FsNet: Feature Selection Network on High-dimensional Biological Data

Anonymous authors
Paper under double-blind review

Abstract

Biological data including gene expression data are generally high-dimensional and require efficient, generalizable, and scalable machine-learning methods to discover their complex nonlinear patterns. The recent advances in machine learning can be attributed to deep neural networks (DNNs), which excel in various tasks in terms of computer vision and natural language processing. However, standard DNNs are not appropriate for high-dimensional datasets generated in biology because they have many parameters, which in turn require many samples. In this paper, we propose a DNN-based, nonlinear feature selection method, called the feature selection network (FsNet), for high-dimensional and small number of sample data. Specifically, FsNet comprises a selection layer that selects features and a reconstruction layer that stabilizes the training. Because a large number of parameters in the selection and reconstruction layers can easily result in overfitting under a limited number of samples, we use two tiny networks to predict the large, virtual weight matrices of the selection and reconstruction layers. Experimental results on several real-world, high-dimensional biological datasets demonstrate the efficacy of the proposed method.

1 Introduction

The recent advancements in measuring devices for life sciences have resulted in the generation of large biological datasets, which are extremely important for many medical and biological applications, including disease diagnosis, biomarker discovery, drug development, and forensics (Li & Chen, 2014). Generally, such datasets are substantially high-dimensional (i.e., many features with small number of samples) and contain complex nonlinear patterns. Machine learning methods, including genome-wide association studies ($d > 10^5$, $n < 10^4$) and gene selection ($d > 10^4$, $n < 10^3$) (Marx, 2013), have been successfully applied to discover the complex patterns hidden in high-dimensional biological and medical data. However, most nonlinear models in particular deep neural networks (DNN) are difficult to train under these conditions because of the significantly high number of parameters. Hence, the following questions naturally arise: 1) are all the features necessary for building effective prediction models? and 2) what modifications are required in the existing machine-learning methods to efficiently process such high-dimensional data?

The answer to the first question is to select the most relevant features, thereby requiring an appropriate feature selection method (Ye et al., 2019; Ming & Ding, 2019; Liao et al., 2019). This problem, called feature selection, consists of identifying a smaller subset (i.e., smaller than the original dataset) that contains relevant features such that the subset retains the predictive capability of the data/model while eliminating the redundant or irrelevant features (Yamada et al., 2014; 2018a; Climente-González et al., 2019). Most state-of-the-art feature selection methods are based on either sparse-learning methods, including Lasso (Tibshirani, 1996), or kernel methods (Masaeli et al., 2010; Yamada et al., 2018b; 2014). These *shallow* approaches satisfactorily work in practice for biological data. However, sparse-learning models including Lasso are in general linear and hence cannot capture high-dimensional biological data. Kernel-based methods can handle the nonlinearlity, but it heavily depends on the choice of the kernel function. Thus, more flexibile approaches that can train an arbitrary nonlinear transformation of features are desired.

An approach to learning such a nonlinear transformation could be based on deep autoencoders (Vincent et al., 2010). However, deep autoencoders are useful for computer-vision and natural language processing tasks, wherein a large number of training samples are available. In contrast, for high-dimensional biological data, the curse of dimensionality prevents us from training such deep models without overfitting. Moreover, these models focus on building useful features rather than selecting features from data. The training of autoencoders for feature selection results in the discrete combinatorial optimization problem, which is difficult to train in an end-to-end manner.

To train neural networks on high-dimensional data without resulting in overfitting, several approaches were proposed. Widely used ones are based on random projection and its variants (Dahl et al., 2013; Wójcik & Kurdziel, 2019). However, their performances significantly depend on the random projection matrix, and their usability is limited to dimensionality reduction only. Therefore, they cannot be applied for feature selection. Another deep learning-based approach employs a concrete autoencoder (CAE) (Balin et al., 2019), which uses concrete random variables (Maddison et al., 2017) to select features without supervision. Although CAE is an unsupervised model with poor performance, it can be extended to incorporate a supervised-learning setup. However, we observed that this simple extension is not efficient because the large number of parameters in the first layer of CAE can easily result in overfitting under a limited number of samples.

To address these issues, we propose a non-linear feature selection network, called FsNet, for high-dimensional biological data. FsNet comprises a selection layer that uses concrete random variables (Maddison et al., 2017), which are the continuous variants of a one-hot vector, and a reconstruction layer that stabilizes the training process. The concrete random variable allows the conversion of the discrete optimization problem into a continuous one, enabling the backpropagation of gradients using the reparameterization trick. During the training period, FsNet selects a few features using its selection layer while maximizing the classification accuracy and minimizing the reconstruction error. However, owing to the large number of parameters in the selection and reconstruction layers, overfitting can easily occur under a limited number of samples. Therefore, to avoid overfitting, we propose using two tiny networks to predict the large, virtual weight matrices of the selection and reconstruction layers. Consequently, the size of the model is significantly reduced and the network can scale high-dimensional datasets on a resource-limited device/machine. Through experiments on various real-world datasets, we show that the proposed FsNet significantly outperforms CAE and the supervised counterpart thereof.

Contributions: Our contributions through this paper are as follows.

- We propose FsNet, an end-to-end trainable neural network based nonlinear feature selection, for high-dimensional data with small number of samples.
- FsNet compares favorably with the state-of-the-art nonlinear feature selection methods for high-dimensional data with small number of samples.
- The model size of FsNet is one to two orders magnitude smaller than that of a standard DNN model, including CAE (Balin et al., 2019).

2 Related Work

Here, we discuss the existing shallow/deep feature selection methods, along with their drawbacks.

Shallow, nonlinear feature selection: *Maximum relevance* is a simple but effective criterion of nonlinear feature selection (Guyon & Elisseeff, 2003). It uses mutual information and the Hilbert-Schmidt Independence Criterion (HSIC) to select the features associated with the outcome (Peng et al., 2005; Song et al., 2007). It is also called sure independence screening in the statistics community (Fan & Lv, 2008; Balasubramanian et al., 2013). However, because it tends to select redundant features, minimum redundancy maximum relevance (mRMR) feature selection was proposed (Peng et al., 2005). Notably, mRMR finds the subset of independent features that are maximally associated with the outcome by using mutual information between features and between each feature and the outcome. Recently, a kernel-based, convex variant of mRMR was proposed, called HSIC Lasso (Yamada et al., 2014; 2018a; Climente-González et al., 2019). They effectively perform nonlinear feature selection on high-dimensional data, producing simple models with parameters that can be easily estimated. However, their performances are limited by the simplicity of the models and depends on the choice of kernels.

DNNs for feature selection: DNNs are nonlinear, complex models that can address the aforementioned problems associated with kernel-based methods. They can be used for feature selection by adding a regularization term to the loss function, or by measuring the effect of an input feature on the target variable (Verikas & Bacauskiene, 2002). Elaborately, an extra feature scoring layer is added to perform element-wise multiplication on the features and score, and then they are entered as inputs into the rest of the network (Wang et al., 2014; Lu et al., 2018). However, DNNs do not select features during the training period, thereby resulting in a performance reduction after feature selection. Moreover, it is generally difficult to obtain a sparse solution using a stochastic gradient. CAE (Balin et al., 2019) addresses this problem by training an autoencoder that contains a feature selection layer with a concrete variable, which is a continuous relaxation of a one-hot vector. Recently, another end-to-end, supervised, feature selection method based on stochastic gates (STGs) was proposed (Yamada et al., 2020). It uses a continuously relaxed Bernoulli variable and performs better than the existing feature selection methods. However, these methods need to train a large number of parameters in the first layer, resulting in overfitting to the training data. Therefore, these approaches may not be appropriate for DNN models with high-dimensional data and a limited number of samples.

Training DNNs on high-dimensional data: The existing DNN-based methods can easily overfit to the high-dimensional biological data, as they suffer from the *curse-of-dimensionality* irrespective of *regularization constraints*. The biggest drawback of DNNs is that they need to have a large number of parameters in the first layers of the decoder and encoder. HashedNets (Chen et al., 2015) addressed this issue by exploiting the inherent redundancy in weights to group them into relatively fewer hash buckets and shared them with all its connections. However, the hash function groups the weights on the basis of their initial values instead of opting for a dynamic grouping, thereby reducing the options to arbitrarily learn weights. Diet Networks (Romero et al., 2017) used tiny networks to predict weight matrices. However, they are limited to the multilayer perceptron only for classification and not for feature selection. A DNN model, referred to as deep neural pursuit (DNP) (Liu et al., 2017), selects features from high-dimensional data with a small number of samples. It is based on changes in the average gradients with multiple dropouts by an individual feature. However, (Liu et al., 2017) reported that the performance of DNP significantly depends on the number of layers.

These issues render the existing approaches inefficient for processing biological data, thereby raising the need to develop a method for efficiently extracting features from biological data.

3 Problem Formulation

Let $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)^\top = (\boldsymbol{u}_1, \cdots, \boldsymbol{u}_d) \in \mathbb{R}^{n \times d}$ be the given data matrix, where $\boldsymbol{x} \in \mathbb{R}^d$ represents the sample vector with d number of features and $\boldsymbol{u} \in \mathbb{R}^n$ the feature vector with n number of samples. Let $\boldsymbol{y} = (y_1, \cdots, y_n)^\top \in \mathbb{R}^n$ be the target vector such that $y_i \in \mathcal{Y}$ represents the output for $\boldsymbol{x}_i$, where $\mathcal{Y}$ denotes the domain of the output vector $\boldsymbol{y}$, which is continuous for regression problems and categorical for classification problems. In this paper, we assume that the number of samples is significantly fewer than that of the dimensions (i.e., $n \ll d$).

The final goal of this paper is to train a neural-network classifier $f(\cdot) : \mathbb{R}^d \to \mathcal{Y}$, which simultaneously identifies a subset $\mathcal{S} \subseteq \mathcal{F} = \{1, 2 \cdots d\}$ of features of a specified size $|\mathcal{S}| = K \ll d$, where the subset can reproduce the remaining $\mathcal{F} \backslash \mathcal{S}$ features with minimal loss.

4 Proposed Method: FsNet

We here present the architecture and training of the proposed FsNet model for selecting nonlinear features from high-dimensional data.

4.1 FsNet Model

We aim to build an end-to-end, trainable, compact, feature selection model. Hence, we employ a concrete random variable (Maddison et al., 2017) to select features, and we also use the weight-predictor models used in Diet Networks to reduce the model size (Romero et al., 2017). We build FsNet, a simple but effective model (see Figure 1). As shown in Figure 1(A), although the selection and reconstruction layers have many connections, they are virtual layers whose weights are

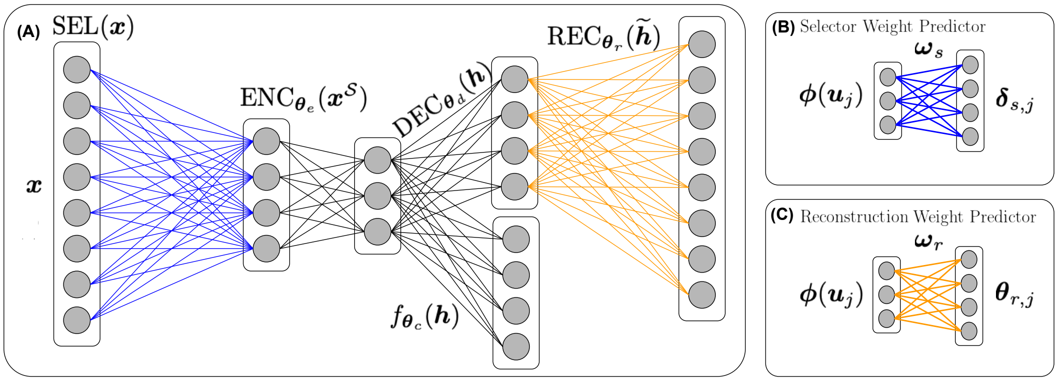

(A) $\mathrm{SEL}(\boldsymbol{x})$
$\boldsymbol{x}$
$\mathrm{ENC}_{\boldsymbol{\theta}_e}(\boldsymbol{x}^{\mathcal{S}})$
$\mathrm{DEC}_{\boldsymbol{\theta}_d}(\boldsymbol{h})$
$\mathrm{REC}_{\boldsymbol{\theta}_r}(\widetilde{\boldsymbol{h}})$
$f_{\boldsymbol{\theta}_c}(\boldsymbol{h})$
(B) Selector Weight Predictor
$\boldsymbol{\omega}_s$
$\boldsymbol{\phi}(\boldsymbol{u}_j)$
$\boldsymbol{\delta}_{s,j}$
(C) Reconstruction Weight Predictor
$\boldsymbol{\omega}_r$
$\boldsymbol{\phi}(\boldsymbol{u}_j)$
$\boldsymbol{\theta}_{r,j}$

Figure 1: (A) Architecture of FsNet. (B) and (C) are the weight-predictor networks for the selection and reconstruction layers, respectively.

predicted from significantly small networks, as shown in Figures 1(B) & (C), respectively. The weight-predictor networks (B) and (C) are trained on the feature embeddings.

The optimization problem of FsNet is given by

$$\min_{\boldsymbol{\Theta}} \sum_{i=1}^{n} \mathrm{Loss}(y_i, f_{\boldsymbol{\theta}_c}(\mathrm{ENC}_{\boldsymbol{\theta}_e}(\boldsymbol{x}_i^{\mathcal{S}}))) + \lambda \sum_{i=1}^{n} \|\boldsymbol{x}_i - \mathrm{REC}_{\boldsymbol{\theta}_r}(\mathrm{DEC}_{\boldsymbol{\theta}_d}(\mathrm{ENC}_{\boldsymbol{\theta}_e}(\boldsymbol{x}_i^{\mathcal{S}})))\|_2^2, \quad (1)$$

where $\mathrm{Loss}(y, f_{\boldsymbol{\theta}_c})$ denotes the categorical cross-entropy loss (between y and $f_{\boldsymbol{\theta}_c}$), $\|\cdot\|_2$ the ℓ_2 norm, $\lambda \geq 0$ the regularization parameter for the reconstruction loss, $\boldsymbol{\Theta}$ all the parameters in the model, $\mathrm{SEL}(\cdot)$ the selection layer, $\boldsymbol{x}_i^{\mathcal{S}} = \mathrm{SEL}(\boldsymbol{x}_i)$, $\mathrm{ENC}(\cdot)$ the encoder network, $\mathrm{DEC}(\cdot)$ the decoder network, and $\mathrm{REC}(\cdot)$ the reconstruction layer. The *pseudocode* for the training of FsNet is provided in Algorithm 2 in the appendix.

Selection Layer (Train): We first describe the selection layer, which is used to select important features in an end-to-end manner. The feature selection problem is generally a combinatorial problem, but it is difficult to train in an end-to-end manner because it breaks the propagation of the gradients. To overcome this obstacle, a concrete random variable (Maddison et al., 2017), which is a continuous relaxation of a discrete one-hot vector, can be used for the training, as it computes the gradients using the reparameterization trick. Specifically, selecting the k-th feature of the input $\boldsymbol{x}$ can be expressed as $\boldsymbol{x}^{(k)} = \boldsymbol{e}_k^\top \boldsymbol{x}$, where $\boldsymbol{e}_k \in \mathbb{R}^d$ denotes the one-hot vector whose k-th feature is 1 and 0 otherwise. The concrete variables for the k^{th} neuron in the selection layer are defined as follows:

$$\boldsymbol{\mu}^{(k)} = \frac{\exp\left((\log \boldsymbol{\delta}_s^{(k)} + \boldsymbol{g})/\tau\right)}{\sum_{j=1}^{d} \exp\left((\log \delta_{sj}^{(k)} + g_j)/\tau\right)}, k = 1, 2, \ldots, K, \quad (2)$$

where $\boldsymbol{g} \in \mathbb{R}^d$ is drawn from the Gumbel distribution. Additionally, τ denotes the temperature that controls the extent of the relaxation, K the number of selected features, and $\boldsymbol{\Delta}_s = (\boldsymbol{\delta}_{s,1}, \ldots, \boldsymbol{\delta}_{s,d}) = (\boldsymbol{\delta}_s^{(1)}, \ldots, \boldsymbol{\delta}_s^{(K)})^\top \in \mathbb{R}^{K \times d}$, $\boldsymbol{\delta}_s^{(k)} \in \mathbb{R}_{>0}^K$ is the model parameter for concrete variables. Notably, $\boldsymbol{\mu}^{(k)}$ becomes a one-hot vector when $\tau \to 0$.

Using the concrete variables $\boldsymbol{M} = (\boldsymbol{\mu}^{(1)}, \boldsymbol{\mu}^{(2)}, \ldots, \boldsymbol{\mu}^{(K)})^\top$, the feature selection process can be simply written by using matrix multiplications as follows:

$$\mathrm{SEL}(\boldsymbol{x}) = \boldsymbol{M}\boldsymbol{x}.$$

Because the feature selection process can be written by using matrix multiplications, it can be trained in an end-to-end manner. However, the number of parameters in the selection layer is $O(dK)$; it depends on the size of the input layer d and the number of neurons in the selection layer K. Thus, for high-dimensional data, the number of model parameters can be high, resulting in overfitting under a limited number of samples n. We address both the issues by using a tiny weight-predictor network $\varphi_{\boldsymbol{\omega}_s}(\cdot) : \mathbb{R}^b \to \mathbb{R}_{>0}^K$ to predict the weights $\boldsymbol{\delta}_{s,j} = \varphi_{\boldsymbol{\omega}_s}(\boldsymbol{\phi}(\boldsymbol{u}_j))$ (see Figure 1(B)), where

$\phi(\boldsymbol{u}_j) \in \mathbb{R}^b$ is the embedding representation of feature j and $b \leq n$ the size of the embedding representation. Specifically, the feature embedding $\phi(\boldsymbol{u}_j)$ for the j^{th} feature vector used for training the weight-predictor networks is defined as $\phi(\boldsymbol{u}_j) = \boldsymbol{\rho}_j \odot \boldsymbol{\nu}_j$, where $\odot$ denotes elementwise multiplication, whereas $\boldsymbol{\rho}_j$ and $\boldsymbol{\nu}_j$ denote the frequencies and means of the histogram bins of feature $\boldsymbol{u}_j$, respectively. In this paper, we use $\boldsymbol{\delta}_{s,j} = \text{softmax}(\boldsymbol{W}_{\boldsymbol{\omega}_s}\phi(\boldsymbol{u}_j))$, where $\boldsymbol{W}_{\boldsymbol{\omega}_s} \in \mathbb{R}^{K \times b}$ is the model parameter for the tiny network. Over epochs, $\boldsymbol{\mu}^{(k)}$ will converge to a one-hot vector. Notably, the model parameter $\boldsymbol{\Delta} \in \mathbb{R}^{K \times d}$ depends on the input dimension d. However because the model size of the weight-predictor network depends on $b \ll d$, we can significantly reduce the network model size using the predictor network. Moreover, the tiny weight-predictor network can also be trained in an end-to-end manner.

Selection Layer (Inference): For inference, we can replace the concrete variables with a set of feature indices. Consequently, the inference becomes faster than before, as we need not compute tiny networks. However, if we simply use the argmax function, it tends to select redundant features, and thus the prediction performance can be degraded. Therefore, we propose the *unique_argmax* function to select non-redundant features and then use the non-redundant feature set for inference. The K best and unique features are selected from the estimated $\boldsymbol{M}$ as $\mathcal{S} =$ uargmax$(\boldsymbol{M}^\top)$. Subsequently, for inference, we use $\boldsymbol{x}^{\mathcal{S}} \in \mathbb{R}^K$ as an input of the encoder network. Although this is a heuristic approach, it works satisfactorily in practice.

Algorithm 1 Unique argmax function uargmax

```
Input: matrix $\boldsymbol{A} \in \mathbb{R}_+^{d \times K}$, with $d$ rows and $K$ cols
Output: selected indices $\mathcal{S}$
$\mathcal{S} \leftarrow \{\}$
for $i = 0 - K$ do
    $(x, y) \leftarrow$ index of max value in $\boldsymbol{A}$
    $\mathcal{S} \leftarrow \mathcal{S} \cup x$
    $A.row(x) \leftarrow \mathbf{0}$
    $A.col(y) \leftarrow \mathbf{0}$
end for
```

Encoder Network: The goal of the encoder network $\text{ENC}_{\boldsymbol{\theta}_e}(\cdot) : \mathbb{R}^K \rightarrow \mathbb{R}^h$ is to obtain a low-dimensional hidden representation $\boldsymbol{h} \in \mathbb{R}^h$ from the output of the selection layer $\boldsymbol{x}^{\mathcal{S}}$. The encoder network is expressed as follows:

$$\text{ENC}_{\boldsymbol{\theta}_e}(\boldsymbol{x}^{\mathcal{S}}) = \sigma(\boldsymbol{W}_{L_e}^{(e)}\sigma(\cdots \boldsymbol{W}_2^{(e)}\sigma(\boldsymbol{W}_1^{(e)}\boldsymbol{x}^{\mathcal{S}})\cdots), \tag{3}$$

where $\boldsymbol{x}^{\mathcal{S}} = \text{SEL}(\boldsymbol{x})$ denotes the output of the selection layer, $\boldsymbol{\theta}_e = \{\boldsymbol{W}_\ell^{(e)}\}_{\ell=1}^{L_e}$ the weight matrix, L_e the number of layers in the encoder network, and $\sigma(\cdot)$ an activation function.

Classifier Network: The classifier network $f_{\boldsymbol{\theta}_c}(\cdot) : \mathbb{R}^h \rightarrow \mathcal{Y}$ predicts the final output from the hidden representation $\boldsymbol{h} = \text{ENC}_{\boldsymbol{\theta}_e}(\boldsymbol{x}^{\mathcal{S}})$ as follows:

$$f_{\boldsymbol{\theta}_c}(\boldsymbol{h}) = \text{softmax}(\boldsymbol{W}_{L_y}^{(y)}\sigma(\cdots \boldsymbol{W}_2^{(y)}\sigma(\boldsymbol{W}_1^{(y)}\boldsymbol{h})\cdots), \tag{4}$$

where $\boldsymbol{\theta}_c = \{\boldsymbol{W}_\ell^{(y)}\}_{\ell=1}^{L_y}$, and L_y denotes the number of layers in the classifier network.

Decoder Network: Generally, a decoder function is employed to reconstruct the original output. However, in this paper, the decoder function $\text{DEC}_{\boldsymbol{\theta}_d}(\cdot) : \mathbb{R}^h \rightarrow \mathbb{R}^{h'}$ computes another hidden representation $\widetilde{\boldsymbol{h}} \in \mathbb{R}^{h'}$ and defines the last reconstruction layer separately. The decoder function is defined as follows:

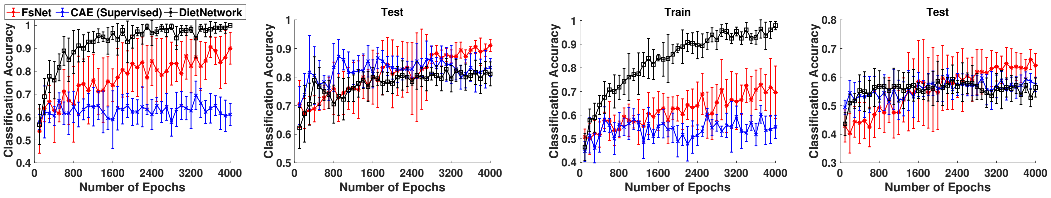
$$\text{DEC}_{\boldsymbol{\theta}_d}(\boldsymbol{h}) = \sigma(\boldsymbol{W}_{L_d}^{(d)}\sigma(\cdots \boldsymbol{W}_2^{(d)}\sigma(\boldsymbol{W}_1^{(d)}\boldsymbol{h})\cdots). \tag{5}$$

where $\boldsymbol{h} = \text{ENC}_{\boldsymbol{\theta}_e}(\boldsymbol{x}^{\mathcal{S}})$, $\boldsymbol{\theta}_d = \{\boldsymbol{W}_\ell^{(d)}\}_{\ell=1}^{L_d}$, and L_d denotes the number of layers in the decoder network.

Reconstruction Layer: To reconstruct the original high-dimensional feature $\boldsymbol{x}$, it must have $O(dh')$ parameters and depend on the dimension d. Thus, in a manner similar to the selection layer, we use a tiny network to predict the model parameters. The reconstruction layer is expressed as follows:

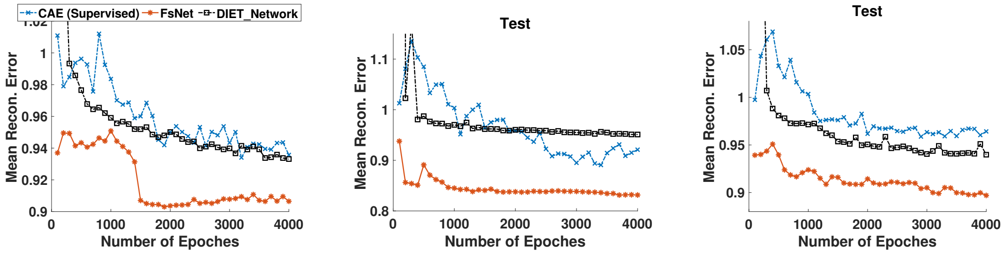
$$\text{REC}_{\boldsymbol{\theta}_r}(\widetilde{\boldsymbol{h}}) = \boldsymbol{W}^{(r)}\widetilde{\boldsymbol{h}}, \tag{6}$$

where $\widetilde{\boldsymbol{h}} = \text{DEC}_{\boldsymbol{\theta}_d}(\boldsymbol{h})$, $\boldsymbol{\theta}_r = \boldsymbol{W}^{(r)} \in \mathbb{R}^{d \times h'}$, and $[\boldsymbol{W}^{(r)^\top}]_j = \boldsymbol{\varphi}_{\boldsymbol{\omega}_r}(\phi(\boldsymbol{u}_j))$ denotes the virtual weights of the j^{th} row in the reconstruction layer. The tiny network $\boldsymbol{\varphi}_{\boldsymbol{\omega}_r}(\cdot) : \mathbb{R}^b \rightarrow \mathbb{R}^{h'}$ is trained on $\phi(\boldsymbol{u}_j) \in \mathbb{R}^b$ to predict the weights that connect the j^{th} row of the reconstruction layer to all the h' neurons of the last layer of the decoder network. In this paper, we use $[\boldsymbol{W}^{(r)^\top}]_j = \tanh(\boldsymbol{W}_{\boldsymbol{\omega}_r}\phi(\boldsymbol{u}_j))$, where $\boldsymbol{W}_{\boldsymbol{\omega}_r} \in \mathbb{R}^{h' \times b}$ is the model parameter for the tiny network.



(A) ALLAML (B) CLL_SUB

Figure 2: Comparison among FsNet, supervised CAE, and Diet Network for mean training and testing accuracies over the epochs. For the neural-network-based approaches, we set the model parameters to $b = 10$ and $K = 10$. (See all the experimental results in Figure 5).



(A) ALLAML (B) CLL_SUB (C) GLI_85

Figure 3: Comparison among the proposed FsNet and existing supervised CAE approaches in terms of the mean test reconstruction error over the epochs. (See the appendix for all the data results).

5 Empirical Evaluation

Here, we compare FsNet with several baselines using benchmark and the real metagenome dataset.

5.1 Setup

We compared FsNet with CAE (Balin et al., 2019), which is a unsupervised, neural-network-based, feature selection method, Diet Networks (Romero et al., 2017), HSIC Lasso (Yamada et al., 2014; 2018a; Climente-González et al., 2019), and mRMR (Peng et al., 2005). Notably, CAE and HSIC Lasso are state-of-the-art, nonlinear feature selection methods, which are deep and shallow, respectively. FsNet and CAE (Balin et al., 2019) were run on a Linux server with an Intel Xeon CPU Xeon(R) CPU E5-2690 v4 @ 2.60 GHz processor, 256 GB RAM, and NVIDIA P100 graphics card. HSIC Lasso (Yamada et al., 2014) and mRMR (Peng et al., 2005) were executed on a Linux server with an Intel Xeon CPU E7-8890 v4 2.20 GHz processor and 2 TB RAM.

For FsNet and CAE, we conducted experiments on all the datasets using a fixed architecture, defined as $[d \to K \to 64 \to 32 \to 16(\to |\mathcal{Y}|) \to 32 \to 64 \to d]$, where d and $|\mathcal{Y}|$ are data dependent, and $K \in \{10, 50\}$. Each hidden layer uses the *leakyReLU* activation function and *dropout* regularization with a dropout rate of 0.2. We implemented FsNet in *keras* and used the *RMSprop* optimizer for all the experiments. For the regularization parameter λ, we used $\lambda = 1$ for all the experiments. We performed the experiments with 4000 epochs at a learning rate of $\eta = 10^{-3}$, initial temperature $\tau_0 = 10$, and end temperature $\tau_E = 0.01$ in the annealing schedule for all the experiments.

5.2 Benchmark Dataset

We used six high-dimensional datasets from biological classification problems[1]. Table 4 in the appendix lists the relevant details of these datasets. The performance was evaluated on the basis of four parameters: classification accuracy, reconstruction error, mutual information between the selected features, and model size. Because neither HSIC Lasso nor mRMR could directly classify the samples, we used a support vector machine (SVM) (with a radial basis function) trained on the selected features. As CAE is an unsupervised method, we added a softmax layer to its loss function to ensure a fair comparison; the resulting model is henceforth referred to as supervised

[1] Publicly available at `http://featureselection.asu.edu/datasets.php`

Table 1: Comparison of the mean testing accuracy among FsNet, supervised CAE, HSIC Lasso (HSIC), and mRMR with $K = 10$ and $K = 50$. Moreover, we report SVM and Diet Networks. * The pyMRMR package, which is a wrapper of the original code, returns a memory error, and we could not execute the models on these datasets.

	$K = 10$				$K = 50$				All features	
Dataset	FsNet	CAE	HSIC	mRMR	FsNet	CAE	HSIC	mRMR	SVM	Diet-net
ALLAML	**0.911**	0.833	0.899	0.848	0.922	**0.936**	0.917	0.919	0.819	0.811
CLL_SUB	**0.640**	0.575	0.604	N/A*	0.582	0.556	**0.680**	N/A*	0.569	0.564
GLI_85	0.874	**0.884**	0.831	N/A*	0.795	0.822	**0.829**	N/A*	0.759	0.842
GLIOMA	**0.624**	0.584	0.595	0.564	0.624	0.604	0.672	**0.693**	0.628	0.712
Prostate_GE	0.871	0.835	**0.924**	0.871	0.878	0.884	0.926	**0.933**	0.846	0.753
SMK_CAN	**0.695**	0.680	0.660	0.620	0.641	0.667	**0.684**	0.668	0.699	0.665

CAE. Because RMSprop is a stochastic optimizer, all the results reported are the means of 20 runs on random splits of the datasets.

Classification accuracy: Figure 2 compares the training and testing behaviors of FsNet and supervised CAE for embedding size $b = 10$ and number of selected features $K = 10$. The results across the datasets show that FsNet can learn better than supervised CAE owing to its reduced number of parameters. The classification performance of FsNet for 10 selected features is comparable or superior to that of the SVM and Diet Networks for all the features across all the datasets. Similarly, the comparable performances of the proposed FsNet for 10 selected features and Diet-Network with all the features across the datasets illustrate that using a concrete random variable for the continuous relaxation of the discrete feature selection objective does not significantly change the objective function. Additionally, the correlation between the testing and training accuracies of FsNet demonstrates its generalization capability in comparison to supervised CAE, which seems to be overfitted under such high-dimensional data with a limited number of samples.

Table 1 presents the testing accuracies of the feature selection methods for various numbers of features selected on the six datasets. The experiments show that FsNet performs consistently better than supervised CAE, HSIC Lasso, mRMR, and Diet Networks for $K = 10$. However, the performance of neural-network-based models deteriorates when the number of features K increases. This is because as the number of parameters increases, the training of the model becomes increasingly difficult. Overall, FsNet tends to outperform the baselines even when the number of selected features is small ($K = 10$), and this is a satisfactory property of FsNet.

The selected features are highly predictive of the target variable. However, they represent the rest of the features in the dataset, as can be seen from the reconstruction error introduced in producing the original features from selected features (see Figure 3). FsNet achieves a more competitive reconstruction error than supervised CAE and Diet Network on all the datasets.

Model-size comparison: The number of parameters in the selection layer of supervised CAE is $O(dK)$, whereas in FsNet, the weight-predictor network of the selection layer has $O(bK)$ parameters. Similarly, the number of parameters in the reconstruction layer of supervised CAE is $O(dh')$, whereas in FsNet, the weight-predictor network of the reconstruction layer has $O(bh')$ parameters. The model compression ratio (CR) for FsNet with respect to supervised CAE is $CR = \frac{|\theta_s|+|\theta_r|+s}{|\omega_s|+|\omega_r|+s} = \frac{dh+h'd+s}{bh+h'b+s} = O\left(\frac{d}{b}\right)$, where $s = |\theta_e| + |\theta| + |\theta_d|$ denotes the number of parameters in the rest of the network. Thus, FsNet has $\approx \frac{d}{b}$ times fewer parameters than supervised CAE.

Table 2: Model-size comparison between supervised CAE and FsNet [2] (in KBs) at $K = 10$. Because FsNet predicts the model parameter by using a fixed-sized neural network, its model size is the same for all the datasets.

Dataset	FsNet	CAE	Compression ratio
ALLAML	108	4280	39.6
CLL_SUB	108	6748	62.5
GLI_85	108	13160	121.9
GLIOMA	108	2704	25.0
Prostate_GE	108	3600	33.3
SMK_CAN	108	11820	109.4

Table 2 lists the model sizes[2] in kilobytes (KBs) for FsNet and supervised CAE. The results show that FsNet can significantly reduce its model size according to the number of selected features (K) and size of the feature embedding (b). FsNet compresses the model size by 25–122 folds in com-

[2] Model size figures are the size of the *keras* model on the disk.

parison to supervised CAE. This reduction in the model size of FsNet is due to the use of tiny weight-predictor networks in the fat selection and reconstruction layers.

Minimum redundancy: The minimum redundancy criterion is important to measure the usefulness of the selected features. According to this criterion, the selected features should have minimum dependencies between themselves. We used the average mutual information between all the pairs of the selected features to compare the validity of the features selected by FsNet and CAE, respectively. The average mutual information is defined as follows: $\hat{I}(\mathcal{S}) = \frac{2}{K(K-1)} \sum_{i,j \in \mathcal{S}, j>i} I(X_i, X_j)$, where $I(X_i, X_j)$ denotes the mutual information between features i and j in the selected set $\mathcal{S}$.

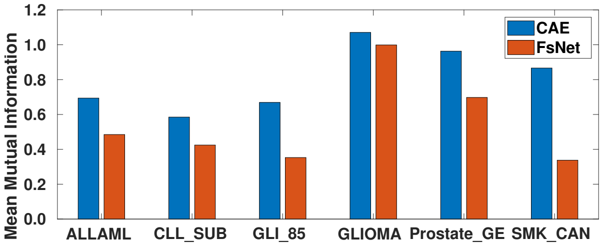


Figure 4: Comparison in terms of the average mutual information between the features selected by CAE and FsNet, respectively. The lower, the better.

As shown in Figure 4, compared with CAE, the average mutual information between the features selected by FsNet is significantly lower on all the datasets. This shows that compared with CAE, FsNet more effectively selects the features with minimum redundancy owing to the use of *unique_argmax* functions in the selection layer.

5.3 Application to inflammatory bowel disease

We studied a metagenome dataset (Lloyd-Price et al., 2019), which contains information regarding the gut bacteria of 359 healthy individuals and 958 patients with inflammatory bowel disease. Specifically, 7 547 features are KEGG orthology accession numbers, which represent molecular functions to which reads from the guts of samples guts are mapped. We included three additional features: age, sex, and race.

Table 3: Classification accuracy of different methods on the metagenome dataset on inflammatory bowel disease.

	Accuracy	
Method	$K = 10$	$K = 50$
FsNet	0.999 ± 0.002	0.994 ± 0.016
CAE	0.999 ± 0.002	0.983 ± 0.033
HSIC Lasso (B=10)	0.945 ± 0.003	0.962 ± 0.002
HSIC Lasso (B=20)	0.939 ± 0.004	0.959 ± 0.003
mRMR	0.941 ± 0.004	0.955 ± 0.003
SVM	0.914 ± 0.003	
Diet-networks	0.999 ± 0.002	

We selected 10 or 50 features on this dataset using FsNet, CAE, HSIC Lasso, STG, and mRMR. For HSIC Lasso, as the number of samples was high, we employed the block HSIC Lasso (Climente-González et al., 2019), where B denotes the tuning parameter of the block HSIC Lasso, and $B = n$ is equivalent to the standard HSIC Lasso (Yamada et al., 2014). The DNN based apporaches outperformed shallow methods. FsNet and CAE could achieve perfect prediction accuracy with only 10 features. Moreover, the compression ratio between FsNet and CAE is 21.41, and thus we conclude that FsNet can obtain preferable performance with much less number of parameters for high-dimensional data. This result indicates that DNN based methods can replace kernel methods even for high-dimensional data.

6 Conclusions

We proposed FsNet, which is an end-to-end trainable, deep learning-based, feature selection method for high-dimensional data with a small number of samples. FsNet can select unique features by using a concrete random variable. Using weight-predictor functions and a reconstruction loss, it not only required few parameters but also stabilized the model and made it appropriate for training with a limited number of samples. The experiments on several high-dimensional biological datasets demonstrated the robustness and superiority of FsNet for feature selection in the chosen settings. Moreover, we evaluated the proposed FsNet on a real-life metagenome dataset, and FsNet outperformed the existing shallow models.

REFERENCES

Krishnakumar Balasubramanian, Bharath K. Sriperumbudur, and Guy Lebanon. Ultrahigh dimensional feature screening via RKHS embeddings. In *AISTATS*, 2013.

Muhammed Fatih Balin, Abubakar Abid, and James Y. Zou. Concrete autoencoders: Differentiable feature selection and reconstruction. In *ICML*, 2019.

Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *ICML*, 2015.

Héctor Climente-González, Chloé-Agathe Azencott, Samuel Kaski, and Makoto Yamada. Block HSIC Lasso: model-free biomarker detection for ultra-high dimensional data. *Bioinformatics*, 35 (14):i427–i435, 07 2019.

G. E. Dahl, J. W. Stokes, Li Deng, and Dong Yu. Large-scale malware classification using random projections and neural networks. In *ICASSP*, 2013.

Jianqing Fan and Jinchi Lv. Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(5):849–911, 2008.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

Yixue Li and Luonan Chen. Big biological data: Challenges and opportunities. *Genomics, Proteomics & Bioinformatics*, 12(5):187–189, 2014.

Shuangli Liao, Quanxue Gao, Feiping Nie, Yang Liu, and Xiangdong Zhang. Worst-case discriminative feature selection. In *IJCAI*, 2019.

Bo Liu, Ying Wei, Yu Zhang, and Qiang Yang. Deep neural networks for high dimension, low sample size data. In *IJCAI*, 2017.

Jason Lloyd-Price, Cesar Arze, Ashwin N Ananthakrishnan, Melanie Schirmer, Julian Avila-Pacheco, Tiffany W Poon, Elizabeth Andrews, Nadim J Ajami, Kevin S Bonham, Colin J Brislawn, et al. Multi-omics of the gut microbial ecosystem in inflammatory bowel diseases. *Nature*, 569(7758):655–662, 2019.

Yang Young Lu, Yingying Fan, Jinchi Lv, and William Stafford Noble. DeepPINK: reproducible feature selection in deep neural networks. In *NeurIPS*, 2018.

C. J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*, 2017.

Vivien Marx. The big challenges of big data. *Nature*, 498(7453), 2013.

Mahdokht Masaeli, Glenn Fung, and Jennifer G. Dy. From transformation-based dimensionality reduction to feature selection. In *ICML*, 2010.

Di Ming and Chris Ding. Robust flexible feature selection via exclusive L21 regularization. In *IJCAI*, 2019.

Hanchuan Peng, Fuhui Long, and Chris H. Q. Ding. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE TPAMI*, 27(8):1226–1238, 2005.

Adriana Romero, Pierre Luc Carrier, et al. Diet networks: Thin parameters for fat genomics. In *ICLR*, 2017.

Le Song, Alex Smola, Arthur Gretton, Karsten M Borgwardt, and Justin Bedo. Supervised feature selection via dependence estimation. In *Proceedings of the 24th international conference on Machine learning*, pp. 823–830, 2007.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal Stat. Society*, 58(1): 267–288, 1996.

Antanas Verikas and Marija Bacauskiene. Feature selection with neural networks. *Pattern Recognition Letters*, 23(11):1323–1335, 2002.

Pascal Vincent, Hugo Larochelle, et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 11:3371–3408, 2010.

Qian Wang, Jiaxing Zhang, Sen Song, and Zheng Zhang. Attentional neural network: Feature selection using cognitive feedback. In *NIPS*, 2014.

Piotr Iwo Wójcik and Marcin Kurdziel. Training neural networks on high-dimensional data using random projection. *PAA*, 22(3):1221–31, 2019.

Makoto Yamada, Wittawat Jitkrittum, Leonid Sigal, Eric P Xing, and Masashi Sugiyama. High-dimensional feature selection by feature-wise kernelized lasso. *Neural computation*, 26(1):185–207, 2014.

Makoto Yamada, Jiliang Tang, et al. Ultra high-dimensional nonlinear feature selection for big biological data. *IEEE TKDE*, 30(7):1352–1365, 2018a.

Makoto Yamada, Yuta Umezu, Kenji Fukumizu, and Ichiro Takeuchi. Post selection inference with kernels. In *AISTATS*, 2018b.

Yutaro Yamada, Ofir Lindenbaum, Sahand Negahban, and Yuval Kluger. Feature selection using stochastic gates. *ICML*, 2020.

Xiucai Ye, Hongmin Li, Akira Imakura, and Tetsuya Sakurai. Distributed collaborative feature selection based on intermediate representation. In *IJCAI*, 2019.

Appendix

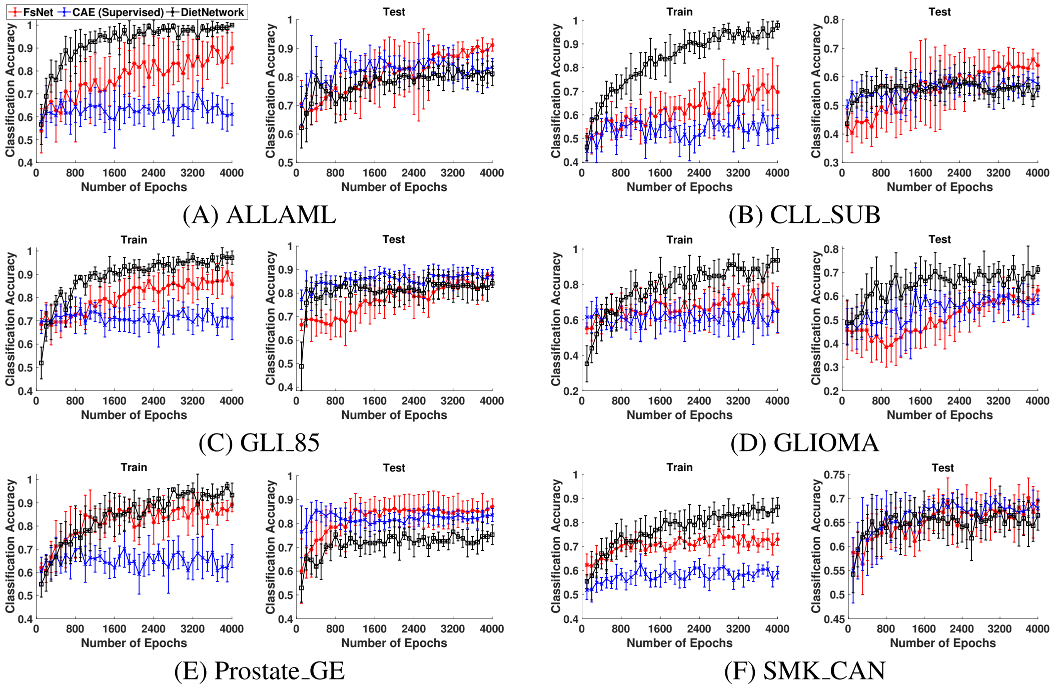
Algorithm 2 Training of FsNet

Input: data matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, output labels $y \in \{1, \cdots, L\})$, K target number of features, encoder network $\text{ENC}_{\boldsymbol{\theta}_e}(\cdot)$, decoder network $\text{DEC}_{\boldsymbol{\theta}_d}(\cdot)$, reconstruction function $\text{REC}_{\boldsymbol{\theta}_r}(\cdot)$, classification network $f_{\boldsymbol{\theta}_c}(\cdot)$, weight prediction networks $\boldsymbol{\varphi}_{\boldsymbol{\omega}_s}(\cdot)$ & $\boldsymbol{\varphi}_{\boldsymbol{\omega}_s}(\cdot)$, learning rate η, start temperature τ_0, end temperature τ_E, and number of epochs E
Output: set of selected features $\mathcal{S}$, model parameters $\boldsymbol{\Theta}$

```
Initialize $\boldsymbol{\Theta} = \{\boldsymbol{\omega}_s, \boldsymbol{\theta}_e, \boldsymbol{\theta}_d, \boldsymbol{\omega}_r, \boldsymbol{\theta}_c\}$.
for $e \in \{1, \cdots, E\}$ do
    Update the temperature $\tau = \tau_0(\tau_E/\tau_0)^{e/E}$
    $(\boldsymbol{\delta}_{s,1}, \cdots \boldsymbol{\delta}_{s,d}) \leftarrow (\boldsymbol{\varphi}_{\boldsymbol{\omega}_s}(\boldsymbol{\phi}(\boldsymbol{u}_1)) \cdots \boldsymbol{\varphi}_{\boldsymbol{\omega}_s}(\boldsymbol{\phi}(\boldsymbol{u}_d)))$
    $\boldsymbol{\mu}^{(k)} \leftarrow$ Concrete($\boldsymbol{\theta}_{\boldsymbol{s}}^{(k)}, \tau$) using (2)
    $\boldsymbol{M} \leftarrow (\boldsymbol{\mu}^{(1)}, \cdots, \boldsymbol{\mu}^{(K)})^\top$
    $\mathcal{S} \leftarrow$ uargmax($\boldsymbol{M}^\top$)
    $\boldsymbol{h} \leftarrow \begin{cases} \text{ENC}_{\boldsymbol{\theta}_e}(\boldsymbol{M}\boldsymbol{x}_i) & \text{if training,} \\ \text{ENC}_{\boldsymbol{\theta}_e}(\boldsymbol{x}^{\mathcal{S}}) & \text{inference} \end{cases}$
    $\widehat{y} \leftarrow f_\theta(\boldsymbol{h})$
    $\widetilde{\boldsymbol{h}} \leftarrow \text{DEC}_{\boldsymbol{\theta}_d}(\boldsymbol{h})$
    $(\boldsymbol{\theta}_r^{(1)}, \cdots \boldsymbol{\theta}_r^{(d)}) \leftarrow (\boldsymbol{\varphi}_{\boldsymbol{\omega}_r}(\boldsymbol{\phi}(\boldsymbol{u}_1)) \cdots \boldsymbol{\varphi}_{\boldsymbol{\omega}_r}(\boldsymbol{\phi}(\boldsymbol{u}_d)))$
    $\widehat{\boldsymbol{x}} \leftarrow \text{REC}_{\boldsymbol{\theta}_r}(\widetilde{\boldsymbol{h}})$
    Define the loss $L$.
    Compute $\nabla_{\boldsymbol{\omega}_r} L$, $\nabla_\theta L$, $\nabla_{\theta_d} L$, and $\nabla_{\theta_e} L$ using backpropagation.
    Compute $\nabla_{\boldsymbol{\omega}_s^{(k)}} L$ using reparameterization trick
    Update $\boldsymbol{\omega}_r \leftarrow \boldsymbol{\omega}_r - \eta\nabla_{\boldsymbol{\omega}_r} L$,   $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\nabla_{\boldsymbol{\theta}} L$,
          $\boldsymbol{\theta}_d \leftarrow \boldsymbol{\theta}_d - \eta\nabla_{\boldsymbol{\theta}_d} L$,   $\boldsymbol{\theta}_e \leftarrow \boldsymbol{\theta}_e - \eta\nabla_{\boldsymbol{\theta}_e} L$, and
          $\boldsymbol{\omega}_r^{(k)} \leftarrow \boldsymbol{\omega}_r^{(k)} - \eta\nabla_{\boldsymbol{\omega}_r^{(k)}} L$
end for
return $\mathcal{S}$, $\boldsymbol{\Theta}$
```

Table 4: Details of Datasets used in this paper

Dataset	**Classes**	**Sample Size** (n)	**Dimensions** (d)
ALLAML	2	72	7,129
CLL_SUB	3	111	11,340
GLI_85	2	85	22,283
GLIOMA	4	50	4,434
Prostate_GE	2	102	5,966
SMK_CAN	2	187	19,993



(A) ALLAML

(B) CLL_SUB

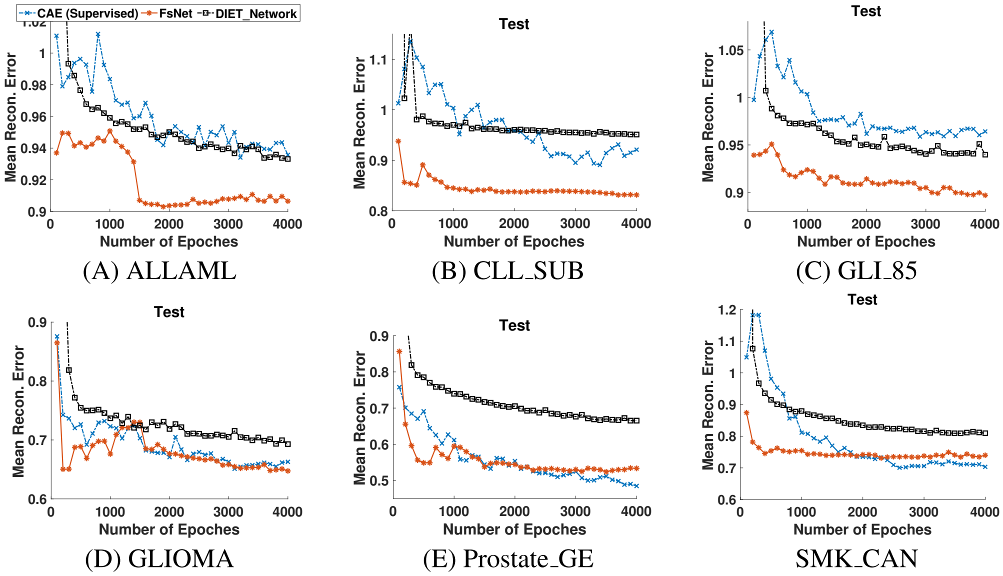
(C) GLI_85

(D) GLIOMA

(E) Prostate_GE

(F) SMK_CAN

Figure 5: Comparison among FsNet, supervised CAE, and Diet Network in terms of mean training and testing accuracies over the epochs. For the neural-network-based approaches, we set the model parameters to $b = 10$ and $K = 10$.



(A) ALLAML

(B) CLL_SUB

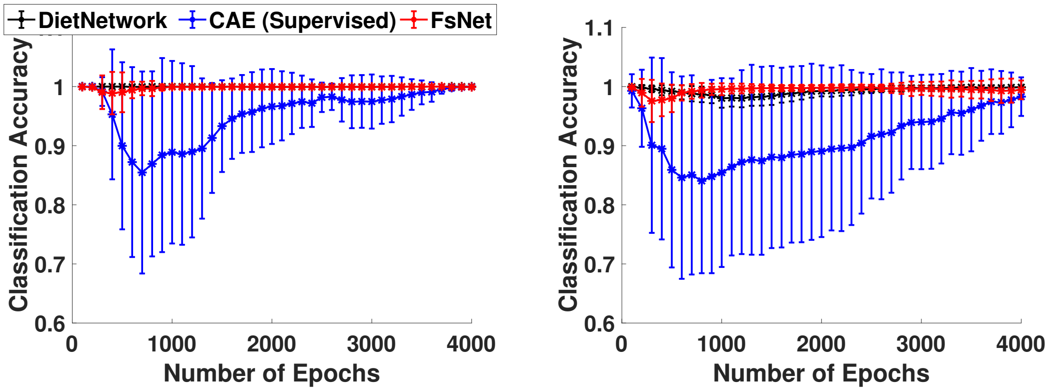
(C) GLI_85

(D) GLIOMA

(E) Prostate_GE

SMK_CAN

Figure 6: Comparison between the proposed FsNet and existing supervised CAE approaches in terms of the mean test reconstruction error over the epochs.



(A) K=10

(B) K=50

Figure 7: Comparison between the proposed FsNet and supervised CAE and Diet Network in terms of the mean test classification accuracy over the epochs on metagenome dataset. The performance of the FsNet and Diet Network is more stable than the supervised CAE.