# Locally Sparse Neural Networks for Tabular Biomedical Data

**Junchen Yang** [*1] **Ofir Lindenbaum** [*2] **Yuval Kluger** [1 3 4]

## Abstract

Tabular datasets with low-sample-size or many variables are prevalent in biomedicine. Practitioners in this domain prefer linear or tree-based models over neural networks since the latter are harder to interpret and tend to overfit when applied to tabular datasets. To address these neural networks' shortcomings, we propose an intrinsically interpretable network for heterogeneous biomedical data. We design a locally sparse neural network where the local sparsity is learned to identify the subset of most relevant features for each sample. This sample-specific sparsity is predicted via a *gating* network, which is trained in tandem with the *prediction* network. By forcing the model to select a subset of the most informative features for each sample, we reduce model overfitting in low-sample-size data and obtain an interpretable model. We demonstrate that our method outperforms state-of-the-art models when applied to synthetic or real-world biomedical datasets using extensive experiments. Furthermore, the proposed framework dramatically outperforms existing schemes when evaluating its interpretability capabilities. Finally, we demonstrate the applicability of our model to two important biomedical tasks: survival analysis and marker gene identification.

## 1. Introduction

Machine learning has revolutionized the way we do scientific research. In recent years, deep neural networks (NN) have closed the performance gap between humans and machines in disciplines such as vision, image processing, audio processing, and natural language processing. The tremendous success of these complex models may be explained by an increase in data size, computational resources that enable training deeper networks (Tishby & Zaslavsky, 2015; Arora et al., 2016), or by implicit properties of the optimization tools (Yaguchi et al., 2018; Soudry et al., 2018). State-of-the-art frameworks, such as convolutional neural networks, recurrent neural networks, and transformers exploit structures or invariants in the data to inform the design of the NN. Unfortunately, these models are not suitable for biomedical applications when the associated datasets are tabular, lack spatial or temporal structure, or are heterogeneous. Therefore, biomedical data pose a challenge for deep nets and require deviating from tried and true methodologies (Borisov et al., 2021; Marais, 2019; Shavitt & Segal, 2018; Shwartz-Ziv & Armon, 2021).

In medicine or biology, practitioners seek for ML models that are *accurate* and *interpretable*. Accuracy is important for improving personalized prognosis and diagnosis. At the same time, interpretability can lead to the identification of driving factors in complex high-dimensional systems and is imperative to help practitioners trust the model. When trained on tabular biomedical data, deep nets are hard to interpret and may lead to low accuracy. This is because biomedical datasets are often low-sample-size (LSS) (Liu et al., 2017; Aoshima et al., 2018), high dimensional, or contain nuisance features. These challenges often lead practitioners to abandon NNs and switch to linear models in biomedicine. Linear feature selection models such as (Tibshirani, 1996; Fan & Li, 2001; Lindenbaum & Steinerberger, 2021a;b) are interpretable and able to cope with high-dimensional low sample size (HDLSS) data but come with the cost of limited expressivity.

In contrast, deep over-parametrized nets are highly expressive but tend to overfit on LSS tabular data; one reason is that in this regime, the vast amount of parameters leads to a large variance of the gradient estimates (Liu et al., 2017). To prevent model overfitting, several authors (Li et al., 2016; Scardapane et al., 2017; Feng & Simon, 2017; Yamada et al., 2020) have proposed to apply different regularization schemes to sparsify the input features. Since these models select one global set of informative features, they are not suited for the heterogeneity of tabular biomedical

[*]Equal contribution [1]Interdepartmental Program in Computational Biology and Bioinformatics, Yale University, New Haven, CT, USA [2]Faculty of Engineering, Bar Ilan University, Ramat Gan, Israel [3]Applied Math Program, Yale University, New Haven, CT, USA [4]Department of Pathology, School of Medicine, Yale University, New Haven, CT, USA. Correspondence to: Yuval Kluger <yuval.kluger@yale.edu>.

*Proceedings of the $39^{th}$ International Conference on Machine Learning*, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

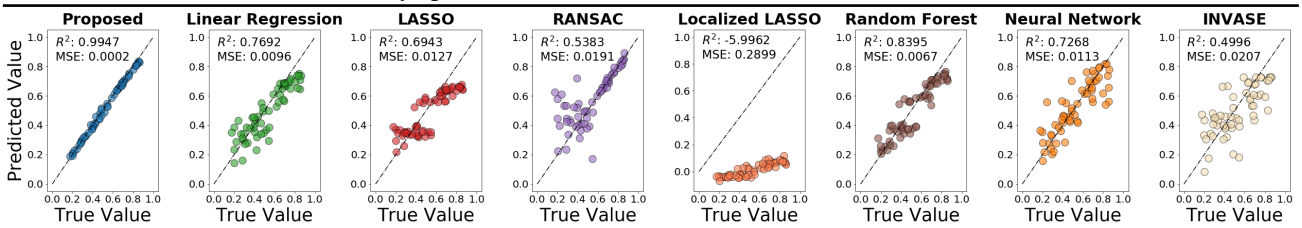



Figure 1: Strong ML models fail to learn the correct regression function when the target depends on different subsets of variables for different samples (see data model in Eq. 1). Each subplot presents the predicted (y-axis) vs. true (x-axis) values for different baselines. Points on the diagonal line indicate correct predictions. R-square and Mean Squared Error (MSE) are reported for each model. In this work, our proposed model (left plot) that effectively identifies the informative features for each sample while learning the regression coefficients.

data. These global feature selection models occasionally result in unsatisfactory performance and do not provide a sample-specific explanation for the predictions.

Our working hypothesis is that since biomedical data is heterogeneous, different samples may require distinct prediction functions. Therefore, we design a simple yet remarkably effective NN-based framework that leads to dramatically higher prediction performance yet is intrinsically interpretable and is less prone to overfitting. Specifically, we propose a Locally SParse Interpretable Network (LSPIN), a NN that incorporates interpretability into its design (self-explanatory) by sparsifying the input variables used by the model locally (for each sample) while learning a *prediction* function. To identify the local sparsity patterns, we train a *gating* network to predict the probabilities of the instance-wise *gates* being active. The parameters of the local gates, along with the model coefficients, are learned in tandem by minimizing a classification or regression loss. Our parametric construction leads to a highly interpretable (locally sparse) model which relies on a small subset of the input features for each instance.

Our contributions are: (i) We propose a probabilistic $\ell_0$ like regularization that leads to sample-specific feature selection that is *stable* across close samples defined by an affinity kernel. (ii) By training our *gating* network alongside a *prediction* network, we obtain a powerful interpretable NN framework for tabular biomedical data. (iii) We show via extensive synthetic simulations that our model, albeit simple, can learn the correct target function and identify the informative variables while requiring a small number of observations. (iv) We demonstrate a strong property of our framework: a linear predictor with local sparsity often outperforms state-of-the-art nonlinear models when applied to real-world datasets. (v) We explore the applicability of the proposed approach to several challenging tasks in biomedicine, including survival analysis and marker genes identification.

## 2. Motivating Example

To motivate our proposed framework, let us consider a data matrix $\boldsymbol{X}$, with $N$ measurements (e.g., patients) and $D$ variables (e.g., genes). Given a target variable $y$, in supervised learning, we are interested in modeling a function $\boldsymbol{f}$ that can predict the target $y$ based on the observations $\boldsymbol{X}$. Unfortunately, if $D > N$ learning such function becomes challenging and may lead to overfitting. Moreover, since the biomedical datasets are heterogeneous, the informative variables may vary within different population subsets. To clarify this point, let us consider the following simple linear regression problem. We are given a small ($N = 10$) synthetic dataset in which the response variable $y$ of different samples depends on different subsets of features of the data matrix $\boldsymbol{X}$. We assume that the data comprise two subpopulations, which we term here group-1 and group-2. The corresponding response variable $y$ for samples in group-1 and group-2 is defined in Eq. 1 where for group-1, the response is a linear combination of the $1^{\text{st}}$,$2^{\text{nd}}$, and $3^{\text{rd}}$ features, and for group-2 it is a linear combination of the $3^{\text{rd}}$,$4^{\text{th}}$, and $5^{\text{th}}$ features:

$$y = \begin{cases} -2\boldsymbol{x}_1 + \boldsymbol{x}_2 - 0.5\boldsymbol{x}_3, & \text{if in group 1,} \\ -0.5\boldsymbol{x}_3 + \boldsymbol{x}_4 - 2\boldsymbol{x}_5, & \text{if in group 2.} \end{cases} \quad (1)$$

Group 1 and 2 are defined by drawing values for $\boldsymbol{x}_1 - \boldsymbol{x}_5$ from separated Gaussians (details appear in Appendix section B.3). The simple example above comprises two different linear relationships between the response $y$ and the observed variables. Since we do not know the membership of each point to one of the two groups, attempting to fit a single model to this data is challenging. One way to model this type of relationship is via Hierarchical Bayesian Models (HBM) (Allenby & Rossi, 2006; Shiffrin et al., 2008). HBMs combine multiple sub-models with Bayes' theorem to estimate the posterior distribution of the data. While HBMs can account for data dependent parameters, ensuring sparsity of the features requires careful design of the prior distributions. To demonstrate the difficulty involved in modeling data generated by the model above, in Fig. 1 we demonstrate that strong ML models fail to learn the correct regression function. In this study, we design LSPIN that is able to learn the correct target function while accurately

identifying the informative features for each sample (see Appendix Fig. A.1).

## 3. Problem Setup and Method

We are interested in the standard supervised learning based on tabular biomedical data points $\{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$, where $\boldsymbol{x}^{(i)} \in \mathbb{R}^D$, with $x_d^{(i)}$ representing the $d^{th}$ feature of the $i^{th}$ vector-valued observations. Our goal is to design a method that can overcome the challenges posed by biomedical datasets while leading to *accurate* and *interpretable* predictions. Specifically, we want to learn an intrinsically interpretable prediction model $\boldsymbol{f_\theta} \in \mathcal{F}$ with:
**P1** Small generalization error even in cases of $N < D$.
**P2** Sample-specific removal of nuisance variables whose inclusion could be detrimental for predictions.
**P3** High expressive power.

Perhaps the most simple model that leads to **P1** is the LASSO (Tibshirani, 1996). The LASSO minimizes the empirical risk of a linear model $\boldsymbol{f}(\boldsymbol{x}^{(i)}) = \boldsymbol{\theta}\boldsymbol{x}^{(i)}$ (with $\boldsymbol{\theta}^T \in \mathbb{R}^D$), while penalizing for the sum of absolute values of active coefficients. This global linear feature selection model is interpretable since it provides an additive quantification to the contribution of each variable. To enable sample-specific variable selection (**P2**), the Localized LASSO was introduced in (Yamada et al., 2017). The authors introduce local weights $\boldsymbol{\theta}^{(i)}$ to the following model $\boldsymbol{f}(\boldsymbol{x}^{(i)}) = \boldsymbol{\theta}^{(i)}\boldsymbol{x}^{(i)}$, and minimize the empirical risk with a network type regularization $\lambda_1 \sum r_{i,j} \|\boldsymbol{\theta}^{(i)} - \boldsymbol{\theta}^{(j)}\|_2 + \lambda_2 \sum \|\boldsymbol{\theta}^{(i)}\|_1^2$. The first term regularizes models parameters to be similar if $r_{i,j} > 0$, where the values of $r_{i,j}$ are given by a graph that represents affinities between samples. The second term encourages local sparsity. While the Localized LASSO addresses **P1** and **P2**, it only relies on linear relations between features to learn the coefficients $\boldsymbol{\theta}^{(i)}$. Furthermore, the model has limited generalization capabilities since coefficients of unseen samples are estimated based on neighbors in the training set.

In this work, we extend the Localized LASSO by using a NN framework to learn the local sparsity patterns and enable more expressive prediction functions. To this end, we propose a NN framework with local sparsity such that predictions are only based on a small subset of features $\mathcal{S}^{(i)} \subset \{1, 2, \ldots, D\}, i = 1, ..., N$ which is optimized for each sample individually. By forcing $|\mathcal{S}^{(i)}| \ll D$, we can reduce the generalization gap of the model and use the (sample-specific) subset of selected features to interpret the prediction model.

### 3.1. Locally Sparse Predictor

Given labeled observations $\{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$, we want to learn a global prediction function $\boldsymbol{f_\theta}$ (parametrized using a NN) and sets of indicator vectors $\boldsymbol{s}^{(i)} \in \{0,1\}^D$ ($s_j^{(i)} = 1$ if $j \in \mathcal{S}^{(i)}$ and 0 otherwise) that will "highlight" which subset of variables the model should rely on for the prediction of each target value $y_i$. This will enable the model to attain to fewer features for each sample and therefore reduce overfiting.

Such a model can be learned by minimizing the following empirical regularized risk

$$\frac{1}{N}\sum_{i=1}^{N} \mathcal{L}\big(\boldsymbol{f_\theta}(\boldsymbol{x}^{(i)} \odot \boldsymbol{s}^{(i)}), y^{(i)}\big) + \frac{\lambda}{N}\sum_{i=1}^{N} \|\boldsymbol{s}^{(i)}\|_0, \quad (2)$$

where $\mathcal{L}$ is a desired loss function (e.g., cross-entropy), and $\odot$ represents the Hadamard product (element-wise multiplication), and $\lambda$ is a regularization parameter that controls the sparsity level of the model. Unfortunately, due to the discrete nature of the $\ell_0$ regularizer, this objective is not differentiable, and finding the optimal solution becomes intractable. Moreover, even if it finds the optimal solution, it is not clear that the model would not overfit when $N < D$. Explicitly, in this regime, the model may select one feature for each sample and "memorize" the training set. Also, it is unclear how such a model can generalize on unseen samples (at test time), namely, because we need to predict the indicator vectors $\boldsymbol{s}$ for the unlabeled data.

### 3.2. Probabilistic Reformulation of the $\ell_0$ Regularizer

Fortunately, the $\ell_0$ norm can be relaxed via a probabilistic differentiable counterpart, specifically, by replacing the binary indicator vector $\boldsymbol{s}$ with a Bernoulli vector $\tilde{\mathbf{s}}$, with independent entries which satisfy $\mathbb{P}(\tilde{\mathrm{s}}_d = 1) = \pi_d$ for $d \in \{1, 2, \ldots, D\}$. Such probabilistic formulation (of binary indicator vectors) converts the combinatorial search (over the discrete space of $\boldsymbol{s} \in \{0,1\}^D$) to a search over the continuous space of Bernoulli parameters ($\boldsymbol{\pi} \in [0,1]^D$). This formulation becomes useful in several applications such as: model compression (Louizos et al., 2017), feature selection (Yamada et al., 2020), discrete softmax (Jang et al., 2016), and many more.

By replacing the deterministic vectors $\boldsymbol{s}^{(i)}$ in Eq. 2 with their probabilistic counterparts $\tilde{\mathbf{s}}^{(i)}$ we can now differentiate through the random variables using REINFORCE (Williams, 1992) or REBAR (Tucker et al., 2017). However, these methods suffer from high variance and require many Monte Carlo samples. Alternatively, as demonstrated in (Maddison et al., 2016; Yamada et al., 2020) a continuous reparametrization of the discrete random variables can reduce the variance of the gradient estimates. In the next section, we propose to learn the indicator vectors $\boldsymbol{s}^{(i)}(\boldsymbol{x}_i)$ by re-formalizing them as random vectors whose parameters (probabilities of being active) are predicted using a NN.

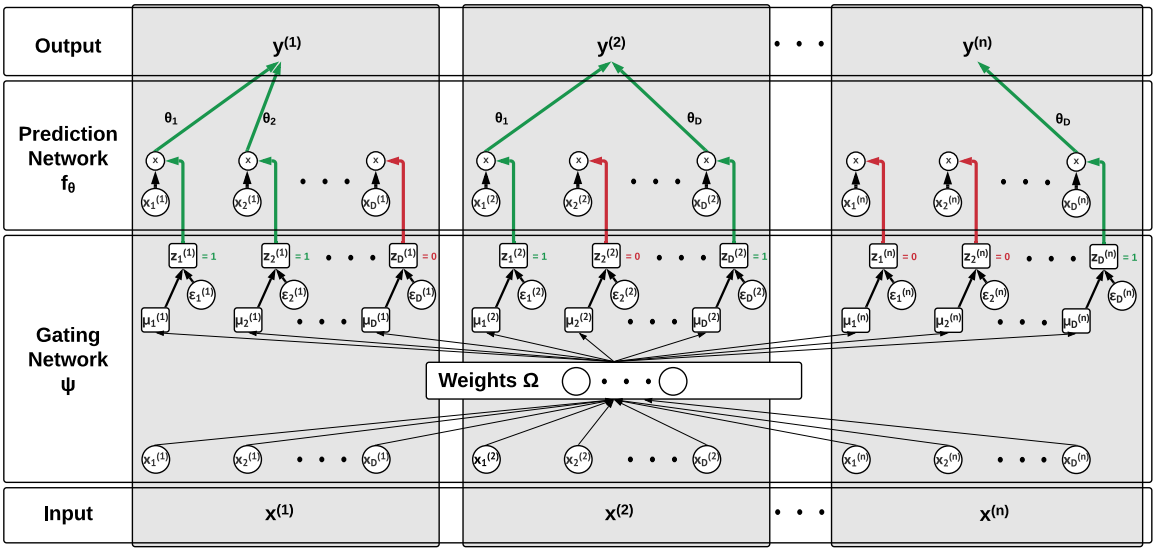



Figure 2: The architecture of Locally Linear SParse Interpretable Networks (LLSPIN). The data $\{\boldsymbol{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, ..., x_D^{(i)}]\}_{i=1}^n$ is fed simultaneously to a *gating* network $\boldsymbol{\Psi}$ and to a *prediction* network $\boldsymbol{f_\theta}$ (which is linear in this example). The *gating* network $\boldsymbol{\Psi}$ learns to predict a set of parameters $\{\mu_d^{(i)}\}_{d=1,i=1}^{D,n}$. The parameters $\mu_d^{(i)}$ depict the behavior of local stochastic gates $\mathrm{z}_d^{(i)} \in [0,1]$ that sparsify (for each instance $i$) the set of features that propagate into in the prediction model $\boldsymbol{f_\theta}$. LLSPIN leads to sample-specific (local) sparsification (obtained via the *gating* network). Therefore, it can handle extreme cases of LSS and lead to interpretable predictions (since the model only uses a small subset of features for each sample). For illustration purposes, we overlay this figure using green (active) and red (non-active) arrows, which indicate that some samples require two features while others only one, in this example. In Section 5, we demonstrate using extensive experiments that our model, leads to accurate predictions in the challenging regime of LSS data.

### 3.3. Locally Sparse NN

This section describes the proposed Locally SParse Interpretable Network (LSPIN). LSPIN is a *prediction* NN with sample-specific gates which sparsify the variables used by the model locally. The sparsity patterns of the gates are learned via a second NN. This leads to a natural framework to predict the sparsity patterns of unseen samples. Fig. 2 shows the architecture of our model where the *prediction* NN is linear. We call this variant of our model Locally Linear SParse Interpretable Network (LLSPIN).

Each stochastic gate (for feature $d$ and sample $i$) is defined based on the following hard thresholding function

$$\mathrm{z}_d^{(i)} = \max(0, \min(1, 0.5 + \mu_d^{(i)} + \epsilon_d^{(i)})), \tag{3}$$

where $\epsilon_d^{(i)}$ is drawn from $\mathcal{N}(0, \sigma^2)$ and $\sigma$ is fixed throughout training. The choice of $\sigma$ (which controls the injected noise) is discussed in Section B in the Appendix. The sample-specific parameters $\boldsymbol{\mu}^{(i)} \in \mathbb{R}^D, i = 1, ..., N$ are predicted based on a *gating* network $\boldsymbol{\Psi}$ such that $\boldsymbol{\mu}^{(i)} = \boldsymbol{\psi}(\boldsymbol{x}^{(i)}|\boldsymbol{\Omega})$, where $\boldsymbol{\Omega}$ are the weights of the *gating* network. These weights are learned simultaneously with the weights of the *prediction* network by minimizing the following loss:

$$\mathbb{E}\big[\mathcal{L}(\boldsymbol{f_\theta}(\boldsymbol{x}^{(i)} \odot \mathbf{z}^{(i)}), y^{(i)}) + \mathcal{R}(\mathbf{z}^{(i)})\big], \tag{4}$$

where $\mathcal{L}$ is a desired loss (e.g. cross entropy), and we compute its empirical expectation over $\boldsymbol{x}^{(i)}, y^{(i)}$ and $\mathbf{z}^{(i)}$, for $i$ in a batch of size $B$. The term $\mathcal{R}(\mathbf{z}^{(i)})$ is a regularizer that we define as

$$\mathcal{R}(\mathbf{z}^{(i)}) = \lambda_1 \|\mathbf{z}^{(i)}\|_0 + \lambda_2 \sum_j K_{i,j} \|\mathbf{z}^{(i)} - \mathbf{z}^{(j)}\|_2^2. \tag{5}$$

After taking the expectation (over $\mathbf{z}^{(i)}$), the leading term in $\mathcal{R}$ can be rewritten using a double sum in terms of the Gaussian error function (erf):

$$\frac{1}{N} \sum_{i=1}^{N} \sum_{d=1}^{D} \left( \frac{1}{2} - \frac{1}{2} \operatorname{erf}\left( -\frac{\mu_d^{(i)} + 0.5}{\sqrt{2}\sigma} \right) \right). \tag{6}$$

The second regularization term (in Eq. 5) is introduced to encourage **stability** of the local variable selection mechanism and is evaluated using Monte Carlo sampling. The kernel $K_{i,j} \geq 0$ is user defined (e.g. radial basis function) and should reflect the affinity between samples $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(j)}$, therefore, we can ensure that for nearby points our model would lead to similar sparsity patterns in $\mathbf{z}^{(i)}$ and $\mathbf{z}^{(j)}$. Altogether, Eq. 4 is optimized using SGD over the model parameters $\boldsymbol{\theta}$ and the parameters of the *gating* network $\boldsymbol{\Omega}$ (see Algorithm 1 in the Appendix for description

of training procedure). If $\mu_d^{(i)}$ of sample $i$ is a large number, then the $d$-th feature will be relevant for predicting $y^{(i)}$ with high probability (and vice versa for very small numbers). The stochasticity of the model plays two important roles: (1) it allows us to train weights of a binary model (the *gating* network). (2) it enables the model to re-evaluate features that are sparsified at an early step of training.

At inference, we remove the stochasticity from the gates and set $\hat{z}_d^{(i)} = \max(0, \min(1, 0.5 + \mu_d^{(i)}))$, which informs what features are selected. In practice, we observe that the coordinates of $\hat{z}_d^{(i)}$ mostly converge to 0 or 1 (see statistics in Table A.1 in the Appendix). This solution is encouraged as it is stable to the injected Gaussian noise ($\epsilon_d^{(i)}$). Namely, once $\mu_d^{(i)} = \{-1, 1\}$ (which are at the boundary of the range of the tanh activation used in the *gating* network), the value of the corresponding $\mathrm{z}_d^{(i)}$ would be with high probability 0 and 1, respectively. This is because the injected noise is less likely to push the values of $\mathrm{z}_d^{(i)}$ into the range $(0, 1)$.

## 4. Related Work

Identifying sample-specific subsets of variables that are important for prediction has been studied in the context of interpretability. Methods such as (Simonyan et al., 2013; Zeiler & Fergus, 2014; Lundberg & Lee, 2017) try to identify a small subset of features that explain the predictions made by a pre-trained model. These models either use the gradients of the pre-trained model or use perturbations to study the influence of different variables on the predictions of each instance. However, as shown in (Jethani et al., 2021), these models either require heavy post-training computations or are inaccurate (Adebayo et al., 2018; Gale et al., 2019). More recent works such as (Dabkowski & Gal, 2017; Chen et al., 2018; Schwab & Karlen, 2019; Yoon et al., 2018) alleviate the computational burden by training a single model to explain all samples. Still, they are all designed to explain pre-trained black-box models and thus cannot reduce the generalization gap in the case of LSS data.

Two recent works (Yoon et al., 2018; Jethani et al., 2021) present solutions which allow for training a prediction model in tandem with an explanatory model. However, both methods try to learn a model that "imitates" the predictions made by a baseline model, which uses the complete set of features. We argue that using the entire feature space in LSS data can lead to overfitting. Moreover, these methods require training a large number of parameters and use REINFORCE (Williams, 1992) or REBAR (Tucker et al., 2017) for learning the sparsity patterns. We demonstrate in the benchmark experiment (see Fig. A.12 in the Appendix) that the method proposed in (Yoon et al., 2018) is computationally expensive and does not scale well to large datasets. Furthermore, in Section 5 we provided extensive empirical evidence that our framework is more accurate and interpretable compared to (Yoon et al., 2018; Jethani et al., 2021).

Sparsification has also been utilized for other purposes, e.g., in Mixture of Experts (Peralta & Soto, 2014; Khalili, 2010; Pan & Shen, 2007; Shazeer et al., 2017; Riquelme et al., 2021), for accelerating the inference of over parametrized deep nets (Dong et al., 2017; Gao et al., 2018; Ashouri et al., 2019; Kurtz et al., 2020; Fedus et al., 2021), and genome-wide association studies (Demetci et al., 2021). In contrast to these models, our method sparsifies the number of features used for predictions of each instance, thus leading to a more robust and interpretable model.

## 5. Experiments

In this section, we evaluate how *accurate* and *interpretable* the proposed approach is on both synthetic and real-world biomedical datasets. We compare to: embedded feature selection methods such as LASSO (Tibshirani, 1996), linear support vector classification (SVC) (Chang & Lin, 2008), tree-based wrapper methods such as Random Forest (RF) (Díaz-Uriarte & De Andres, 2006) and XGBoost (Chen & Guestrin, 2016), fully connected neural network (with no convolution, distortion, weight decay, or unsupervised pre-training) and other neural network-based feature selection and interpretability methods such as STG (Yamada et al., 2020), INVASE (Yoon et al., 2018), L2X (Chen et al., 2018), TabNet (Arık & Pfister, 2020), and REAL-x (Jethani et al., 2021). Additionally we compare to RANSAC (Fischler & Bolles, 1981) and Localized LASSO (Yamada et al., 2017) for the motivating example, both RF and NN with SHAP (Lundberg & Lee, 2017; Shrikumar et al., 2017) for the MNIST example, and DeepSurv (Katzman et al., 2018), COX-LASSO, COX-STG (Yamada et al., 2020), and Random Survival Forest (Ishwaran et al., 2008) for the survival analysis result. We failed to compare to Localized LASSO, since is not suited for classification and it did not converge on the regression examples. Details of the datasets availability, training procedure, and hyper-parameter tuning are included in Appendix section B.

### 5.1. Nonlinear Prediction on Synthetic Datasets

This section uses synthetic datasets where the target value only depends on a subset of variables that varies across samples. Since the per sample subset of informative variables is known, we can perform a controlled evaluation of the predictivity and interpretability of our model. First, we focus on classification data models **E1**-**E3** which were also used for evaluation in (Yoon et al., 2018; Jethani et al., 2021; Arık & Pfister, 2020). We further design a higher dimensional example **E4**, and a highly nonlinear "moving-XOR" regression example **E5**. In all datasets, we use less than 2000 samples

| | E1 | | E2 | | E3 | | E4 | | E5 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | F1 | ACC | F1 | ACC | F1 | ACC | F1 | ACC | F1 | MSE |
| LASSO | 0.5000 | 52.00 | 0.5000 | 74.50 | 0.6250 | 71.50 | 0.1290 | 64.00 | 0.3704 | 1.0190 |
| SVC | 0.4444 | 51.50 | 0.5000 | 74.50 | 0.6667 | 71.50 | 0.2353 | 68.00 | NA | NA |
| RF | 0.5333 | 88.50 | 0.5333 | 88.50 | 0.6250 | 87.00 | 0.0769 | 86.00 | 0.2500 | 0.2499 |
| XGBoost | 0.5333 | 93.00 | 0.5333 | **95.00** | 0.6250 | 86.50 | 0.0769 | 96.00 | 0.2500 | 0.0118 |
| MLP | NA | 78.00 | NA | 88.50 | NA | 84.50 | NA | 64.00 | NA | 0.6526 |
| Linear STG | 0.4000 | 55.50 | 0.4000 | 76.00 | 0.3750 | 69.00 | 0.6667 | 70.00 | 0.5000 | 1.0067 |
| Nonlinear STG | 0.7272 | 84.50 | 0.7272 | 90.00 | 0.7143 | 86.00 | 0.6667 | 76.00 | 0.7500 | **0.0004** |
| INVASE | 0.5390 | 89.00 | 0.7000 | 88.00 | 0.6923 | 86.00 | 0.6667 | 94.00 | 0.1526 | 3.1264 |
| L2X | 0.7986 | 88.00 | 0.6050 | 94.50 | 0.2450 | 87.00 | 0.5000 | 92.00 | 0.6081 | 0.5134 |
| TabNet | 0.4789 | 54.50 | 0.5426 | 65.50 | 0.6905 | 78.50 | 0.0036 | 60.00 | 0.4454 | 1.0317 |
| REAL-x | 0.8306 | 85.00 | 0.7089 | 88.50 | 0.7823 | 86.00 | 0.8511 | 90.00 | NA* | NA* |
| LLSPIN | 0.3337 | 80.50 | 0.7216 | 86.50 | 0.4741 | 73.50 | 0.9458 | 90.00 | 0.6815 | 0.4927 |
| LSPIN | **0.9761** | **94.00** | **0.8600** | **95.00** | **0.9296** | **89.00** | **0.9615** | **98.00** | **1** | 0.0019 |

Table 1: Nonlinear synthetic datasets (see Eqs. 9-13 in Appendix section B.4). We compare the proposed LLSPIN/LSPIN to other baselines in terms of the F1 score of the selected features and accuracy(%)/ MSE for the prediction performance. Across these examples, LSPIN correctly identifies the informative features with a substantially higher F1 and higher accuracy/lower mse. *We attempted to implement REAL-x for the regression task, but the model failed to converge.

for training, a regime which is more challenging than what was previously studied by (Yoon et al., 2018; Jethani et al., 2021; Arık & Pfister, 2020). The exact data models and training procedures are described in the Appendix section B.4. We evaluate all models by measuring the F1 score of the selected features (F1 $= \frac{\text{TP}}{\text{TP}+\frac{1}{2}(\text{FP}+\text{FN})}$ where TP is the number of informative features that are selected by the model, FP is the number of selected features that are uninformative, and FN is the number of informative features unrecovered by the model) and the prediction performance (accuracy for classification and Mean Squared Error (MSE) for regression).

As shown in Table 1, LSPIN consistently outperforms existing baselines in terms of its ability to identify the informative variables (evaluated using the F1 score). At the same time, our model leads to improved predictive capabilities compared to these baselines. Our linear model (LLSPIN) leads to relatively high F1 score in **E2**, **E4**, **E5** despite the fact that the data contains nonlinear feature interactions.

### 5.2. Interpretability Evaluation on MNIST

In this section, we demonstrate that the proposed method is exceptionally *interpretable* while leading to predictions that are more *accurate* than state-of-the-art nonlinear models. To evaluate interpretability, previous authors suggest the following criteria:

**Faithfulness:** Are the identified features significant for prediction?

**Stability:** Are explanations to similar samples consistent?

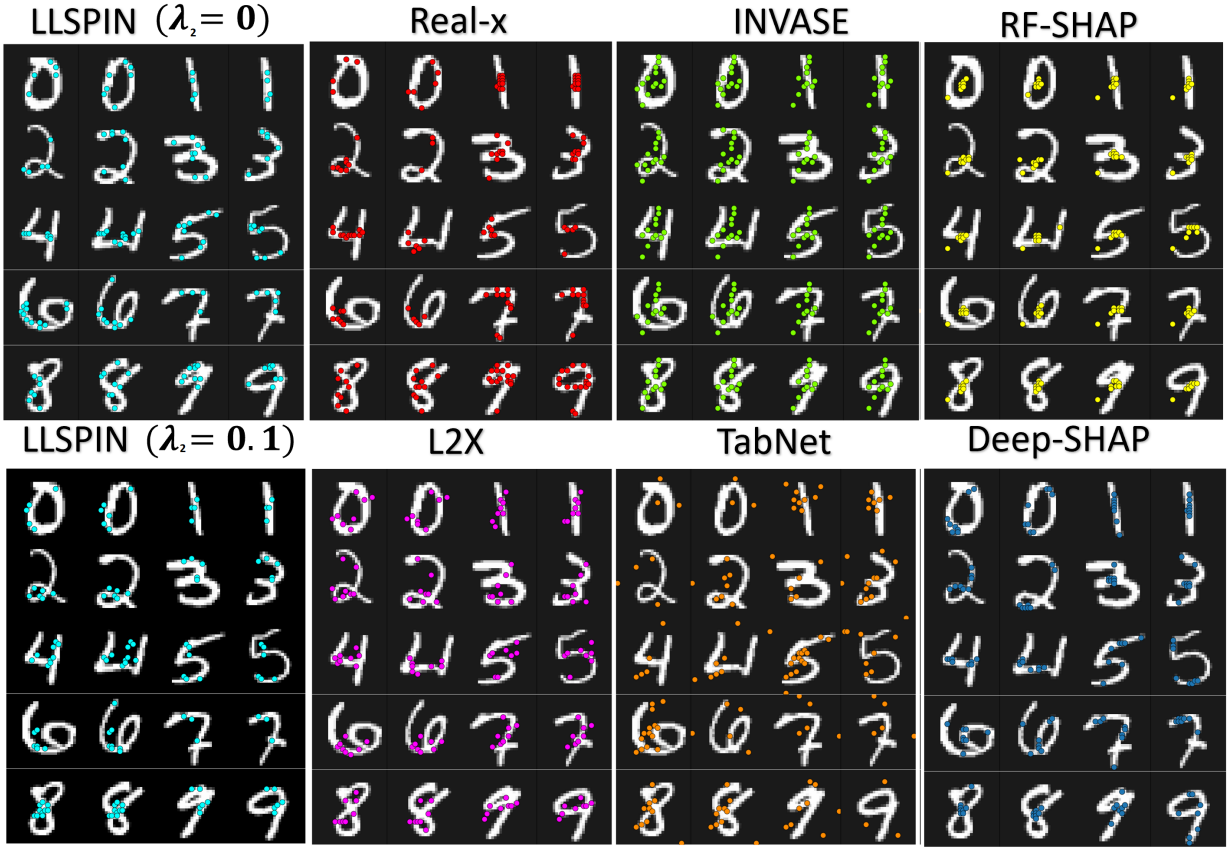
**Diversity:** How different are the selected variables for instances of distinct classes?

**Generalizability:** Are the selected features beneficial for making accurate predictions using other simple models?

We use MNIST handwritten dataset as a table with 784 features and do not consider spatial information (since we are interested in tabular data). We compare our model to both RF and NN with SHAP and other leading NN interpretability models, including L2X, REAL-x, INVASE, and TabNet.

As suggested by (Alvarez-Melis & Jaakkola, 2018), **faithfulness** could be evaluated by removing features one by one (based on their importance) and calculating the correlation between the predictivity drop and the feature importance. Following (Alvarez-Melis & Jaakkola, 2018; Yoshikawa & Iwata, 2020), **stability** is evaluated by computing the Lipchitz constant of the explanation function. This is estimated for $\boldsymbol{x}_i$ using $\max_{\boldsymbol{x}_i,\boldsymbol{x}_k \leq \epsilon} \frac{\|\boldsymbol{w}_i - \boldsymbol{w}_k\|_2}{\|\boldsymbol{x}_i - \boldsymbol{x}_k\|_2}$, where $\boldsymbol{w}_i$ is the explanation vector for sample $i$ provided by each method. Then we average the Lipchitz constant over all samples. To evaluate **diversity** we take the per class median selected features and use the Jaccard index to count the portion of non-overlapping selected features across classes (exact formula appears in Appendix A.7). To evaluate the **generalizability** of the selected features, we measure the accuracies of SVM and $k$-means when applied to the data, which is masked by the selected features. We expect the performance to be preserved if the selected features are crucial for prediction (compared to the accuracy obtained when using all features).

We tune all models to identify the $\sim 10$ most informative pixels per image and present these in Fig. 3. As visually indicated by this figure, LLSPIN, Real-X, L2X, and DeepSHAP tend to select pixels with non-zero values that cover "unique" patterns in the digits. INVASE seems to lead to a




| Method | ACC | # Feat | Stability | Diversity | Faithfulness | Gen-SVM | Gen-$k$-means |
|---|---|---|---|---|---|---|---|
| RF-SHAP | 96.81 | 8 | 0.418 | 41.15 | 0.531 | 52.42 | 40.15 |
| Deep-SHAP | 97.86 | 8 | 0.634 | 88.10 | 0.780 | 83.56 | 56.31 |
| REAL-x | 96.95 | 10 | 0.415 | 80.55 | 0.885 | 94.04 | 87.94 |
| L2X | 89.11 | 8 | 0.268 | 94.79 | 0.791 | 94.18 | 89.56 |
| INVASE | 85.07 | 11 | 0.162 | 13.43 | 0.864 | 69.67 | 43.02 |
| TabNet | 96.79 | 6 | 0.265 | 89.17 | 0.759 | 54.42 | 43.22 |
| LLSPIN ($\lambda_2 = 0$) | 98.26 | 7 | 0.294 | 99.14 | 0.950 | 98.22 | 97.50 |
| LLSPIN ($\lambda_2 = 0.1$) | 98.18 | 6 | 0.098 | 99.05 | 0.926 | 96.23 | 96.84 |
| LSPIN ($\lambda_2 = 0$) | **98.45** | 7 | 0.256 | **99.84** | **0.987** | **98.43** | **97.99** |
| LSPIN ($\lambda_2 = 0.1$) | 98.29 | 6 | **0.065** | 99.39 | 0.917 | 98.42 | 97.57 |

Figure 3: Top: Random samples from the MNIST dataset. For each example, we overlay the image with color dots indicating locations of the estimated informative pixels by each method. Bottom: Test accuracy (ACC), the median number of features selected by each the model (# Feat), and several metrics for evaluating interpretability capabilities (explained in Section 5.2). Our model leads to high classification accuracy while relying on a few faithful and generalizable features. The selected features by our model are diverse across different classes while remaining stable for nearby samples. We compare our models with and without the second regularization term ($\lambda_2$ in Eq. 5), and demonstrate that it improves stability.

more global selection; RF-SHAP concentrates on one region per digit, and TabNet selects several non-active pixels. Since TabNet still leads to high classification accuracy, we suspect that it encodes the prediction within the binary interpretation vector as an input to the prediction model (as suggested in (Jethani et al., 2021)).

Next, we use the interpretability metrics described above to compare all baselines. As indicated in the table (see bottom panel of Fig. 3), both LSPIN and LLSPIN perform exceptionally well in terms of accuracy and interpretability. We compare our models with and without the second regularization term (see $\lambda_2$ in Eq. 5). Our results suggest that including this term improves the stability of the selected features without a significant compromise for other qualities. The results highlight three encouraging findings: (i) our linear model leads to an accuracy close to CNN level (which is 98.9%). (ii) applying $k$-means to the data gates by the selected features leads to a dramatic improvement

in clustering accuracy. Namely, it improves from $\sim 55\%$ to an accuracy higher than $96\%$ when using the features selected by LLSPIN or LSPIN. (iii) our model improves robustness and uncertainty estimates (Ovadia et al., 2019) under distributional shifts compared with a standard fully connected network (see results in Appendix A.12).

### 5.3. Classification of LSS Real World Data

In this section, we evaluate LLSPIN and LSPIN on several challenging LSS real-world datasets (properties are summarized at the bottom of Table 2). BASEHOCK, RELATHE, PCMAC, COLON, TOX171 are from the feature selection dataset collections[1], and the purified PBMC dataset is from (Zheng et al., 2017).

For the BASEHOCK, RELATHE, PCMAC, and PBMC datasets, $5\%$ of each dataset is set aside as a validation set. We split the remaining $95\%$ of the data into 5 non-overlapping folds, with 1 fold for training and the remaining folds for testing each time (see details in Appendix section B.5). This procedure is repeated for several regularization parameters; then, we report the best average performance (test accuracy) for each method, along with the corresponding standard deviation and the number of selected features. Since COLON and TOX-171 are of extreme LSS, we use a grid of regularization parameters for each method and identify the best average performance across ten runs (using $80\%$ of the samples for training and $20\%$ for testing).

In Table 2, we present the average test accuracy, standard deviation, and number of selected features for all baselines. As evident across several datasets, our framework dramatically improves the accuracy compared to standard MLP while using a small portion of the input set of variables. Moreover, in most cases, our model outperforms state-of-the-art models such as XGBoost, TabNet, and REAL-x. We rank the methods based on the average classification accuracy for each dataset. Our models (LLSPIN/LSPIN) reach the top 2 places based on the median rank across all datasets.

In these examples, to our surprise, LLSPIN outperforms LSPIN. We reason that LLSPIN (the linear version of our model) remains highly expressive in the high dimensional setting since it learns several linear relationships, each based on a small set of coefficients. Moreover, since this prediction model does not contain any nonlinearity, overfitting is less likely to happen than in the nonlinear models. These results suggest that LLSPIN can serve as an accurate and highly interpretable model in LSS data regimes. In Appendix A.12 we evaluate our models predictive *uncertainty* using the Negative Log Likelihood (NLL) and demonstrate that it leads to more calibrated uncertainty estimates (Ovadia et al., 2019) compared with other baselines.

[1] https://jundongl.github.io/scikit-feature/datasets.html

### 5.4. Survival Analysis

Survival Analysis involves predicting the survival time of individual patients based on different clinical variables. In Survival Analysis, instance-level interpretation of the selected features is of particular interest as it can answer what variables have the most significant effect on the survival of individual patients. We integrate our models (LSPIN/LLSPIN) into DeepSurv (Katzman et al., 2018), which is a neural network framework for Cox regression. Then, we apply the integrated models (COX-LLSPIN/COX-LSPIN) on a Surveillance, Epidemiology, and End Results (SEER) breast cancer dataset [2] to perform survival analysis.

We evaluate the performance of our models by computing test Concordance Index (C-Index) w.r.t. the number of selected features. We compare the performance with other Survival Analysis models as shown in Fig. 4 (Left). We can see that the COX-LLSPIN/COX-LSPIN is comparable to state-of-the-art schemes when using more than 28 features and outperforms all existing methods when focusing on small subsets of selected features. More importantly, our model (COX-LLSPIN) can provide more interpretable results while remaining accurate.

Fig. 4 (Right) shows the frequency of the selected feature sets among the different subjects (samples). For instance, $57.2\%$ of the samples have Age and TsizeMerged selected as important factors for the prediction. In contrast, $1.2\%$ of the samples have only NodesRatio selected, demonstrating that our models can characterize the heterogeneity among these samples. We argue that this is an important property for practitioners since knowing what variables affect each patient's outcome can improve personalized treatments.

### 5.5. Marker Gene Identification

Accurate cell classification is imperative for the success of many single-cell genomics studies. Developing an automated way to identify genes that allow identification of cell types (marker genes) is an ongoing challenge (Dai et al., 2021). Here, we apply our model to a Single Nucleus RNA-sequencing dataset. The cell types in this data, namely Microglia and Oligodendrocyte Precursor Cells, are well characterized by *ITGAM* gene and *PDGFRA* gene, respectively. We aim to apply our model to identify these markers for each cell type automatically. Details of the data preprocessing and split are in Appendix section B.7.

Towards this goal, we aim to encourage our model to select a diverse set of features for each class. Therefore, we modify the second term in our regularizer (see Eq. 5) to $\lambda_2 \sum_j (1 - K_{i,j}) \times (-\|\mathbf{z}^{(i)} - \mathbf{z}^{(j)}\|_2^2)$. Intuitively, when sample $i$ and $j$ are dissimilar ($K_{i,j}$ is small), the corresponding gates are

[2] www.seer.cancer.gov

| | BASEHOCK | RELATHE | PCMAC | PBMC | COLON | TOX-171 | Median Rank |
|---|---|---|---|---|---|---|---|
| LASSO | $74.46 \pm 5.19$ [34] | $58.69 \pm 1.59$ [18] | $68.09 \pm 4.08$ [21] | $90.30 \pm 0.36$ [31] | $81.54 \pm 9.85$ [24] | $87.71 \pm 4.62$ [49] | 6.5 |
| SVC | $74.46 \pm 3.37$ [22] | $56.48 \pm 3.00$ [6] | $67.41 \pm 3.72$ [12] | $89.02 \pm 0.74$ [30] | $76.15 \pm 9.39$ [25] | $81.14 \pm 7.47$ [38] | 8.5 |
| RF | $64.46 \pm 4.52$ [10] | $71.42 \pm 3.50$ [50] | $67.44 \pm 7.00$ [9] | $48.56 \pm 6.18$ [10] | $79.23 \pm 9.76$ [47] | $53.71 \pm 9.96$ [42] | 11.5 |
| XGBoost | $90.37 \pm 1.05$ [45] | $76.75 \pm 1.67$ [32] | $83.93 \pm 0.67$ [43] | $76.58 \pm 0.72$ [64] | $76.15 \pm 12.14$ [7] | $67.43 \pm 5.60$ [38] | 6 |
| MLP | $56.51 \pm 1.43$ | $55.44 \pm 2.38$ | $54.38 \pm 1.27$ | $61.57 \pm 1.45$ | $81.54 \pm 7.84$ | $62.59 \pm 8.03$ | 12.5 |
| Linear STG | $89.36 \pm 1.40$ [27] | $69.94 \pm 5.05$ [16] | $\mathbf{85.11 \pm 1.07}$ [42] | $88.22 \pm 0.82$ [27] | $74.62 \pm 11.44$ [14] | $71.14 \pm 5.78$ [16] | 7 |
| Nonlinear STG | $89.24 \pm 1.18$ [20] | $74.83 \pm 3.95$ [27] | $84.16 \pm 0.90$ [32] | $86.29 \pm 1.31$ [19] | $76.15 \pm 13.95$ [8] | $67.43 \pm 7.25$ [14] | 6.5 |
| INVASE | $84.02 \pm 0.81$ [42] | $70.81 \pm 1.56$ [43] | $77.06 \pm 1.01$ [48] | $86.34 \pm 0.81$ [30] | $76.92 \pm 12.40$ [6] | $76.86 \pm 7.39$[26] | 7.5 |
| L2X | $88.48 \pm 2.01$ [1] | $77.10 \pm 5.19$ [10] | $78.69 \pm 3.62$ [10] | $70.77 \pm 11.24$ [10] | $78.46 \pm 8.28$ [8] | $71.71 \pm 10.42$[9] | 6.5 |
| TabNet | $88.21 \pm 2.00$ [3] | $67.84 \pm 15.40$ [10] | $69.35 \pm 10.49$ [4] | $\mathbf{92.13 \pm 0.59}$ [3] | $64.62 \pm 12.02$ [28] | $30.00 \pm 6.29$ [34] | 9.5 |
| REAL-x | $89.80 \pm 1.96$ [5] | $80.61 \pm 1.31$ [3] | $80.98 \pm 3.05$ [6] | $83.39 \pm 2.19$ [24] | $75.38 \pm 12.78$ [15] | $77.71 \pm 7.65$ [42] | 5 |
| LSPIN | $89.37 \pm 1.48$ [3] | $80.59 \pm 1.95$ [3] | $78.51 \pm 1.48$ [3] | $88.67 \pm 0.64$ [15] | $71.54 \pm 6.92$ [1] | $90.29 \pm 5.45$[1] | 4.5 |
| LLSPIN | $\mathbf{91.56 \pm 1.51}$ [4] | $\mathbf{82.01 \pm 2.20}$ [11] | $81.48 \pm 1.74$ [3] | $90.43 \pm 0.6$ [18] | $\mathbf{83.85 \pm 5.38}$ [7] | $\mathbf{92.57 \pm 6.41}$ [6] | **1** |
| Train / Test | 379 / 1514 | 271 / 1084 | 369 / 1476 | 721 / 2880 | 49 / 13 | 136 / 35 | |
| Dim/ Classes | 4862 / 2 | 4322 / 2 | 3289 / 2 | 2000 / 4 | 2000 / 2 | 5748 / 4 | |
| Type | Text | Text | Text | Biomedical | Biomedical | Biomedical | |

Table 2: Classification on real-world tabular datasets. We report the average accuracy and standard deviation, with the corresponding median number of selected features in square brackets. The number of training/test samples, dimensions, classes, and data types are also reported.

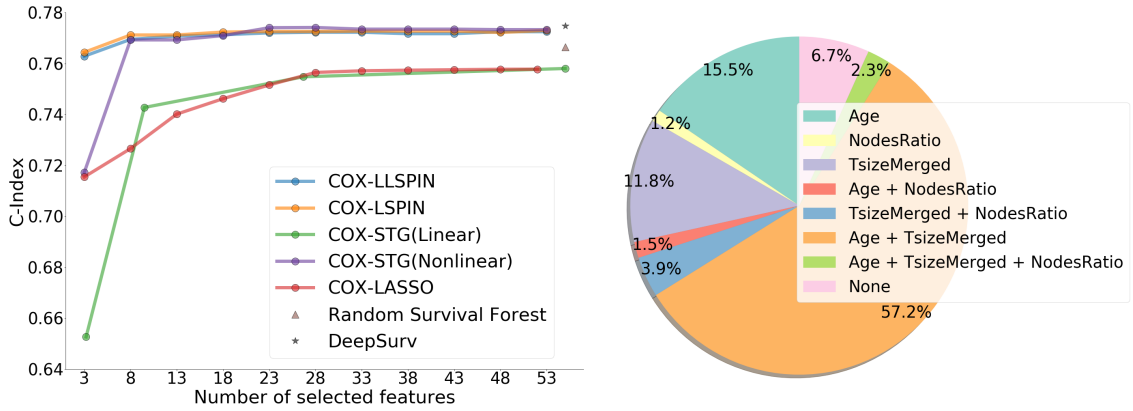



Figure 4: Survival analysis based on the SEER breast cancer data. Left: Comparing the test C-Index obtained using subsets of most informative features. Right: Frequency of variables selected by COX-LLSPIN across the different patients.

encouraged to be different. Here, we fix $\lambda_2 = 1$.

LLSPIN successfully identifies the two cell-type-specific markers (see Appendix Fig. A.11) while predicting the correct cell with $99.0\%$ accuracy. We further evaluate other instance-wise feature selection methods on this example. As indicated by the F1 score of the selected genes in Table 3 our approach significantly outperforms other schemes in its ability to identify the marker genes correctly.

| | LLSPIN | INVASE | L2X | TabNet | REAL-x |
|---|---|---|---|---|---|
| F1 | **0.9950** | 0.4900 | 0.4900 | 0.2817 | 0.5000 |

Table 3: Marker gene identification using several baselines. LLSPIN accurately identifies the known marker genes reflected by the F1 score computed based on selected features.

## 6. Conclusion

We present a NN framework for making *accurate* and *interpretable* predictions based on tabular biomedical datasets. To achieve these goals, we design a special kind of sample-specific regularizer that leads to sparsification that is *stable* for similar samples. Our regularizer is parametrized using a *gating* network that is trained simultaneously with a *prediction* network and learns for each sample the set of most informative features. This leads to an intrinsically interpretable model, which can handle cases of low-sample-size (LSS) data that is either high dimensional or contains nuisance features. We demonstrate using synthetic and real datasets that our model can outperform state-of-the-art classification and regression models. Furthermore, when applied to datasets with nuisance variables, our model correctly identifies the subsets of informative features.

## Acknowledgements

The authors thank Mihir Khunte and Michal Marczyk for the preprocessing steps of the SEER breast cancer data, Le Zhang, Serena Spudich, and Mark Gerstein for providing the snRNA-seq data. Y.K. acknowledges support by NIH grant R01GM131642, UM1DA051410, U54AG076043, P50CA121974, and U01DA053628.

## References


Abid, A., Balin, M. F., and Zou, J. Y. Concrete autoencoders for differentiable feature selection and reconstruction. *CoRR*, abs/1901.09346, 2019.

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. Sanity checks for saliency maps. *arXiv preprint arXiv:1810.03292*, 2018.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. *KDD*, 2019.

Allenby, G. M. and Rossi, P. E. Hierarchical bayes models. *The handbook of marketing research: Uses, misuses, and future advances*, pp. 418–440, 2006.

Alvarez-Melis, D. and Jaakkola, T. S. Towards robust interpretability with self-explaining neural networks. *arXiv preprint arXiv:1806.07538*, 2018.

Aoshima, M., Shen, D., Shen, H., Yata, K., Zhou, Y.-H., and Marron, J. A survey of high dimension low sample size asymptotics. *Australian & New Zealand journal of statistics*, 60(1):4–19, 2018.

Arık, S. O. and Pfister, T. Tabnet: Attentive interpretable tabular learning. *arXiv*, 2020.

Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*, 2016.

Ashouri, A. H., Abdelrahman, T. S., and Dos Remedios, A. Retraining-free methods for fast on-the-fly pruning of convolutional neural networks. *Neurocomputing*, 370: 56–69, 2019.

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., and Kasneci, G. Deep neural networks and tabular data: A survey. *arXiv preprint arXiv:2110.01889*, 2021.

Chang, Y.-W. and Lin, C.-J. Feature ranking using linear svm. In *Causation and Prediction Challenge*, pp. 53–64, 2008.

Chen, J., Song, L., Wainwright, M., and Jordan, M. Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning*, pp. 883–892. PMLR, 2018.

Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794. ACM, 2016.

Dabkowski, P. and Gal, Y. Real time image saliency for black box classifiers. *arXiv preprint arXiv:1705.07857*, 2017.

Dai, M., Pei, X., and Wang, X.-J. Accurate and fast cell marker gene identification with cosg. *bioRxiv*, 2021.

Demetci, P., Cheng, W., Darnell, G., Zhou, X., Ramachandran, S., and Crawford, L. Multi-scale inference of genetic trait architecture using biologically annotated neural networks. *PLoS genetics*, 17(8):e1009754, 2021.

Díaz-Uriarte, R. and De Andres, S. A. Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, 7(1):3, 2006.

Dong, X., Huang, J., Yang, Y., and Yan, S. More is less: A more complicated network with less inference complexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5840–5848, 2017.

Fan, J. and Li, R. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021.

Feng, J. and Simon, N. Sparse-Input Neural Networks for High-dimensional Nonparametric Regression and Classification. *ArXiv e-prints*, November 2017.

Fischler, M. A. and Bolles, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

Gao, X., Zhao, Y., Dudziak, Ł., Mullins, R., and Xu, C.-z. Dynamic channel pruning: Feature boosting and suppression. *arXiv preprint arXiv:1810.05331*, 2018.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Ishwaran, H., Kogalur, U., Blackstone, E., and Lauer, M. Random survival forests. *Annals of Applied Statistics*, 2 (3):841–860, 9 2008. ISSN 1932-6157. doi: 10.1214/08-AOAS169.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Jethani, N., Sudarshan, M., Aphinyanaphongs, Y., and Ranganath, R. Have we learned to explain?: How interpretability methods can learn to encode predictions in their interpretations. In *International Conference on Artificial Intelligence and Statistics*, pp. 1459–1467. PMLR, 2021.

Katzman, J. L., Shaham, U., Cloninger, A., Bates, J., Jiang, T., and Kluger, Y. Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC Medical Research Methodology*, 18, 2018.

Khalili, A. New estimation and feature selection methods in mixture-of-experts models. *Canadian Journal of Statistics*, 38(4):519–539, 2010.

Kurtz, M., Kopinsky, J., Gelashvili, R., Matveev, A., Carr, J., Goin, M., Leiserson, W., Moore, S., Nell, B., Shavit, N., et al. Inducing and exploiting activation sparsity for fast neural network inference. In *37th International Conference on Machine Learning, ICML 2020*, volume 119, 2020.

Li, Y., Chen, C.-Y., and Wasserman, W. W. Deep feature selection: theory and application to identify enhancers and promoters. *Journal of Computational Biology*, 23(5): 322–336, 2016.

Lindenbaum, O. and Steinerberger, S. Randomly aggregated least squares for support recovery. *Signal Processing*, 180: 107858, 2021a.

Lindenbaum, O. and Steinerberger, S. Refined least squares for support recovery. *arXiv preprint arXiv:2103.10949*, 2021b.

Linderman, G. C., Zhao, J., Roulis, M., Bielecki, P., Flavell, R. A., Nadler, B., and Kluger, Y. Zero-preserving imputation of single-cell rna-seq data. *Nature Communications*, 13(1):1–11, 2022.

Liu, B., Wei, Y., Zhang, Y., and Yang, Q. Deep neural networks for high dimension, low sample size data. In *IJCAI*, pp. 2287–2293, 2017.

Louizos, C., Welling, M., and Kingma, D. P. Learning sparse neural networks through l0 regularization. *CoRR*, abs/1712.01312, 2017.

Lundberg, S. and Lee, S.-I. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*, 2017.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Marais, J. A. *Deep learning for tabular data: an exploratory study*. PhD thesis, Stellenbosch: Stellenbosch University, 2019.

National Cancer Institute. Surveillance, epidemiology, and end results (SEER) program research data (1975–2016). In National Cancer Institute, DCCPS, Surveillance Research Program, Surveillance Systems Branch, released April 2019, based on the November 2018 submission. SEER. (www.seer.cancer.gov)., 2019.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J. V., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530*, 2019.

Pan, W. and Shen, X. Penalized model-based clustering with application to variable selection. *Journal of machine learning research*, 8(5), 2007.

Peralta, B. and Soto, A. Embedded local feature selection within mixture of experts. *Information Sciences*, 269: 176–187, 2014.

Riquelme, C., Puigcerver, J., Mustafa, B., Neumann, M., Jenatton, R., Pinto, A. S., Keysers, D., and Houlsby, N. Scaling vision with sparse mixture of experts. *arXiv preprint arXiv:2106.05974*, 2021.

Scardapane, S., Comminiello, D., Hussain, A., and Uncini, A. Group sparse regularization for deep neural networks. *Neurocomput.*, 241(C):81–89, June 2017. ISSN 0925-2312. doi: 10.1016/j.neucom.2017.02.029.

Schwab, P. and Karlen, W. Cxplain: Causal explanations for model interpretation under uncertainty. *arXiv preprint arXiv:1910.12336*, 2019.

Shavitt, I. and Segal, E. Regularization learning networks: deep learning for tabular datasets. *arXiv preprint arXiv:1805.06440*, 2018.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Shiffrin, R. M., Lee, M. D., Kim, W., and Wagenmakers, E.-J. A survey of model evaluation approaches with a tutorial on hierarchical bayesian methods. *Cognitive Science*, 32 (8):1248–1284, 2008.

Shrikumar, A., Greenside, P., and Kundaje, A. Learning important features through propagating activation differences. In *International conference on machine learning*, pp. 3145–3153. PMLR, 2017.

Shwartz-Ziv, R. and Armon, A. Tabular data: Deep learning is not all you need. *arXiv preprint arXiv:2106.03253*, 2021.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Soudry, D., Hoffer, E., Nacson, M. S., Gunasekar, S., and Srebro, N. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

Tishby, N. and Zaslavsky, N. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5. IEEE, 2015.

Tucker, G., Mnih, A., Maddison, C. J., Lawson, D., and Sohl-Dickstein, J. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. *arXiv preprint arXiv:1703.07370*, 2017.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Yaguchi, A., Suzuki, T., Asano, W., Nitta, S., Sakata, Y., and Tanizawa, A. Adam induces implicit weight sparsity in rectifier neural networks. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 318–325. IEEE, 2018.

Yamada, M., Koh, T., Iwata, T., Shawe-Taylor, J., and Kaski, S. Localized lasso for high-dimensional regression. In *Artificial Intelligence and Statistics*, pp. 325–333. PMLR, 2017.

Yamada, Y., Lindenbaum, O., Negahban, S., and Kluger, Y. Feature selection using stochastic gates. In *International Conference on Machine Learning*, pp. 10648–10659. PMLR, 2020.

Yoon, J., Jordon, J., and van der Schaar, M. Invase: Instance-wise variable selection using neural networks. In *International Conference on Learning Representations*, 2018.

Yoshikawa, Y. and Iwata, T. Gaussian process regression with local explanation. *arXiv preprint arXiv:2007.01669*, 2020.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Zheng, G. X., Terry, J. M., Belgrader, P., Ryvkin, P., Bent, Z. W., Wilson, R., Ziraldo, S. B., Wheeler, T. D., McDermott, G. P., Zhu, J., et al. Massively parallel digital transcriptional profiling of single cells. *Nature communications*, 8:14049, 2017.

# A. Additional Results

In the following sections, we provide additional experimental results to support the effectiveness of the proposed approach.

## A.1. Visualization of the Motivating Example and Extended Evaluations

For the motivating example in section 2, we used a training set with only 10 samples. As shown in Fig. A.1, LLSPIN correctly identifies the sample-specific features.

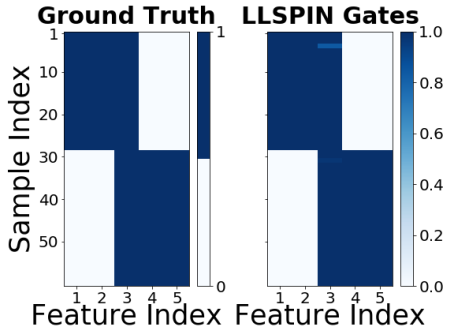



Figure A.1: Heat-map comparison between the ground truth informative features for each sample (Left, 1 for truly explanatory and 0 for not) and the identified features by the proposed LLSPIN's gates (Right, $> 0$ for open gates and 0 for closed). The values across the x-axis correspond to the feature indices, and the values across the y-axis correspond to the sample indices. Samples are sorted based on their ground truth groups.

Next, we extend our evaluation to other numbers of training samples (60,30,18,12,6) to explore the bound of N for which our model can help with performance improvement. Fig. A.2a demonstrates that LLSPIN consistently outperforms other methods. Fig. A.2b reveals that in each case, LLSPIN correctly uncovers the corresponding interpretable features for each sample in the gate matrices compared to the ground truth, except when the training is limited with just 6 training samples where LLSPIN misses one feature for the second sample group. This simulation demonstrates the effectiveness of LLSPIN on low sample size (LSS) datasets. Details of the data model, data split, and hyper-parameter tuning are in the Appendix section B.3.

## A.2. Additional Experiment Involving Linear Synthetic Dataset with Unequal Regression Coefficients

To further demonstrate our approach's applicability in a more challenging setting, we modified the linear synthetic dataset (Eq. 1 in the main text). Specifically, the coefficient of $\boldsymbol{x}_3$ in group 2 is set to $0.5$ instead of $-0.5$, while the coefficient of $\boldsymbol{x}_3$ in group 1 remains $-0.5$, as shown below:

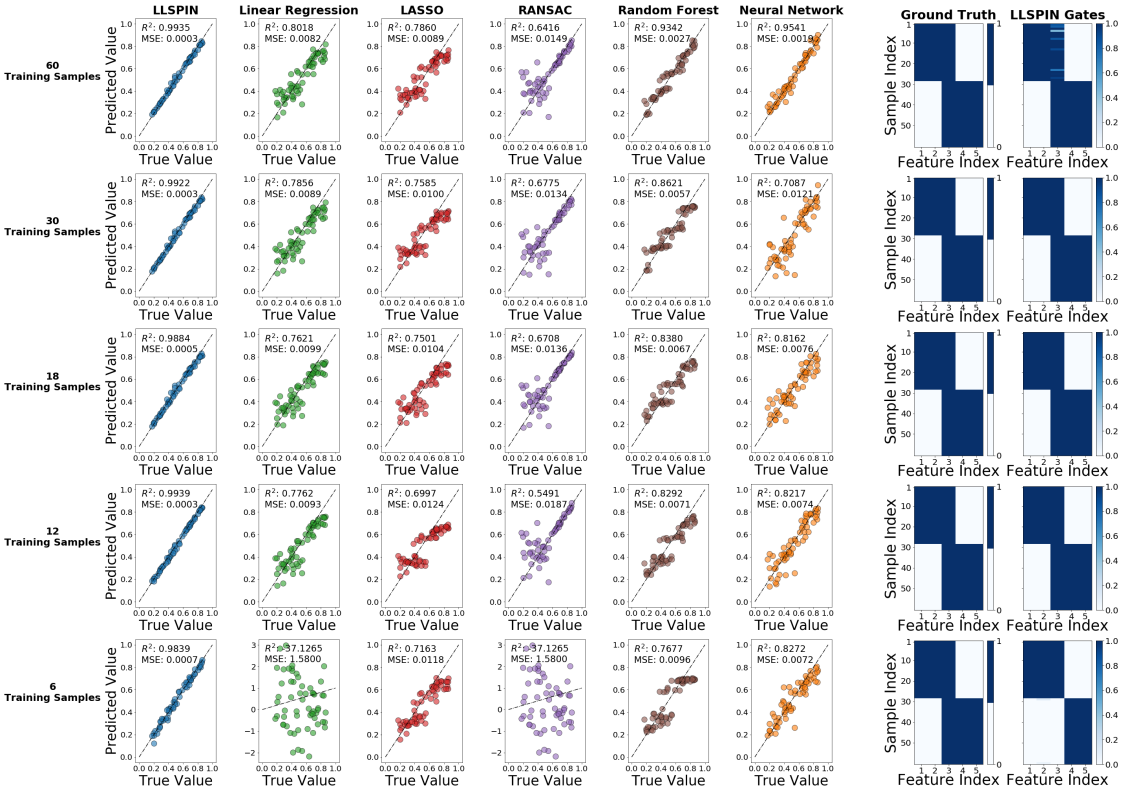
$$\mathrm{y} = \begin{cases} -2\boldsymbol{x}_1 + \boldsymbol{x}_2 - 0.5\boldsymbol{x}_3, & \text{if in group 1,} \\ 0.5\boldsymbol{x}_3 + \boldsymbol{x}_4 - 2\boldsymbol{x}_5, & \text{otherwise.} \end{cases} \tag{7}$$

We note that to learn this more complex regression function (due to the unequal feature coefficients), we had to change the prediction model to a fully connected deep network with nonlinear activation (since the regression function is no longer linear). We applied our nonlinear model (LSPIN) on this example and obtained a $100\%$ true positive rate and $0\%$ false discovery rate in terms of discovering the correct features on the test set. The mean square error on the test is $0.000199$.

## A.3. Nonlinear Synthetic Datasets

In the following subsection, we first demonstrate LSPIN's interpretability capabilities. We examine the sparsification patterns of LSPIN on the test sets for the 5 nonlinear synthetic examples (see Section 5.1 and the Appendix section B.4 for details), as shown in Fig. A.3 where LSPIN correctly identifies the informative features in most of the examples.

Next, we aim to demonstrate the applicability of our approach in a scenario with overlapping features; we experimented




(a) (b)

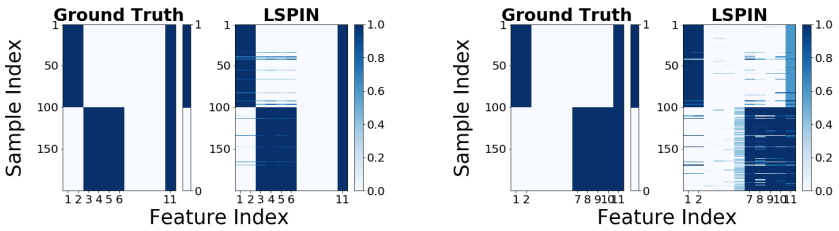
Figure A.2: (a) Evaluation of the performance for different training sizes. Each row indicates a different number of training samples, from top to bottom: 60, 30, 18, 12, and 6, respectively. The x-axis represents the true target value $y$, and the y-axis presents the predicted value $\hat{y}$ for each model (as indicated in the subtitles of the columns). Points on the diagonal line indicate correct predictions. R-square and Mean Squared Error (MSE) are reported for each model.
(b) Heat-map comparison between the ground truth informative features for each sample (Left, 1 for truly informative and 0 for not) and the identified features by LLSPIN's local gates (Right, $> 0$ for open gates and 0 for closed gates). The values across the x-axis correspond to the feature indices, and the values across the y-axis correspond to the sample indices. Samples are sorted based on their group assignments (see model description in Eq. 1 in the main text).

with a modified version of **E1** (see Eq. 9 in the Appendix section B.4) by changing the model to consist of two overlapping features. Specifically, the $\text{Logit}(\boldsymbol{x})$ in the new function we evaluate becomes:

$$\text{Logit}(\boldsymbol{x}) = \begin{cases} e^{(\boldsymbol{x}_1 \times \boldsymbol{x}_3)}, & \text{if } \boldsymbol{x}_{11} < 0, \\ e^{(\sum_{i=3}^{6} \boldsymbol{x}_i^2 - 4)}, & \text{otherwise} \end{cases} \tag{8}$$

We note that in this example, both $x_{11}$ and $x_3$ are overlapping. $x_3$ has an unequal factor, and the nonlinear function is not the same in both groups. We applied LSPIN and obtained a true positive rate of $94.68\%$, a false discovery rate of $12.58\%$ on the test set, and test accuracy of $93.00\%$.

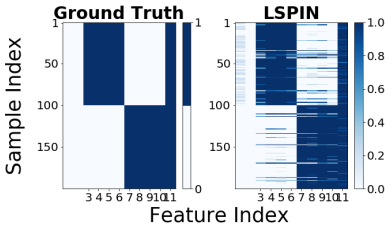
For this modified version of **E1**, we generate data of $6000$ samples by $11$ features with $90\%$ as the training set, $5\%$ as the validation set, and $5\%$ as the test set. In this example, we set the architecture of LSPIN to $5$ hidden layers with $100$ neurons in each layer in the *prediction* network. The number of hidden layers in the *gating* network is $3$ with $100$ neurons in each layer. We set the batch size to $1000$ for training. $\lambda$ is set to $0.15$, the learning rate is set to $0.08$, and the number of epochs is set to $5000$.



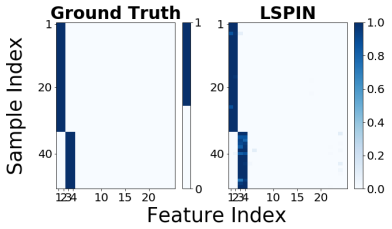

(a) **E1**



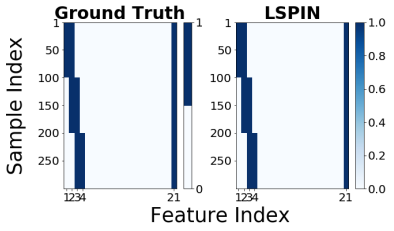

(b) **E2**




(c) **E3**




(d) **E4**




(e) **E5**

Figure A.3: Informative feature identification using the nonlinear synthetic datasets. We present heat maps comparing the ground truth informative features (the left panel in each subfigure) and identified features by LSPIN based on the 5 synthetic datasets (see description in Appendix section B.4). For more convenient visualization, we only present the first 25 features for **E4**.

### A.4. Performance Evaluation with Additional Noisy Features on the Nonlinear Synthetic Dataset

In the following subsection, we explore the bound of D for which our model can help with performance improvement. Using the nonlinear synthetic dataset **E1**, we fix the number of samples and then further add different numbers of noisy features ($D = 10, 20, 30, 40, 50, 100, 150, 200$, sampled from $U(-1, 1)$) and evaluate the performance of our model (LSPIN) in each scenario.

Fig. A.4 shows that our model performs reasonably well in terms of test accuracy and F1 score, with fewer than 50 noisy features added. As the number of noisy features increases, the performance of our model gradually drops. On the other hand, increasing the number of samples would allow our model to perform better even when many noisy features are added.

### A.5. Comparison of Model's Ability to Alleviate Overfitting

To evaluate our model's ability to alleviate the overfitting problem, we compare our model (LSPIN) to three other models (Neural Network, Neural Network with dropout (drop out rate 0.1), and Neural Network with weight decay ($L_2$ regularizer with scale 0.1)) in terms of the training and validation loss change on the nonlinear synthetic dataset (**E4**).

Fig. A.5 shows that our model maintains a small generalization error between training and validation set compared to other baselines.

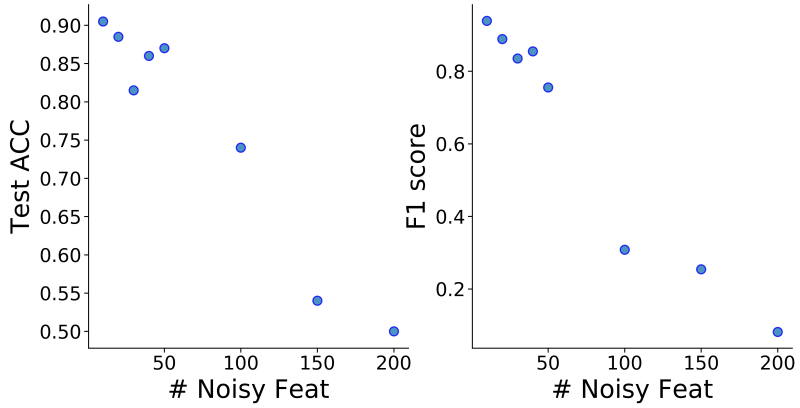



Figure A.4: Evaluation of Test ACC and F1 score of LSPIN on **E1** with different number of noisy features

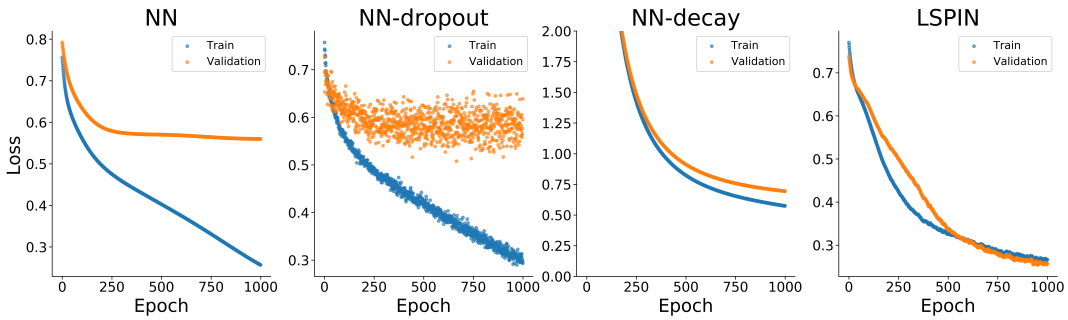
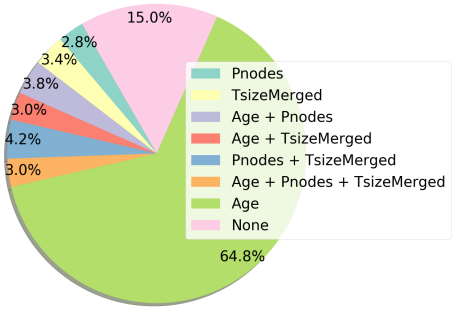



Figure A.5: Comparison of the loss change on **E4**

### A.6. Statistics of the Selected Features for Other Methods on the SEER Dataset

In Fig. A.6, we present the selected variables on the SEER dataset for other methods (COX-LSPIN,COX-STG,COX-LASSO) compared to the result of COX-LLSPIN as shown in the piechart of Fig. 4. Random Survival Forest and DeepSurv are not shown since both methods select almost all variables.

The global set of selected features appears to be similar, however, we argue that sample-specific variable selection as in LLSPIN/LSPIN discards variables that are not most relevant to the prediction for some patients and leads to improved prediction performance, as shown on the left of Fig. 4.




| Method | Selected Variables |
|---|---|
| COX-STG(Linear) | Age, Pnodes, TsizeMerged, NodesRatio |
| COX-STG(Nonlinear) | Age, Pnodes, TsizeMerged |
| COX-LASSO | Age, Pnodes, TsizeMerged, NodesRatio |

Figure A.6: Left: Frequency of variables selected by COX-LSPIN across the different patients. Right: Selected variables by the global feature selection methods

### A.7. Diversity Evaluation Using the Jaccard Index

We expect an excellent interpretability model to identify different sets of variables as driving factors for explaining distinct classes. To evaluate the diversity of all models, we propose using the following Jaccard-based metric.

First, we compute for each method the per class median set of active features (as indicated by the gates for our method). We denote this set for class $c_i, i = 1, ..., M$, as $\mathcal{S}_{c_i}$. Then, for each pair of classes we compute intersection using the Jaccard index, namely

$$J(\mathcal{S}_{c_i}, \mathcal{S}_{c_j}) = \frac{|\mathcal{S}_{c_i} \cap \mathcal{S}_{c_j}|}{|\mathcal{S}_{c_i} \cup \mathcal{S}_{c_j}|}, i \neq j,$$

then we sum over all possible pairs and normalize by the size of this set, and scale it to $[0, 100]$, specifically

$$Diversity = 100(1 - \sum_{i \neq j} \frac{J(\mathcal{S}_{c_i}, \mathcal{S}_{c_j})}{M(M-1)/2}).$$

This quantifies what is the portion of non overlapping features between distinct classes.

### A.8. Extended results for the MNIST Experiment

Extending from the MNIST experiment demonstrated in Section 5.2, here we present additional randomly selected images for the different MNIST classes and superimpose the images using gates with non zero values in Figs. A.7,A.8, and A.9. In these examples we highlight the effect of adding the second regularization term, controlled by $\lambda_2$. Finally, in Fig. A.10 we present the selected pixels when we tune our model to select more features per sample.

### A.9. Experiment with Low Sample Size MNIST Dataset

To further test the performance of our method on the LSS regime, we evaluate LLSPIN on a low sample size MNIST variant. Specifically, we use 6k samples for training and 10k samples for testing. A fully connected MLP with four layers [784,300,100,10] and a tanh activation leads to a test accuracy of $94.2\%$. Using the linear variant of our method LLSPIN, we reach a test accuracy of $94.8\%$ while using a median of $10$ pixels per image with no activation in the *prediction* network. In comparison, this may seem like a minor improvement and far from state-of-the-art (which requires a convolution layer). However, we argue that because our prediction model is linear, we can easily interpret each prediction since we obtain a small subset of active pixels and their linear coefficient values.

### A.10. Identification of Cell-type Specific Markers

In this subsection, we examine LLSPIN's interpretability capabilities on the Single Nucleus RNA-sequencing dataset described in Section 5.5.

We note that in our original implementation (see Eq. 5), the model encourages similar samples to select similar features but doesn't encourage dissimilar samples to select different features. For this marker gene selection task, we find that this model tends to select genes that are stable across cells even though they are of different cell types (left of Fig. A.11).

To encourage our model to select different genes for each cell type, we modify the second term in our regularizer (see Eq. 5) to $\lambda_2 \sum_j (1 - K_{i,j}) \times (-\|\mathbf{z}^{(i)} - \mathbf{z}^{(j)}\|_2^2)$, such that when sample $i$ and $j$ are dissimilar ($K_{i,j}$ is small), the corresponding gates are encouraged to be different.

As shown on the right of Fig. A.11, LLSPIN with the modified regularizer identifies the correct cell type specific marker genes for each cell type group (*ITGAM* gene for Microglia cells and *PDGFRA* gene for Oligodendrocyte Precursor Cells (OPC)). Therefore, we recommend to use our model with the modified regularizer for this task.

### A.11. Sparsity of the Local Gates

To demonstrate that our model performs local sparsification of input features, we evaluate the statistics of the gate values. Table A.1 presents the distribution of local gates values in some of the synthetic experiments. Our results demonstrate that most gates converge to $0$ and $1$ with only a few gates converging to values in the range $(0, 1)$.

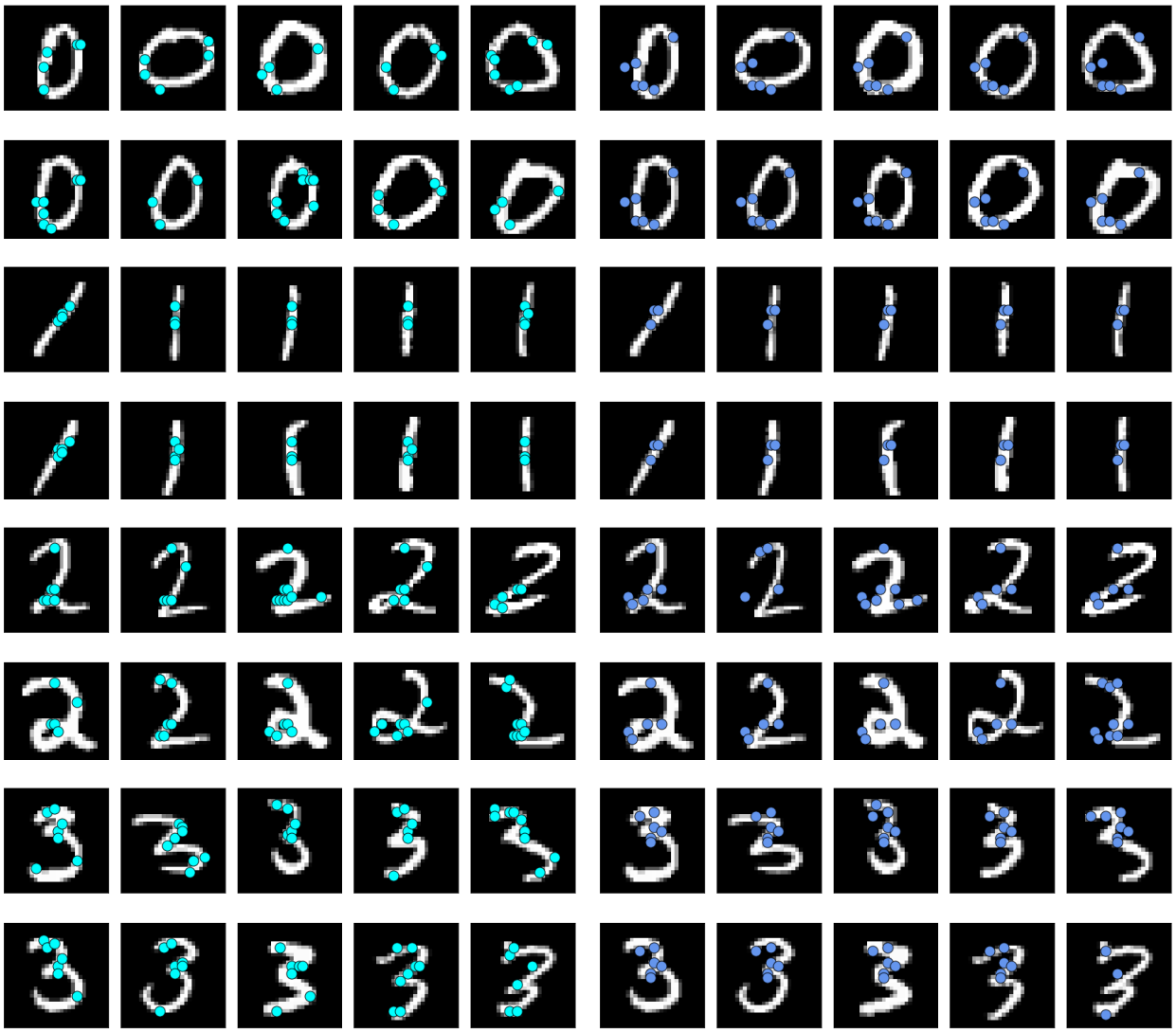


Figure A.7: Random representative samples of $0-3$ from the MNIST dataset. For each example, we overlay the image with active gates, here color dots indicate locations of non zero gates. Left columns (cyan dots) represents the selected pixels with LLSPIN using $\lambda_2 = 0$. Right columns (darker blue dots) represents the selected pixels with LLSPIN using $\lambda_2 = 0.1$. Notice that when we increase $\lambda_2$ the selection becomes more stable across samples within a class.

### A.12. Evaluation of the Fairness and Robustness of LLSPIN

Following the analysis in (Ovadia et al., 2019), we conducted several experiments to evaluate how our proposed method performs under distributional shifts. Specifically, we have compared the accuracy, Negative Log-Likelihood (NLL), and Expected Calibration Error (ECE) of our model (LLSPIN) to those obtained by a fully connected nonlinear MLP. Table A.2 details the results for different rotation angles of the test samples from MNIST datasets.

Next, we present the NLL results based on the experiments performed on MNIST, low sample size MNIST, and real-world datasets in Table A.3. Based on the results presented in the table, we conclude that our model leads to better-calibrated uncertainty. This is evident by the lower ECE and NLL values our model obtains compared with others.

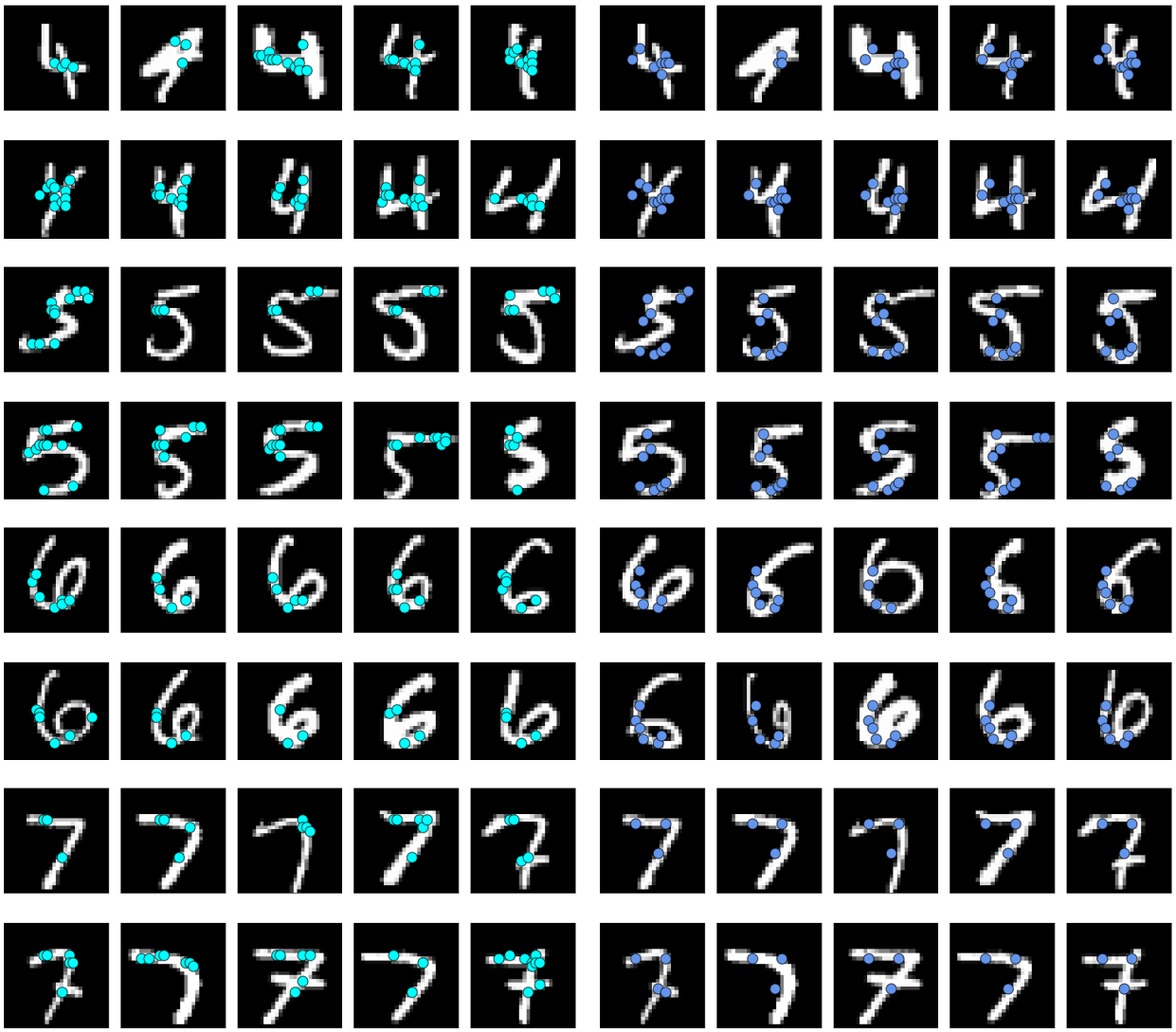


Figure A.8: Random representative samples of $4-7$ from the MNIST dataset. We overlay the image with active gates locations for each example, indicated here as color dots. Left columns (cyan dots) represents the selected pixels with LLSPIN using $\lambda_2 = 0$. Right columns (darker blue dots) represents the selected pixels with LLSPIN using $\lambda_2 = 0.1$. Notice that when we increase $\lambda_2$ the selection becomes more stable across samples within a class.

### A.13. Discussion on the Number of Selected Features

When applied to high dimensional data, the model may select a large set of features across some samples. This phenomenon suggests that the model is overfitting. Based on our experience, this happens for a small set of observations in the dataset (both in the train and test). For example, in the PBMC data for $8.6\%$ of the train samples, the model selects more than $10\%$ of the features (genes). This information could indicate that the model may be overfitting on a small subset of samples. Therefore, we believe that the local gates could provide additional information for practitioners to help interpret the sample-specific predictions made by the model instead of just relying on the average accuracy the model obtains on the test set. We did not observe this overfitting phenomenon in some of the low-dimensional datasets evaluated in the paper. For example, on MNIST, the model selects a median of $8$ features, a union of $165$ features. Furthermore, we provide Table A.4 to demonstrate the statistics of the selected features on MNIST.

Another example is the SEER cancer dataset; when the model selects a median of 3 features (leftmost point of Fig. 4 (Left)),

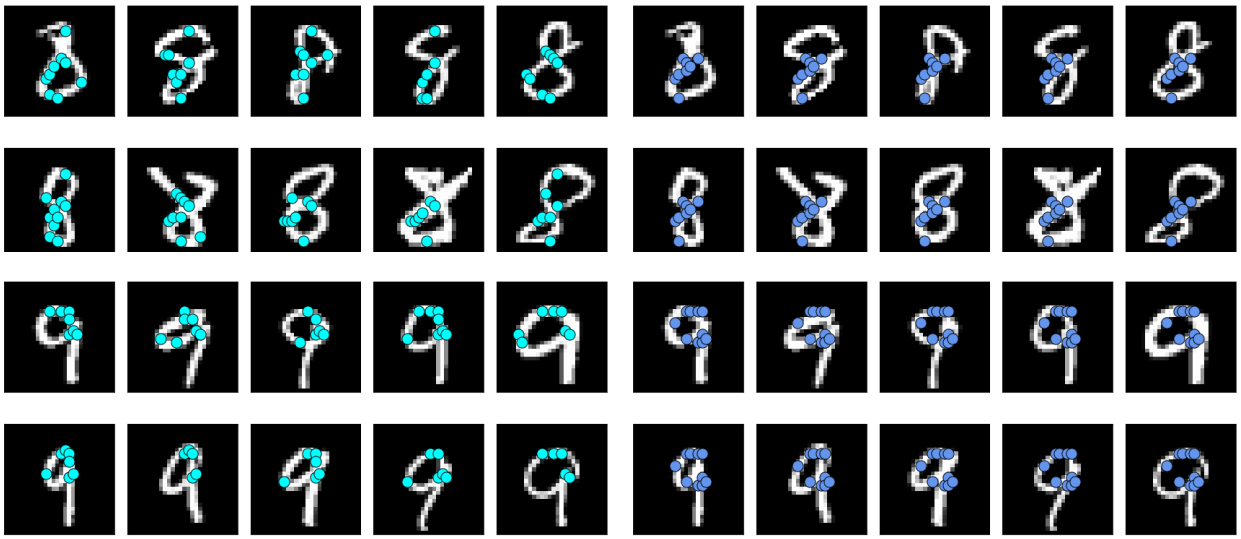


Figure A.9: Random representative samples of $8-9$ from the MNIST dataset. We overlay the image with active gates locations for each example, indicated here as color dots. Left columns (cyan dots) represents the selected pixels with LLSPIN using $\lambda_2 = 0$. Right columns (darker blue dots) represents the selected pixels with LLSPIN using $\lambda_2 = 0.1$. Notice that when we increase $\lambda_2$ the selection becomes more stable across samples within a class.

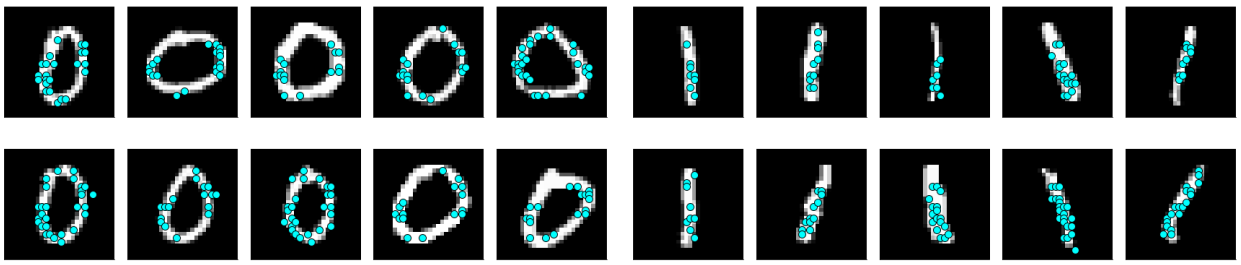


Figure A.10: Random representative samples of $0-1$ from the MNIST dataset. We overlay the image with active gates locations for each example, indicated here as color dots. Here, we tune the model to select a median of 24 informative features per image.

the union number of selected features is 9. In Table A.5, we present the statistics of the features chosen for this example.

In these two examples, the model can also reduce the total number of selected features (since the union of features used by the model is relatively small).

### A.14. Time Benchmark Results

To demonstrate the computational efficiency of our models, we first compare the training running time between LLSPIN/LSPIN and INVASE on the nonlinear synthetic example **E1** (Eq. 9 in Appendix section B.4) where we vary the number of training samples $(200, 600, 1000, 1400, 1800)$ as shown in Fig. A.12a. We can see that the running time of INVASE increases rapidly with more training samples, whereas our models remain scalable.

To benchmark the running time when there are more features, we use the dataset from **E2** (Eq. 10 in Appendix section B.4) and generate additional noisy features (2000 features in total including the informative 11 features) by sampling values from $\mathcal{N}(0, 1)$. We show the comparison in Fig. A.12b. In this high dimensional regime, LLSPIN/LSPIN remain scalable compared with INVASE.

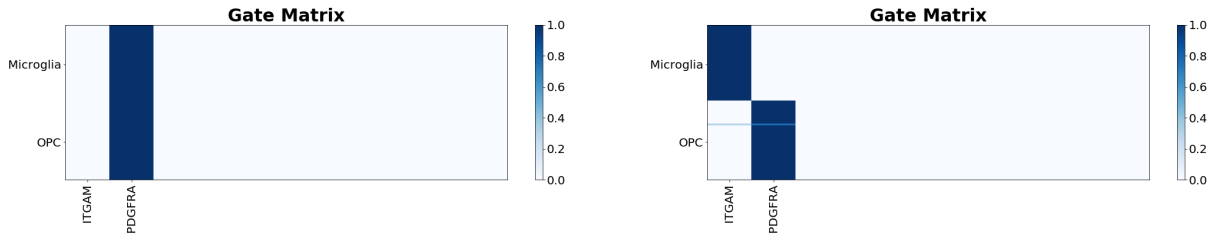



Figure A.11: Identification of the cell-type-specific markers by LLSPIN's local gates ($> 0$ for open gates and $0$ for closed gates) with the original regularizer (left) and the modified regularizer (right). The values across the x-axis correspond to the different genes (with only the 2 marker genes shown), and the values across the y-axis correspond to the different cells (sorted by the cell types: Microglia and Oligodendrocyte Precursor Cells (OPC)). For more convenient visualization, we only present the first 30 genes.

| | | Training | | | Test | | |
|---|---|---|---|---|---|---|---|
| Experiments | Models | % of 0s | % of 1s | **% 0-1** | % of 0s | % of 1s | **% 0-1** |
| Linear 60 samples | LLSPIN | 40.00 | 57.67 | **2.33** | 40.00 | 57.33 | **2.67** |
| Linear 30 samples | LLSPIN | 40.00 | 60.00 | **0.00** | 40.00 | 60.00 | **0.00** |
| Linear 18 samples | LLSPIN | 40.00 | 60.00 | **0.00** | 40.00 | 60.00 | **0.00** |
| Linear 12 samples | LLSPIN | 40.00 | 60.00 | **0.00** | 40.00 | 60.00 | **0.00** |
| Linear 10 samples | LLSPIN | 40.00 | 58.00 | **2.00** | 40.00 | 59.33 | **0.67** |
| Linear 6 samples | LLSPIN | 50.00 | 50.00 | **0.00** | 50.00 | 49.33 | **0.67** |
| Nonlinear **E1** | LLSPIN | 91.11 | 8.60 | **0.29** | 91.14 | 7.27 | **1.59** |
| | LSPIN | 62.90 | 36.54 | **0.56** | 62.27 | 37.00 | **0.73** |
| Nonlinear **E2** | LLSPIN | 78.36 | 20.68 | **0.96** | 78.59 | 20.09 | **1.32** |
| | LSPIN | 63.25 | 35.53 | **1.22** | 62.91 | 35.59 | **1.50** |
| Nonlinear **E3** | LLSPIN | 83.67 | 13.69 | **2.63** | 83.36 | 13.32 | **3.32** |
| | LSPIN | 52.04 | 44.90 | **3.07** | 51.82 | 44.73 | **3.45** |
| Nonlinear **E4** | LLSPIN | 95.50 | 3.56 | **0.94** | 95.68 | 3.28 | **1.04** |
| | LSPIN | 95.82 | 3.74 | **0.44** | 95.72 | 3.68 | **0.60** |

Table A.1: Statistics of the local gate values. The percentage of gates between $0$ and $1$ is listed in bold.

| Rotation MNIST | LLSPIN | | | FULL | | |
|---|---|---|---|---|---|---|
| Angle | ACC | NLL | ECE | ACC | NLL | ECE |
| 10 | **96.9** | **1.403** | **0.014** | 96.8 | 1.987 | 0.023 |
| 20 | **92.1** | **3.777** | **0.039** | 90.8 | 6.455 | 0.071 |
| 30 | **77.9** | **10.341** | **0.126** | 75.8 | 19.859 | 0.195 |
| 40 | **59.8** | **20.532** | **0.243** | 56.4 | 43.771 | 0.367 |
| 50 | **40.1** | **31.006** | **0.377** | 38.6 | 71.346 | 0.532 |
| 60 | **27.8** | **40.641** | **0.473** | 26.8 | 96.177 | 0.643 |
| 70 | **19.49** | **48.614** | **0.543** | 18.4 | 119.586 | 0.724 |
| 80 | 14.24 | **55.249** | **0.601** | **14.6** | 139.701 | 0.769 |
| 90 | 11.7 | **60.477** | **0.637** | **12.3** | 155.147 | 0.795 |

Table A.2: Comparison of classification accuracy (ACC), Negative Log Likelihood (NLL), Expected Calibration Error (ECE) on rotated MNIST.

We design LLSPIN/LSPIN/INVASE using 2 hidden layers with 200 neurons each to perform a fair comparison. We set 2 hidden layers with 100 neurons each for the gating network of LLSPIN/LSPIN and the selector network of INVASE. For all the three models, $\lambda$ is set to 1, the batch size is set to full batch training, and epochs is set to 3000. We set the learning rate for INVASE to $0.0001$ (Adam optimizer) and $0.1$ for SGD optimizer of LLSPIN/LSPIN.

| | LLSPIN | FULL | INVASE |
|---|---|---|---|
| MNIST | **0.826** | 1.043 | 2.008 |
| low sample size MNIST | **1.985** | 3.311 | 4.675 |
| TOX-171 | **0.760±0.411** | 3.968±0.500 | 2.593±0.799 |
| RELATHE | **0.879±0.189** | 1.858±0.047 | 1.098±0.123 |
| BASEHOCK | **0.367±0.057** | 1.813±0.040 | 0.668±0.103 |
| COLON | **0.884±0.415** | 1.071±0.394 | 1.318±0.653 |
| PBMC | **1.012±0.046** | 4.062±0.059 | 1.441±0.048 |
| PCMAC | **0.816±0.118** | 1.873±0.049 | 0.905±0.050 |

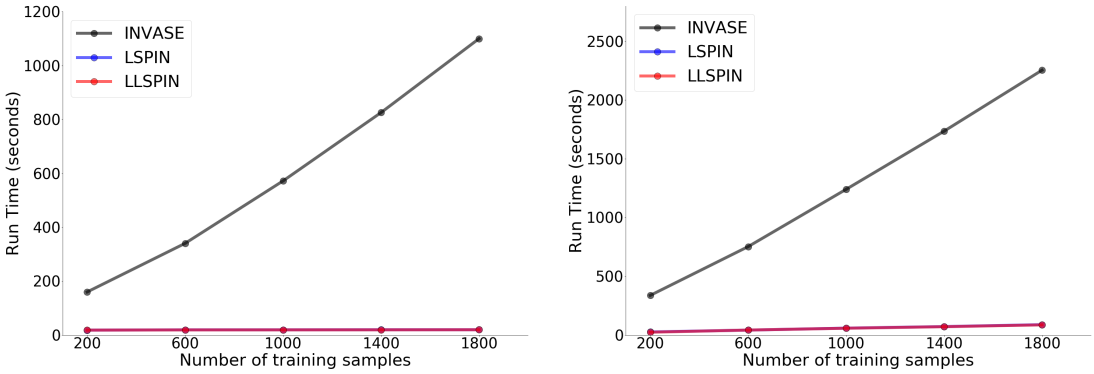
Table A.3: Comparison of Negative Log-Likelihood (NLL) on MNIST, low sample size MNIST, and the real-world datasets.

| Number of selected features | Percent of samples |
|---|---|
| $< 5$ | 5.90% |
| $5 - 9$ | 79.36% |
| $10 - 14$ | 14.50% |
| $15 - 19$ | 0.24% |
| $20+$ | 0.00% |

Table A.4: Statistics of the selected features on MNIST

| Number of selected features | Percent of samples |
|---|---|
| $< 3$ | 95.25% |
| $3 - 5$ | 4.73% |
| $6 - 7$ | 0.02% |
| $8+$ | 0.00% |

Table A.5: Statistics of the selected features on the SEER dataset.




(a) (b)

Figure A.12: Time Benchmark between LLSPIN/LSPIN and INVASE on 2 datasets

## B. Reproducibility and Additional Details

In the following subsections, we provide additional experimental details required for the reproduction of the experiments provided in the main text. The CPU model used for the experiments is Intel(R) Xeon(R) Gold 6150 CPU @ 2.70GHz (72 cores total). GPU model is NVIDIA GeForce RTX 2080 Ti. The operating system is Ubuntu 20.04.2 LTS. The memory storage is 1 TB in total. The software dependencies are specified in the associated codes.

We apply batch normalization to the prediction network of LLSPIN/LSPIN and STG models throughout the experiments,

except for the MNIST example and the survival analysis. We note that the application of batch normalization to the MNIST data did not improve performance. In the survival analysis, the performance was satisfactory without the application of batch normalization.

For LLSPIN/LSPIN/STG models, the network weights are initialized by drawing $\mathcal{N}(0, s)$ and bias terms are set to $0$. For LLSPIN/LSPIN, we set $s$ to be $0.1$ for the MNIST example, time benchmark experiments, and the synthetic datasets experiments (Except for LSPIN on **E1** and LLSPIN/LSPIN on **E3** in Section 5.1 where $s$ is set to $0.05$ for better convergence). We set $s = \frac{1}{\sqrt{D}}$ ($D$ is the input dimensionality) following Xaiver initialization (Glorot & Bengio, 2010) for the real-world examples, and $s = 0.05$ for the cox survival analysis. For marker gene identification, $s$ in the prediction network is $0.1$, and $s$ in the gating network is set to $0.001$ which we found to be helpful for stabilizing the training. For STG models, $s$ is set to $0.1$ for the real-world datasets and $0.05$ for the cox survival analysis. Unless expressly noted, we use tanh as the hidden layer activation function for both the gating network of LLSPIN/LSPIN and the nonlinear prediction network of LSPIN/STG. For other neural network-based methods, we use their default activation functions.

Across the experiments, we only enabled the second regularization term in the MNIST example and the marker gene identification example.

For the experiments, standard metrics in supervised learning, including classification accuracy, $R^2$ and mean squared error and concordance index, are adopted to evaluate the performance of different models in classification, regression, and survival analysis tasks.

### B.1. Algorithms

**Algorithm 1** Locally SParse Interpretable Networks (LSPIN) Pseudo-code

**Training:**
**Input:** observations $\{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$, regularization parameter $\lambda$, number of epochs $T$, batch size $B$, learning rate $\gamma$.
**Output:** Gating network $\boldsymbol{\Psi}_{\boldsymbol{\Omega}}$ and prediction model $\boldsymbol{f}_{\boldsymbol{\theta}}$
Initialize the weights $\boldsymbol{\Omega}$ of gating network $\boldsymbol{\Psi}$
**for** $t = 1$ **to** $T$ **do**
  **for** each size B batch **do**
    **for** $i = 1$ **to** $B$ **do**
      Compute $\boldsymbol{\mu}^{(i)} = \boldsymbol{\psi}(\boldsymbol{x}^{(i)}|\boldsymbol{\Omega})$
      Sample $\epsilon^{(i)}$ from $\mathcal{N}(0, \boldsymbol{I}\sigma^2)$
      Compute local stochastic gates:
        $\mathbf{z}^{(i)} = \max(0, \min(1, 0.5 + \boldsymbol{\mu}^{(i)} + \epsilon^{(i)}))$
    **end for**
    Compute the loss:
    $\tilde{\boldsymbol{L}} = \frac{1}{B}\sum_{i=1}^{B}(\|f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)} \odot \mathbf{z}^{(i)}) - y^{(i)}\|_2 + \boldsymbol{R}(\mathbf{z}^{(i)}))$ (where $\boldsymbol{R}(\mathbf{z}^{(i)})$ is defined in Eq. 5)
    Update $\boldsymbol{\theta} = \boldsymbol{\theta} - \gamma\nabla_{\boldsymbol{\theta}}\tilde{\boldsymbol{L}}, \quad \boldsymbol{\Omega} = \boldsymbol{\Omega} - \gamma\nabla_{\boldsymbol{\Omega}}\tilde{\boldsymbol{L}}$
  **end for**
**end for**

**Inference:**
**Input:** observations $\{\boldsymbol{x}^{(i)}\}_{i=1}^{M}$ with $\boldsymbol{x}^{(i)} \in \mathbb{R}^D$, Trained gating network $\boldsymbol{\Psi}_{\boldsymbol{\Omega}}$, and prediction model $f_{\boldsymbol{\theta}}$
**Output:** Local gates:
$\{\boldsymbol{z}^{(i)} = \max(0, \min(1, 0.5 + \boldsymbol{\psi}(\boldsymbol{x}^{(i)}|\boldsymbol{\Omega})))\}_{i=1}^{M}$
Predictions: $\{\tilde{y}^{(i)} = f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)} \odot \boldsymbol{z}^{(i)})\}_{i=1}^{M}$

### B.2. Regularization Term

The leading term in our regularizer is expressed by :

$$\begin{aligned}\mathbb{E}_Z \left\|\boldsymbol{Z}\right\|_0 &= \sum_{d=1}^{D} \mathbb{P}[z_d > 0] = \sum_{d=1}^{D} \mathbb{P}[\mu_d + \sigma\epsilon_d + 0.5 > 0] \\ &= \sum_{d=1}^{D} \{1 - \mathbb{P}[\mu_d + \sigma\epsilon_d + 0.5 \leq 0]\} \\ &= \sum_{d=1}^{D} \{1 - \Phi(\frac{-\mu_d - 0.5}{\sigma})\} \\ &= \sum_{d=1}^{D} \Phi\left(\frac{\mu_d + 0.5}{\sigma}\right) \\ &= \sum_{d=1}^{D} \left(\frac{1}{2} - \frac{1}{2}\,\mathrm{erf}\left(-\frac{\mu_d + 0.5}{\sqrt{2}\sigma}\right)\right)\end{aligned}$$

To tune $\sigma$, we follow the suggestion in (Yamada et al., 2020). Specifically, the effect of $\sigma$ can be understood by looking at the value of $\frac{\partial}{\partial \mu_d}\mathbb{E}_Z||\boldsymbol{Z}||_0$. In the first training step, $\mu_d$ is 0. Therefore, at initial training phase, $\frac{\partial}{\partial \mu_d}\mathbb{E}_Z||\boldsymbol{Z}||_0$ is close to $\frac{1}{\sqrt{2\pi\sigma_d^2}} e^{-\frac{1}{8\sigma_d^2}}$. To enable sparsification, this term (multiplied by the regularization parameter $\lambda$) has to be greater than the derivative of the loss with respect to $\mu_d$ because otherwise $\mu_d$ is updated in the incorrect direction. To encourage such behavior, we tune $\sigma$ to the value that maximizes the gradient of the regularization term. As demonstrated in Fig. B.1 this is obtained when $\sigma = 0.5$. Therefore, we keep $\sigma = 0.5$ throughout our experiments unless specifically noted.

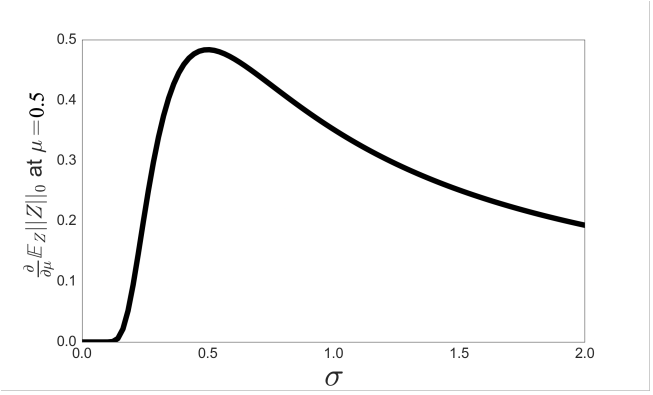



Figure B.1: The value of $\frac{\partial}{\partial \mu}\mathbb{E}_Z||\boldsymbol{Z}||_0|_{\mu=0.5} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{8\sigma^2}}$ for $\sigma = [0.001, 2]$.

### B.3. Linear Regression Example Details

#### B.3.1. Details of the Motivating Example and Extended Evaluations

First, we describe the data model used in the motivating example (see Section 2) and the extended results in section A.1.

In total, the data matrix $\boldsymbol{X}$ has $2N = 600$ samples where the first 300 samples (group 1) are i.i.d. based on $\mathcal{N}(\mathbf{1}, 0.5\boldsymbol{I})$, and the remaining 300 i.i.d. samples (group 2) are drawn from $\mathcal{N}(-\mathbf{1}, 0.5\boldsymbol{I})$ where $\boldsymbol{I}$ is a $5 \times 5$ identity matrix. 10% of the samples (60 data points) are used as validation set, 10% of the samples (60 data points) are used as the test set. From the remaining data (480 data points), we randomly pick small subsets of samples (60,30,18,12,10,6 samples) as our training sets. We compute y based on Eq. 1 in the main text for all the samples. For the motivating example, the training set has 10 samples. For the extended evaluations, the training set has 60,30,18,12,6 samples for each case.

For LLSPIN, LASSO, Random Forest, and Neural Network, we optimize each model using 100 trials of Optuna (a hyper-parameter optimization software (Akiba et al., 2019)) on the validation set by minimizing the validation mean squared error, with grids of parameters listed in Table B.1. After Optuna selects the parameters, we test each model's performance on the test set.

| Parameters | Search Range |
|---|---|
| learning rate (LLSPIN,Neural Net) | [1e-2,2e-1] |
| epochs (LLSPIN,Neural Net) | $\{2000, 5000, 10000, 15000\}$ |
| $\lambda$ (LLSPIN) | [1e-3,1e-2] |
| $\alpha$ (LASSO) | [1e-3,5e-1] |
| n_estimators (RF) | [1,500] |
| max_depth (RF) | [1,30] |
| min_samples_split (RF) | [2,10] |

Table B.1: Parameter settings for different models applied to the linear synthetic example

For the motivating example where the training size is 10, we include Localized LASSO and INVASE into comparison. For Localized LASSO, the affinity between samples are computed using the default gaussian kernel from sklearn.metrics.pairwise.pairwise_kernels. We also optimize these 2 models via 100 trials of Optuna. For Localized LASSO, the grid of $lam_net$ and $lam_exc$ is set to be $\{0.001, 0.01, 0.1, 1, 5, 10, 50, 100\}$, and the grid of number of iterations is $\{100, 300, 500, 1000\}$. For INVASE, the grid of $\lambda$ is $[1e-3, 1e-2]$. The grid of number of epochs is $\{2000, 5000, 10000, 15000\}$. The grid of learning rate is $[1e-5, 1e-4]$.

In this example, we set the architecture of LLSPIN and the baseline Neural Network to 4 hidden layers with 100, 100, 10, 1 neurons in each layer, respectively. For the *gating* network of LLSPIN we use 1 hidden layer with 10 neurons. We use full batch training for both LLSPIN and the Neural Network. INVASE is set to have identical architectures.

#### B.3.2. Details of the Experiment involving Linear Synthetic Dataset with Unequal Feature Coefficients

We take all the 480 remaining data points as training samples for this experiment. y is computed based on Eq. 7 in Section A.2 for all the samples. In this example, we set the architecture of LSPIN to 3 hidden layers with 100, 10, 1 neurons in each layer, respectively. For the *gating* network of LSPIN we use 2 hidden layer with 100 neurons in each layer. The activation function of the *prediction* network is set to relu for this particular example. The $\lambda$ is set to $10^{-5}$, the learning rate is set to 0.2, and the number of epochs is set to 3500. We use full batch training for this example.

### B.4. Nonlinear Synthetic Datasets Details

Here we provide details for reproduction of the example presented in Section 5.1.

#### B.4.1. Data Generation and Split

**E1**, **E2**, and **E3** are adapted from (Yoon et al., 2018) (see Eq. 9, 10, and 11). In each example, we generate the data matrix $\boldsymbol{X}$ with 2000 samples and 11 features that are sampled independently from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ where $\boldsymbol{I}$ is an $11 \times 11$ identity matrix. The response is $\text{y} = \mathbf{1}_A(\frac{1}{1+\text{Logit}(\boldsymbol{x})} > 0.5)$ where $\mathbf{1}_A$ is an indicator function and the $\text{Logit}(\boldsymbol{x})$ for each sample is calculated based on different features depending on the sign of the 11th feature $x_{11}$. Each Logit is defined based on one of the following equations

$$\textbf{E1:} \quad \text{Logit} = \begin{cases} e^{(\boldsymbol{x}_1 \times \boldsymbol{x}_2)}, & \text{if } \boldsymbol{x}_{11} < 0, \\ e^{(\sum_{i=3}^{6} \boldsymbol{x}_i^2 - 4)}, & \text{otherwise} \end{cases} \tag{9}$$

$$\textbf{E2:} \quad \text{Logit} = \begin{cases} e^{(\boldsymbol{x}_1 \times \boldsymbol{x}_2)}, & \text{if } \boldsymbol{x}_{11} < 0, \\ e^{(-10 \sin(0.2\boldsymbol{x}_7) + |\boldsymbol{x}_8| + \boldsymbol{x}_9 + e^{-\boldsymbol{x}_{10}} - 2.4)}, & \text{otherwise} \end{cases} \tag{10}$$

$$\textbf{E3:} \quad \text{Logit} = \begin{cases} e^{(\sum_{i=3}^{6} \boldsymbol{x}_i^2 - 4)}, & \text{if } \boldsymbol{x}_{11} < 0, \\ e^{(-10 \sin(0.2\boldsymbol{x}_7) + |\boldsymbol{x}_8| + \boldsymbol{x}_9 + e^{-\boldsymbol{x}_{10}} - 2.4)}, & \text{otherwise} \end{cases} \tag{11}$$

To evaluate a LSS regime, the number of samples we use is far fewer compared with the number of samples used in experiments conducted in (Yoon et al., 2018). We split the data and used $90\%$ for training and $10\%$ for testing. $5\%$ of the training set is set aside as a validation set.

Additionally, to further demonstrate LSPIN's robustness on challenging domains, we design a $4^{\text{th}}$ example (termed **E4**, see Eq. 12), for which we generate the data matrix $\boldsymbol{X}$ with $2N = 1000$ samples and $4$ features that consists of 2 sample groups. The first $N$ samples are sampled from $\mathcal{N}(\mathbf{1}, 0.5\boldsymbol{I})$, and the second $N$ samples are sampled from $\mathcal{N}(-\mathbf{1}, 0.5\boldsymbol{I})$ where $\boldsymbol{I}$ is a $4 \times 4$ identity matrix. The response is defined as $\text{y} = \mathbf{1}_A(\frac{1}{1+\text{Logit}(\boldsymbol{x})} > 0.5)$, where $\mathbf{1}_A$ is an indicator function and $\text{Logit}(\boldsymbol{x})$ depends on features $x_1$ and $x_2$ for the first $N$ samples, and on features $x_3$ and $x_4$ for the remaining $N$ samples. To make the classification task harder, we add other $46$ nuisance features (irrelevant for the prediction task) sampled from $\mathcal{N}(\mathbf{0}, 0.5\boldsymbol{I})$ where $\boldsymbol{I}$ is an $46 \times 46$ identity matrix.

$$\textbf{E4:} \quad \text{Logit} = \begin{cases} e^{(\boldsymbol{x}_1 \times \boldsymbol{x}_2 - 0.9)}, & \text{if first N samples} \\ e^{(\boldsymbol{x}_3^2 + \boldsymbol{x}_4^2 - 2.5)}, & \text{otherwise} \end{cases} \tag{12}$$

For **E4**, we split the data and use $95\%$ for training and $5\%$ for testing. $10\%$ of the training set is set aside as validation set.

Lastly, to evaluate our model in the nonlinear regression regime, we design a moving XOR dataset as the 5th example (**E5**, see Eq. 13). Specifically, we generate the data matrix $\boldsymbol{X}$ with $3N = 2100$ samples and $20$ features, where each entry is sampled from a fair Bernoulli distribution ($P(x_{ij} = 1) = P(x_{ij} = -1) = 0.5$). Then we add an additional feature $x_{21}$ for each sample where $x_{21} = -1$ for the first $N$ samples, $x_{21} = 0$ for the second $N$ samples, $x_{21} = 1$ for the last $N$ samples. Based on the value of $x_{21}$, the response variable y for different samples will have different subset of features, as defined in Eq. 13.

$$\textbf{E5:} \quad \text{y} = \begin{cases} \boldsymbol{x}_1 \times \boldsymbol{x}_2 + 2\boldsymbol{x}_{21}, & \text{if } \boldsymbol{x}_{21} = -1, \\ \boldsymbol{x}_2 \times \boldsymbol{x}_3 + 2\boldsymbol{x}_{21}, & \text{if } \boldsymbol{x}_{21} = 0, \\ \boldsymbol{x}_3 \times \boldsymbol{x}_4 + 2\boldsymbol{x}_{21}, & \text{if } \boldsymbol{x}_{21} = 1, \end{cases} \tag{13}$$

For **E5**, the training set has $1500$ samples, the validation and test have $300$ samples each.

### B.4.2. Training Procedures and Hyper-parameter Tuning and Settings

For these $5$ experiments, we optimize each model on the validation set (minimizing classification error for classification and mean squared error for regression) using Optuna and evaluate the optimized models on the test sets. For the neural network based methods, the F1-score of the selected features is also evaluated on the test set.

For lasso, we optimize the $l_1$ regularization parameter with 20 trials and the grid range is $[1e-2, 1e3]$. For SVC, we optimize the regularization parameter $c$ with 20 trials and the grid range is $[1e-2, 1e3]$. For Random Forest, we optimize the number of estimators, max_depth, and min_samples_split with 100 trials and the corresponding grid ranges are $[1, 500]$, $[1, 30]$, $[2, 10]$. For XGBoost, we optimize the number of estimators and max_depth with 20 trials and the corresponding grid ranges are $[1, 500]$ and $[1, 30]$.

For MLP/STG(Linear&Nonlinear)/LLSPIN/LSPIN/INVASE/L2X/REAL-x, the parameter settings and grids are listed in Table B.2. The number of hidden layers and nodes are identical for these models. For **E2**, **E3**, **E4**, we use 2 hidden layers with 200 nodes each for the prediction network architecture. We use 2 layers with 100 nodes each for the gating network of LLSPIN/LSPIN and the selector network of INVASE/L2X/REAL-x. For **E1**, we add one layer with 200 nodes to the prediction architecture for all models and one layer with 100 nodes to the gating network of LLSPIN/LSPIN and selector network of INVASE/L2X/REAL-x. For **E5**, we use 3 hidden layers with 500, 100, 1 nodes each for the prediction network architecture. We use 1 layer with 100 nodes for the gating network of LLSPIN/LSPIN and the selector network of INVASE/L2X. For **E5**, we use leaky relu as the activation function for the prediction network of LSPIN.

To visualize the selected features for LSPIN (Fig. A.3), we repeated the optimization procedure 5 times and plotted the average gate values on the test set.

| | Batch Size | Number of Epochs | Learning Rate | $\lambda/k$ |
|---|---|---|---|---|
| **E1** | Full | 10000<br>(MLP/STG/LLSPIN/LSPIN/INVASE)<br>{1000,3000,5000,7000,9000} (L2X)<br>{500,1000,2000,5000,10000} (REAL-x) | 1e-1<br>(MLP/STG/LLSPIN/LSPIN)<br>1e-4 (INVASE)<br>[1e-5,1e-2] (L2X, REAL-x) | [2e-1,3e-1]/<br>{1,2,3,4,5,6,7} |
| **E2** | Full | 10000<br>(MLP/STG/LLSPIN/LSPIN/INVASE)<br>{1000,3000,5000,7000,9000} (L2X)<br>{500,1000,2000,5000,10000} (REAL-x) | 1e-1<br>(MLP/STG/LLSPIN/LSPIN)<br>1e-4 (INVASE)<br>[1e-5,1e-2] (L2X, REAL-x) | [0.1,0.15]/<br>{1,2,3,4,5,6,7} |
| **E3** | Full | 3000<br>(MLP/STG/LLSPIN/LSPIN/INVASE)<br>{1000,3000,5000,7000,9000} (L2X)<br>{500,1000,2000,5000,10000} (REAL-x) | 1e-1<br>(MLP/STG/LLSPIN/LSPIN)<br>1e-4 (INVASE)<br>[1e-5,1e-2] (L2X, REAL-x) | [0.15,0.2]/<br>{1,2,3,4,5,6,7} |
| **E4** | Full | {1000, 1200}<br>(MLP/STG/LLSPIN/LSPIN/INVASE)<br>{1000,3000,5000,7000,9000} (L2X)<br>{500,1000,2000,5000,10000} (REAL-x) | [3e-2,5e-2]<br>(MLP/STG/LLSPIN/LSPIN)<br>[1e-5,1e-4] (INVASE)<br>[1e-5,1e-2] (L2X, REAL-x) | [1.33,1.35]/<br>{1,2,3,4,5,6,7} |
| **E5** | Full | {2000,3000,5000,7000}<br>(MLP/STG/LLSPIN/LSPIN/INVASE)<br>{1000,3000,5000,7000,9000} (L2X) | [1e-2,1e-1]<br>(MLP/STG/LLSPIN/LSPIN)<br>[1e-5,1e-3] (INVASE)<br>[1e-5,1e-2] (L2X) | 1/<br>{1,2,3,4,5,6,7} |

Table B.2: Parameter settings and grids for the 5 nonlinear experiments. For LLSPIN/LSPIN/INVASE, we run 5 Optuna trials for **E1**, **E2**, **E3** and 20 trials for **E4** and **E5**. For L2X and REAL-x, we run 100 trials across datasets. Note that LLSPIN/LSPIN use SGD Optimizer and others use Adam optimizer. * We note that for Linear STG in **E1**, we observed that [2e-1,3e-1] is too high for $\lambda$ that no features are selected, therefore, we adjust the corresponding range to [1e-2,1e-1].

To evaluate a fair comparison of the prediction performance, we sparsify the input to the prediction network of REAL-x by multiplying the original information with the output from the selector network.

For TabNet, the grid of the regularization parameter $\lambda_{sparse}$ is set to {0.0001, 0.001,0.01,0.1,0.2,0.3,0.5}. The grids of $n_d$ and $n_a$ are both {8,16,24,32,64,128}. The grid of number of steps is {3,4,5,6,7,8,9,10}. The grid of $\gamma$ is {1.0,1.2,1.5,2.0}. The grid of learning rate is {0.005,0.01,0.02,0.025}, and the grid of momentum is {0.6,0.7,0.8,0.9,0.95,0.98}. The scheduler function is set to StepLR and the grids of the corresponding step_size and decay rate are {500,2000,8000,10000,20000} and {0.4,0.8,0.9,0.95}, respectively. The grid of max number of epochs is {4000,10000,20000}. The batch size is set to full batch and the virtual batch size is set to 5. The early stopping patience is set to 30 epochs. Other parameters are set to be default. We optimize the model on each dataset with 100 trials of Optuna.

### B.5. Real-world Datasets Details

Here we provide details for reproduction of the example presented in Section 5.3.

### B.5.1. PBMC Dataset Preprocessing Steps

The purified Peripheral Blood Mononuclear Cells (PBMC) dataset is collected from (Zheng et al., 2017), in which the data matrix has cells as samples, genes as features, and each entry represents the number of mRNAs expressed from the corresponding gene of that cell. This raw data matrix is first filtered (cells that have less than 400 expressed genes are excluded, and genes that are expressed in less than 100 cells are excluded) and normalized by the library size (total number of mRNAs expressed per cell). We then exclude the non-protein-coding genes and retain only cells that belong to the following $4$ cell types: memory T cells, naive T cells, regulatory T cells, naive cytotoxic T cells.

We use $34,115$ cells ($90\%$ of the data) to select the $2000$ most variable genes and use the remaining $3,791$ cells ($10\%$ of the data) with these $2000$ genes as the final processed dataset. Then, we split training/test/validation sets as described in the following subsection.

### B.5.2. Training Procedures

In this section, we introduce our training procedures for the real-world LSS datasets. Specifically, for the BASEHOCK, RELATHE, PCMAC, and PBMC datasets, $5\%$ of each dataset is set aside as a validation set. Let us denote the remaining $95\%$ of the data as $\bar{\boldsymbol{X}}$. We split $\bar{\boldsymbol{X}}$ into $5$ non-overlapping folds.

We train each model on $1$ fold of $\bar{\boldsymbol{X}}$ and test it on the remaining non-overlapping $4$ folds of $\bar{\boldsymbol{X}}$. The hyper-parameters are optimized on the validation set via Optuna (50 trials for the neural network-based methods and tree-based methods and five trials for INVASE due to long computation time) based on the model trained on a single fold and tested on the remaining $4$ non-overlapping folds. These (fixed) hyper-parameters are then used to train the model on the second fold and test it on the remaining $4$ non-overlapping folds. Similarly, we use these fixed hyper-parameters to train models for folds No.3, No.4, and No.5, and each time test these models on the remaining $4$ non-overlapping folds. This training and testing procedure is repeated for several regularization parameters; then, we report the best average performance for each method.

Since COLON and TOX-171 are of extreme LSS, we use a grid of regularization parameters for each method and identify the best average performance (test accuracy and the number of selected features) across ten runs (using 80% of the samples for training and 20% for testing).

The regularization parameters are tuned to select fewer than $50$ features, except for XGBoost when we applied it to the PBMC dataset. In this case, the minimum possible number of features chosen by XGBoost was $64$. For local methods including LLSPIN, LSPIN, and INVASE, L2X, TabNet, and REAL-x, the average (over the five folds) median (over the training samples) number of selected features is reported.

### B.5.3. Hyper-parameter Tuning and Settings

For LLSPIN/LSPIN/STG/Neural Network model, the prediction network architecture is set to 3 hidden layers with 100, 50, 30 neurons, respectively, for all the datasets. The gating network for LLSPIN/LSPIN models is set to one single layer with 500 neurons for BASEHOCK, RELATHE, PCMAC, PBMC datasets, and two hidden layers with 100 and 10 neurons respectively for the COLON and TOX-171 datasets. For INVASE/L2X/REAL-x, the network architecture is set to default. The predictor network has two hidden layers and 200 neurons on each layer, and the selector network has two hidden layers and 100 neurons on each layer.

For these neural network-based methods, the grids of regularization parameter $\lambda$ are listed in Table B.3, along with the grids of learning rate and epochs that are optimized via Optuna for the BASEHOCK, RELATHE, PCMAC, PBMC datasets, and the settings of learning rate and epochs in the COLON and TOX-171 datasets. We use full batch training for all neural network-based methods for all $6$ datasets, except for TabNet, where we set batch size and virtual batch size to be $100$ and $10$ for BASEHOCK/RELATHE/PCMAC/PBMC and $20$ and $4$ for COLON and TOX-171.

We studied the TOX-171 dataset, setting the grid of $\lambda$ of the STG models in the range of [1,2]. We observed that in this range, the number of features was too high; therefore we extended the range to [1,10] for the STG models.

For LASSO and SVC, the grid for their regularization parameter $c$ is set to [1e-3,1e-1] for the BASEHOCK, RELATHE, PCMAC, PBMC datasets, and [1e-2,1e3] for the COLON and TOX-171 datasets.

For Random Forest and XGBoost, we use number_of_estimators to replace the regularization parameter proposed in the previously training procedures. The grid of number_of_estimators is $\{1,5,10,20,30,50,100,200,500,1000\}$

| Dataset | Method | $\lambda$/$k$/$\lambda_{sparse}$ | Learning Rate | Number of Epochs |
|---|---|---|---|---|
| BASEHOCK RELATHE PCMAC PBMC | LLSPIN | [1,10] | [1e-2,1e-1] | $\{1000, 3000, 5000, 7000, 9000\}$ |
| | LSPIN | [1,10] | [1e-2,1e-1] | $\{1000, 3000, 5000, 7000, 9000\}$ |
| | STG (l) | [1,10] | [1e-1,2e-1] | $\{3000, 5000, 7000, 9000\}$ |
| | STG (n) | [1,10] | [1e-1,2e-1] | $\{3000, 5000, 7000, 9000\}$ |
| | Neural Net | None | [1e-2,1e-1] | $\{1000, 3000, 5000, 7000, 9000\}$ |
| | INVASE | $\{1, 5, 10\}$ | [1e-5,1e-4] | 10000 |
| | L2X | $\{1, 5, 10\}$ | [1e-5,1e-2] | $\{1000, 3000, 5000, 7000, 9000\}$ |
| | TabNet | $\{0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 0.5\}$ | [0.005,0.025] | $\{2000, 4000, 6000, 8000, 10000\}$ |
| | REAL-x | $\{10, 30, 50, 70, 90\}$ | [1e-5,1e-2] | $\{100, 200, 500, 1000, 2000\}$ |
| COLON | LLSPIN | [1,2] | 0.1 | 7000 |
| | LSPIN | [1,2] | 0.05 | 7000 |
| | STG (l) | [1,2] | 0.5 | 7000 |
| | STG (n) | [1,2] | 0.5 | 7000 |
| | Neural Net | None | 0.1 | 7000 |
| | INVASE | $\{1, 1.5, 2\}$ | 0.0001 | 10000 |
| | L2X | $\{1, 3, 5, 7, 9, 10\}$ | 0.0001 | 10000 |
| | TabNet | $\{0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 0.5\}$ | 0.0001 | 10000 |
| | REAL-x | $\{1, 5, 10, 30, 50\}$ | 0.0001 | 1000 |
| TOX-171 | LLSPIN | [1,2] | 0.1 | 7000 |
| | LSPIN | [1,2] | 0.05 | 7000 |
| | STG (l) | [1,10] | 0.5 | 7000 |
| | STG (n) | [1,10] | 0.5 | 7000 |
| | Neural Net | None | 0.1 | 7000 |
| | INVASE | $\{1, 1.5, 2\}$ | 0.0001 | 10000 |
| | L2X | $\{1, 3, 5, 7, 9, 10\}$ | 0.0001 | 10000 |
| | TabNet | $\{0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 0.5\}$ | 0.0001 | 10000 |
| | REAL-x | $\{1, 5, 10, 30, 50\}$ | 0.0001 | 1000 |

Table B.3: Parameter settings for the neural network based methods on the real-world data. Note that INVASE/L2X/TabNet/REAL-x use Adam Optimizer and others use SGD optimizer. For the regularization parameter $\lambda$, the grid size for LSPIN and STG models is 5 on BASEHOCK/RELATHE/PCMAC/PBMC datasets and 20 on COLON and TOX-171 datasets.

for both methods when we applied them to the BASEHOCK, RELATHE, PCMAC, PBMC datasets, and is $\{$1,2,3,4,5,8,10,15,20,25,30,40,50,60,70,80,100,200,500,1000$\}$ when we applied them to the COLON and TOX-171 datasets.

Other parameter settings for XGBoost are as follows: For the BASEHOCK, RELATHE, PCMAC, PBMC datasets, we optimize max_depth via Optuna with grid range [1,10]. For the COLON and TOX-171 datasets, we set max_depth to 10.

Other parameter settings for Random Forest are as follows: For the BASEHOCK, RELATHE, PCMAC, PBMC datasets, we optimize max_depth and minimum_samples_to_split via Optuna with grid range [1,10] and [2,50], respectively. For the COLON and TOX-171 datasets, we set max_depth to 10 and minimum_samples_to_split to 5.

To evaluate a fair comparison of the prediction performance, we sparsify the input to the prediction network of REAL-x by multiplying the original information with the output from the selector network.

For TabNet, the early stopping patience is set to 30 epochs. Other parameters are set to be Default.

In the MNIST experiment we use a batch size of 100 with a learning rate of 0.1 and train for 300 epochs. When $\lambda_2 > 0$, we use a warm up procedure where we first train the model for 200 epochs with $\lambda_2 = 0$ and then increase $\lambda_2$. We observe that this stabilizes the training procedure.

### B.6. Cox Proportional Hazard Models for Survival Analysis Details

Here we provide details for reproduction of the example presented in Section 5.4.

#### B.6.1. SEER Dataset Preprocessing Steps

The data for this study were collected from the Surveillance, Epidemiology, and End Results (SEER) public datasets (National Cancer Institute, 2019). Female patients, ages 25-85, diagnosed with histologically confirmed non-metastatic breast cancer between Jan 1, 2000, and Dec 31, 2016, are included. Patients with metastatic disease and those with missing data on stage, T grade, number of positive nodes, number of T nodes, vital status, and survival time are excluded. Only those patients who underwent surgery and had a known tumor size of less than 200 mm are included. Patients with bilateral breast cancer, inflammatory disease, and in-situ tumor are excluded. We use one-hot encoding for the categorical variables and drop features with unknown/unspecified values. Continuous variables are z-scored. We further add 3 random uniform variables as noise to the data. In total, we have $538,315$ patients and $55$ features after processing.

#### B.6.2. Training Procedures and Hyper-parameter Tuning and Settings

We apply a training procedure similar to the one we used to the BASEHOCK, RELATHE, PCMAC, PBMC datasets for the SEER data.

$5\%$ of each dataset is set aside as a validation set. Let us denote the remaining $95\%$ of the data as $\bar{\boldsymbol{X}}$. We split $\bar{\boldsymbol{X}}$ into $10$ non-overlapping folds.

We train each model on $1$ fold of $\bar{\boldsymbol{X}}$ and test it on the remaining non-overlapping $9$ folds of $\bar{\boldsymbol{X}}$. The hyper-parameters are optimized on the validation set via Optuna ($50$ Optuna trials for the neural network methods on the learning rate and epochs as in Table B.4) based on the model trained on a single fold and tested on the remaining $9$ non-overlapping folds. These (fixed) hyper-parameters are then used in training the model on the second fold and testing it on the remaining $9$ non-overlapping folds. Similarly, we use these fixed hyper-parameters to train models for folds No.3, No.4, No.5, ..., No.10, and each time test these models on the remaining $9$ non-overlapping folds. We then compute the average (over the ten folds) performance (test concordance index and the number of selected features). This training and testing procedure is repeated for several regularization parameters to produce the results in the interpolation plot Fig.4 (Left). We note that Random Survival Forest selects almost all the features over different n_estimators parameters as shown as an isolated interpolation point in Fig.4 (Left).

| Parameters | Range |
|---|---|
| learning rate (all Neural Network methods) | [1e-2,1] |
| epochs (all Neural Network methods) | $\{500, 1000, 2000\}$ |
| $\lambda$ (COX-LLSPIN, COX-LSPIN) | [1e-9,1e-5] |
| $\lambda$ (COX-STG(Linear/Nonlinear)) | [1e-3,1e-1] |
| $\alpha$ (COX-LASSO) | [1e-7,1] |
| n_estimators (Random Survival Forest) | {1,10,100,500,1000} |

Table B.4: Parameter settings for different models for the survival analysis example. For COX-LLSPIN/COX-LSPIN/COX-STG, the regularization parameter is $\lambda$. For COX-LASSO, the regularization parameter is $\alpha$. For Random Survival Forest, we use n_estimators to replace the regularization parameter.

In this example, we set the nonlinear neural network methods (COX-LSPIN, COX-STG(Nonlinear), DeepSurv) to 3 hidden layers with 100, 30, and 5 neurons, respectively. The linear neural network methods (COX-LLSPIN, COX-STG(Linear)) have no hidden layers. The gating network of LLSPIN/LSPIN is set to 1 hidden layer with 300 neurons. For the local methods, including COX-LLSPIN and COX-LSPIN, the average (over the $10$ folds) median (over the training samples) number of selected features is reported.

### B.7. Single Nucleus RNA-seq Dataset Details

#### B.7.1. Data Preprocessing and Split

Similar to the scRNA-seq data, in the data matrix, the samples are cells, the features are the genes, and each entry represents the number of mRNAs expressed from the corresponding gene of that cell. The cells are first filtered based on the number of genes that have non-zeros values (lower threshold is $500$ and upper threshold is $7500$) and then filtered based on the mitochondrial ratio ($10\%$). The data are imputed using ALRA (Linderman et al., 2022).

We randomly sample $1000$ cells of each type (Microglia cells and Oligodendrocyte Precursor Cells (OPC)), then use $50\%$ of the data to select $100$ most variable genes that are not correlated with *ITGAM* and *PDGFRA* as noisy genes. For the remaining data, we keep these $100$ genes along with *ITGAM* and *PDGFRA* as features and use $80\%/10\%/10\%$ as the train/validation/test split.

#### B.7.2. Hyper-parameter Tuning and Settings

For each model in Table 3, we optimize over a grid of the corresponding regularization parameter by minimizing the classification error (if two models have the same performance, the sparse one will be chosen). For LLSPIN/INVASE/REAL-x, the grid of $\lambda$ is $\{1, 10, 20, 30, 40, 50\}$. For L2X, the grid of $k$ is $\{1, 2, 3, 4, 5, 6, 7\}$. For TabNet, the grid of $\lambda_{sparse}$ is $\{0.5, 1, 5, 10, 20, 50\}$.

For LLSPIN, the prediction network is $2$ layers with $200$ and $100$ nodes each. The gating network is a $1$ layer with $100$ nodes. The learning rate is set to be $0.1$. The number of epochs is $2000$. The batch size is set to full batch training. The standard deviation ($\sigma$) of the Gaussian reparameterization is set to $1$.

For INVASE/L2X/REAL-x, the network architecture is set to default. The predictor network has two hidden layers and 200 neurons on each layer, and the selector network has two hidden layers and 100 neurons on each layer. The learning rate is set to be $1e-3$, the number of epochs is $2000$. To evaluate a fair comparison of the prediction performance, we sparsify the input to the prediction network of REAL-x by multiplying the original information with the output from the selector network.

For TabNet, the learning rate is set to $1e-3$, and the number of epochs is $2000$. The batch size is $100$, and the virtual batch size is $10$. The early stopping patience is $30$ epochs. Other parameters are set as default.

## C. Strengths and Limitations

The proposed model leads to an interpretable prediction model that can handle datasets of low sample size (LSS). Our results demonstrate that local sparsity tied with a linear model can be a robust classifier on real biological datasets. As for the societal impact, we don't know the effect of adversarial examples on the hazards model.

Currently, the sparsity of our model is tuned via a regularization parameter $\lambda_1$ while stability is tuned via $\lambda_2$ (see Eq. 5). In certain setting, tuning these parameters could be a demanding task, since it involves balancing with the main loss term. In the future, we plan to explore a more flexible mechanism for local feature selection. One possible way that this could be realized is using a concrete layer as proposed in (Abid et al., 2019).