

Friedman, J. H.; and Stuetzle, W. 1981. Projection pursuit regression. *Journal of the American Statistical Association*, 76(376): 817–823.

Goyal, Y.; Wu, Z.; Ernst, J.; Batra, D.; Parikh, D.; and Lee, S. 2019. Counterfactual visual explanations. In *International Conference on Machine Learning*, 2376–2384. PMLR.

Graf, F.; Zeng, S.; Rieck, B.; Niethammer, M.; and Kwitt, R. 2022. On measuring excess capacity in neural networks. *Advances in Neural Information Processing Systems*, 35: 10164–10178.

Hastie, T.; and Tibshirani, R. 1986. Generalized additive models. *Statistical Science*, 1(3): 297–310.

He, F.; Liu, T.; and Tao, D. 2020. Why resnet works? residuals generalize. *IEEE transactions on neural networks and learning systems*, 31(12): 5349–5362.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Jacot, A. 2023. Implicit Bias of Large Depth Networks: a Notion of Rank for Nonlinear Functions. In *The Eleventh International Conference on Learning Representations*.

Jacot, A.; Gabriel, F.; and Hongler, C. 2018. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Ledent, A.; and Alves, R. 2024. Generalization Analysis of Deep Non-linear Matrix Completion. In Salakhutdinov, R.; Kolter, Z.; Heller, K.; Weller, A.; Oliver, N.; Scarlett, J.; and Berkenkamp, F., eds., *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 26290–26360. PMLR.

Ledent, A.; Mustafa, W.; Lei, Y.; and Kloft, M. 2021. Norm-Based Generalisation Bounds for Deep Multi-Class Convolutional Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9): 8279–8287.

Lei, Y.; Dogan, U.; Zhou, D.-X.; and Kloft, M. 2019. Data-Dependent Generalization Bounds for Multi-Class Classification. *IEEE Transactions on Information Theory*, 65(5): 2995–3021.

Liu, P. 2023. An integrated framework on human-in-the-loop risk analytics. *Journal of Financial Data Science*, 5(1): 58–64.

Liu, Z.; Wang, Y.; Vaidya, S.; Ruehle, F.; Halverson, J.; Soljačić, M.; Hou, T. Y.; and Tegmark, M. 2024. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*.

Long, P. M.; and Sedghi, H. 2020. Size-free generalization bounds for convolutional neural networks. In *International Conference on Learning Representations*.

Mustafa, W.; Lei, Y.; Ledent, A.; and Kloft, M. 2021. Fine-grained Generalization Analysis of Structured Output Prediction. In Zhou, Z.-H., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 2841–2847. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Nagarajan, V.; and Kolter, J. Z. 2019. Deterministic PAC-Bayesian generalization bounds for deep networks via generalizing noise-resilience. *CoRR*, abs/1905.13344.

Roth, V. 2004. The generalized LASSO. *IEEE transactions on neural networks*, 15(1): 16–28.

Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5): 206–215.

Sardy, S.; and Tseng, P. 2004. AMlet, RAMlet, and GAMlet: Automatic Nonlinear Fitting of Additive Models, Robust and Generalized, With Wavelets. *Journal of Computational and Graphical Statistics*, 13(2): 283–309.

Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2013. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *CoRR*, abs/1312.6034.

Stormo, G. D.; Schneider, T. D.; Gold, L.; and Ehrenfeucht, A. 1982. Use of the 'Perceptron'algorithm to distinguish translational initiation sites in E. coli. *Nucleic acids research*, 10(9): 2997–3011.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–9.

TAN, M.; DAI, Y.; TANG, D.; FENG, Z.; HUANG, G.; JIANG, J.; LI, J.; and SHI, S. 2022. Exploring and adapting Chinese GPT to pinyin input method. Association for Computational Linguistics.

Tibshirani, R. J. 2013. The lasso problem and uniqueness.

Tikhomirov, V. M. 1993. *ϵ-Entropy and ϵ-Capacity of Sets In Functional Spaces*, 86–170. Dordrecht: Springer Netherlands. ISBN 978-94-017-2973-4.

Vandermeulen, R. A.; and Ledent, A. 2021. Beyond smoothness: Incorporating low-rank analysis into nonparametric density estimation. *Advances in Neural Information Processing Systems*, 34: 12180–12193.

von Luxburg, U.; and Bousquet, O. 2004. Distance-Based Classification with Lipschitz Functions. *J. Mach. Learn. Res.*, 5(Jun): 669–695.

Wachter, S.; Mittelstadt, B.; and Russell, C. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.*, 31: 841.

Wei, C.; and Ma, T. 2019. Data-dependent Sample Complexity of Deep Neural Networks via Lipschitz Augmentation. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32*, 9725–9736. Curran Associates, Inc.

Wu, L.; Ledent, A.; Lei, Y.; and Kloft, M. 2021. Fine-grained generalization analysis of vector-valued learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 10338–10346.

Zeiler, M. D.; and Fergus, R. 2013. Visualizing and Understanding Convolutional Networks. *CoRR*, abs/1311.2901.

Zhang, Y.; Tiňo, P.; Leonardis, A.; and Tang, K. 2021. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5): 726–742.

Zhou, B.; Bau, D.; Oliva, A.; and Torralba, A. 2018. Interpreting Deep Visual Representations via Network Dissection. *IEEE transactions on pattern analysis and machine intelligence*.

Zhou, B.; Khosla, A.; Lapedriza, À.; Oliva, A.; and Torralba, A. 2014. Object Detectors Emerge in Deep Scene CNNs. *CoRR*, abs/1412.6856.

Zou, H. 2006. The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476): 1418–1429.