Unveiling the Power of Sparse Neural Networks for Feature Selection

Zahra Atashgahi[a,*], **Tennison Liu**[b], **Mykola Pechenizkiy**[c], **Raymond Veldhuis**[a], **Decebal Constantin Mocanu**[d,c] **and Mihaela van der Schaar**[b]

[a]Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, The Netherlands
[b]Department of Applied Mathematics and Theoretical Physics, University of Cambridge, United Kingdom
[c]Department of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands
[d]Department of Computer Science, University of Luxembourg, Luxembourg

Abstract.

Sparse Neural Networks (SNNs) have emerged as powerful tools for efficient feature selection. Leveraging the dynamic sparse training (DST) algorithms within SNNs has demonstrated promising feature selection capabilities while drastically reducing computational overheads. Despite these advancements, several critical aspects remain insufficiently explored for feature selection. Questions persist regarding the choice of the DST algorithm for network training, the choice of metric for ranking features/neurons, and the comparative performance of these methods across diverse datasets when compared to dense networks. This paper addresses these gaps by presenting a comprehensive systematic analysis of feature selection with sparse neural networks. Moreover, we introduce a novel metric considering sparse neural network characteristics, which is designed to quantify feature importance within the context of SNNs. Our findings show that feature selection with SNNs trained with DST algorithms can achieve, on average, more than 50% memory and 55% FLOPs reduction compared to the dense networks, while outperforming them in terms of the quality of the selected features. Our code and the supplementary material are available on GitHub (`https://github.com/zahraatashgahi/Neuron-Attribution`).

1 Introduction

With the ever-increasing generation of big data, high-dimensional data has become ubiquitous in various fields such as health care, bioinformatics, and social media. Yet, high dimensionality poses substantial challenges for the analysis and interpretation of data such as, curse of dimensionality, overfitting, and high memory and computation demands [29].

Feature selection emerges as a pivotal approach to address the challenges raised by high-dimensional data. It selects the most relevant attributes of the data for the final learning task [10]. Feature selection can reduce the computational, memory, and as a result energy costs, increase interpretability, decrease data collection costs, and potentially enhance the model generalization.

Lately, neural networks have emerged as a powerful tool for feature selection. Deep learning methods gain increasing attention in various tasks due to their intrinsic attributes: automatic feature engineering, learning from data streams, facilitation of multi-source learning (multi-modal/multi-view learning), parameters pre-training, and the feasibility of an end-to-end data processing paradigm. This made them attractive for learning feature representations and AutoML [2, 43, 14, 8, 20]. One intriguing advantage of neural network feature selection compared to most traditional approaches is the ability to capture complex non-linear dependencies among features. Notably, neural network-based feature selection has exhibited substantial success, often outperforming conventional feature selection methods in identifying more relevant features for the final prediction tasks [6, 23, 28, 47, 26, 40]. Inspired by Lasso [45], a subset of neural network-based feature selection methods introduces sparsity within networks to perform feature selection. [47] introduces sparsity in the input features of a neural network via stochastic gates. [28] sparsifies the network by selecting the relevant input features for the entire network to perform global feature selection.

However, a main challenge with the existing neural network feature selection method is the high computational cost due to their large over-parameterized networks, particularly when applied to high-dimensional data [4]. This makes the deployment of such models infeasible in low-resource environments, e.g., mobile phones. In addition, in extreme cases, training and deploying over parameterized deep learning models, significantly raise energy consumption in data centers, resulting in high carbon emissions exceeding a human's annual carbon footprint [44].

Therefore, another group of works exploits the characteristics of a sparse neural network (SNN) [21] trained with dynamic sparse training (DST) [32] to find the most important attributes of a dataset. This line of works first initiated by [4] and followed by [42, 5] has demonstrated competitive feature selection capabilities, comparable to state-of-the-art algorithms, all while maintaining computational efficiency. Unlike the methods that regularize weights to achieve sparsity, the latter group of works exploits hard sparsity (zero-out weights). Moreover, they exploit dynamic sparse training to train the network sparsely from scratch to be efficient during the entire training process. Besides, they deploy sparsity in all layers and not only the initial layer; therefore, they are much more computationally efficient.

Despite showcasing superior feature selection performance coupled with increased efficiency, certain ambiguities persist concerning feature selection with sparse neural networks. *Is sparsity beneficial for all datasets when performing feature selection? What metric to choose for*

* Corresponding Author. Email: z.atashgahi@utwente.nl.
Accepted for publication at ECAI 2024

each dataset when measuring the feature's importance? How does the choice of the DST algorithm affect the feature selection performance?

In this paper, we provide an in-depth analysis of supervised feature selection with SNNs trained with dynamic sparse training. Our contributions are:

- We conduct a novel and extensive exploration and analysis of feature selection utilizing SNNs trained with dynamic sparse training. We show that SNNs trained with DST can achieve stable feature selection results regardless of the training algorithm and considered metric, even outperforming the dense neural network feature selection performance in most cases considered, while significantly reducing memory and FLOPs.
- We introduce a novel feature importance metric based on neuron attribution, highlighting its advantages over existing feature importance methods in capturing the feature relevance.

This paper aims to deepen the understanding of feature selection using sparse neural networks and provide a robust framework for evaluating various feature importance metrics and training algorithms in feature selection with sparse neural networks and DST framework. It should be noted that in this work, we focus on global feature selection rather than local feature selection (e.g. instance-based feature selection, LIME, local feature importance).

2 Backgound & Related Work

2.1 Feature Selection

Methods to perform feature selection are divided into three main categories: Filter, wrapper, and Embedded methods. **Filter Methods** [16, 19, 10] exploit variable scoring techniques to rank the features. They are model-agnostic and do not rely on the learning algorithm. As the ranking is done before classification, filter methods are prone to selecting irrelevant features. **Wrapper Methods** [49, 27, 30] aim to find a subset of the feature that achieves the highest classification performance. To tackle the NP-hard problem of evaluating numerous subsets, they employ search algorithms to find the best subset. However, due to the multiple rounds of training, these methods are costly in terms of computation. **Embedded Methods** integrate feature selection into the learning process thus being able to select relevant features while being costly-efficient. Various techniques are employed to perform embedded feature selection including, mutual information [7, 36], the SVM classifier [17], and neural networks [39].

Neural network-based feature selection has gained significant attention in recent years, both in supervised [31, 28, 47, 46, 26, 12, 37] and unsupervised [6, 18, 9, 11] settings. These methods leverage the advantages of neural networks in capturing non-linear dependencies and performing well on large datasets. However, many existing neural network-based feature selection methods suffer from over-parameterization, resulting in high computational costs, especially for high-dimensional datasets. To address these issues, a new category of methods exploits sparse neural networks to perform efficient feature selection [4, 42, 5]. Detailed discussion on SNNs for feature selection can be found in Section 2.3.2.

2.1.1 Problem Formulation

Depending on label availability, feature selection can be carried out in a supervised or unsupervised manner. This study focuses on addressing the supervised feature selection challenge. Given a dataset $\mathbb{X}$ comprising m samples denoted as $(\boldsymbol{x}^{(i)}, y^{(i)})$, where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$ represents the i-th sample in the data matrix $\boldsymbol{X} \in \mathbb{R}^{m \times d}$ (with d being the dataset's dimensionality or the number of features) and $y^{(i)}$ is the corresponding label for supervised learning, the objective is to choose a subset of the most discriminative and informative features from $\boldsymbol{X}$, denoted as $\mathbb{F}_s \subset \mathbb{F}$. Here, $|\mathbb{F}_s| = K$, with $\mathbb{F}$ being the original feature set, and K is a hyperparameter indicating the number of selected features. In supervised feature selection, we seek to optimize:

$$\mathbb{F}_s^* = \underset{\mathbb{F}_s \subset \mathbb{F}, |\mathbb{F}_s| = K}{\operatorname{argmin}} \sum_{i=0}^{m-1} J(f(\boldsymbol{x}_{\mathbb{F}_s}^{(i)}; \boldsymbol{\theta}), y^{(i)}), \tag{1}$$

where $\mathbb{F}_s^*$ is the final selected feature set, J is a desired loss function, and $f(\boldsymbol{x}_{\mathbb{F}_s}^{(i)}; \boldsymbol{\theta})$ is a classification function parameterized by $\boldsymbol{\theta}$ estimating the target for the i-th sample using a feature subset $\boldsymbol{x}_{\mathbb{F}_s}^{(i)}$. However, directly optimizing Equation 1 is an NP-hard problem [28]. Therefore, by estimating the importance of the features using the neuron attribution method, we try to find the closest set of features to the optimal feature set found by Equation 1.

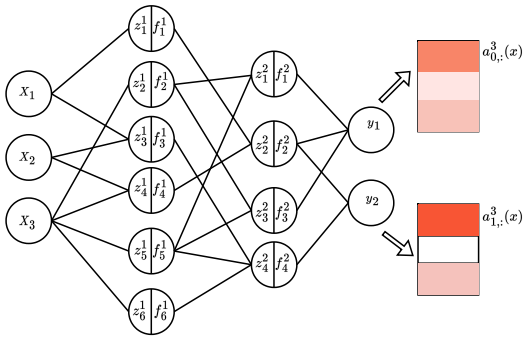


Figure 1: Neuron Attribution visualization in a sparse neural network. The contribution of each input feature for any output neuron is measured by neuron attribution methods. Darker colors show a higher contribution of the corresponding input neuron to the output neuron.

2.2 Neuron Attribution

In this work, we propose a new feature selection metric from neural networks that is based on neuron attribution. Attribution methods aim to explain the predictions of a neural network. These methods have also been used to explain the predictions of a model[1]. Many works have regularized network attribution to achieve the desired output and improve the learning of neural networks [38, 15, 34, 25]. Given a neural network, an attribution method aims to determine the contribution/relevancy of each input feature to the output of $i-th$ output neuron (f_i^{L-1}) [1]. Usually, the gradient information is used to measure the neuron attribution. Neuron attribution for the i^{th} neuron in the output layer w.r.t. the feature x_j can be simply defined as:

$$a_{ij}^{L-1}(x) = \frac{\partial f_i^{L-1}(x)}{\partial x_j}, \tag{2}$$

where L is the number of network layers and $L-1$ is the index of the output layer. The higher the absolute value of $a_{ij}^{L-1}(x)$ is, the more sensitive output neuron i is to the changes in the input feature x_j for observation x. Other gradient attribution methods can be used to measure the neuron attribution in Equation 2 [1]. An example of neuron attribution attribution is visualized in Figure 1.

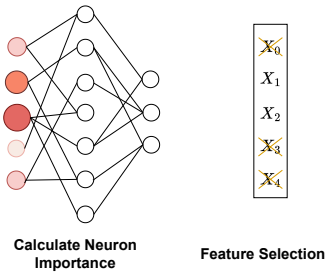


Figure 2: Feature selection using sparse neural network. The importance of each input neuron can be measured using the network's characteristics. Darker colors and larger neurons show a higher importance of the corresponding input neuron.

2.3 *Sparse Neural Networks*

Sparse neural networks are an approach to address the overparameterization of deep neural networks. By introducing sparsity in the connectivity and/or the units of a dense neural network, they aim to match the performance of dense neural networks while reducing computational and memory costs [21, 33]. Due to the reduction of learned noise, they can even improve the generalization of the network [21]. A sparse neural network is represented by $f(\boldsymbol{x}, \boldsymbol{\theta}_s)$ that has a sparsity level of P, where P is calculated as $1 - \frac{\|\boldsymbol{\theta}_s\|_0}{\|\boldsymbol{\theta}\|_0}$. Here, $\boldsymbol{\theta}_s$ denotes a subset of parameters of the dense network parameterized by $\boldsymbol{\theta}$, and $\|\boldsymbol{\theta}_s\|_0$ and $\|\boldsymbol{\theta}\|_0$ refer to the number of parameters of the sparse and dense network respectively.

2.3.1 *Dynamic Sparse Training (DST)*

DST comprises a range of techniques to train sparse neural networks from scratch. The goal of DST methods is to optimize the sparse connectivity of the network during training, without resorting to dense network matrices at any point [33, 32, 48]. To achieve this, DST methods begin with initializing a random sparse neural network. During training, DST methods periodically update the sparse connectivity of the network by removing a fraction ζ of the parameters $\boldsymbol{\theta}_s$ and adding the same number of parameters to the network to keep the sparsity level fixed. In the literature, usually, weight magnitude has been used as a criterion for dropping the connections. However, there exist various approaches for weight regrowth including, random [32], gradient [13, 24], and neuron similarity [3].

2.3.2 *DST for Feature Selection*

QuickSelection was the first work to show that sparse neural networks trained with DST can perform feature selection. It proposed the *neuron strength* [4] in a sparse denoising autoencoder to determine the importance of each input neuron in the input layer of a sparse neural network (and the corresponding input feature in the original dataset)

Table 1: Comparison with related works that exploit sparsity for feature selection.

Method	Sparsity	Scalability	Neuron Attribution Importance
Lasso [45]	Regularization	✓	✗
STG [47]	Regularization	✗	✗
LassoNet [28]	Regularization	✗	✗
QuickSelection [4]	DST	✓	✗
SET-Attr (ours)	DST	✓	✓

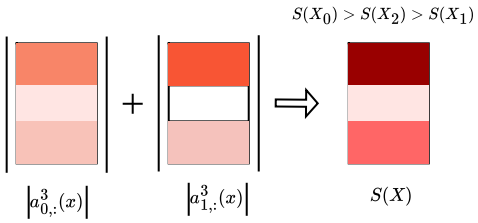


Figure 3: A toy example of neuron attribution-based importance calculation. Darker colors indicate higher contributions of the corresponding input neuron.

in the unsupervised learning settings. Neuron strength is computed as:

$$S(X_j) = \sum_{i=0}^{n^{h^1}-1} |W^1_{ji}|, \tag{3}$$

where n^{h^1} is the number of hidden neurons in the first hidden layer, and W^l is the weight matrix of the lth layer.

Later on, [42], proposed to use a combination of neuron strength and gradient of loss to the output neurons in an autoencoder to measure the importance of neurons. [5] proposed to perform input neuron updating in a sparse MLP trained with DST to gradually reduce the number of input neurons and then use neuron strength to rank the features.

In this paper, we study the efficacy of SNNs for feature selection and propose a new importance metric based on neuron attribution to measure the importance of input features in neural networks. Table 1, compares our proposed method with the closest related work that exploit sparsity to perform feature selection. Figure 2 shows a toy example of feature selection using sparse neural networks.

3 Methodology

The contributions of this research are twofold. Firstly, we provide an extensive analysis of the performance of SNNs trained with DST for feature selection. Secondly, we propose a new metric to derive the importance of features in an SNN trained with DST. We elaborated on the experimental analysis settings and findings in Section 4. In this section, we describe our proposed approach to measure neuron/feature importance based on neuron attribution.

Motivation. As elucidated in Section 2.2, neuron attribution serves as a metric to estimate the relevance of an input feature to a specific output neuron, given an input sample. Consequently, the neuron attribution vector for a hidden neuron provides insights into which set of features is more relevant to derive the output. Looking at the contribution vectors of all output neurons, we can select the most relevant input features for each output feature. Thus, neuron attribution enables us to rank input features based on their relevance to the output features. Features with the highest contributions in the output neurons offer a robust estimate of the output, implicitly guiding us toward identifying the optimal feature set, as represented in Equation 1. Building on this premise, we introduce the neuron attribution feature selection metric in the context of neural networks.

3.1 *Input Neuron Importance*

The importance of the input neurons is computed based on the neuron attribution of the output neurons. We propose to compute the importance score of each input neuron as the following:

Table 2: Datasets characteristics.

Type	Dataset	#Samples	#Features	#Classes
Image (Hand-written)	MNIST	70000	784	10
	USPS	9298	256	10
	Gisette	7000	5000	2
Image	COIL20	1440	1024	20
	ORL	400	1024	40
	Yale	165	1024	15
Text	BASEHOCK	1993	4862	2
	PCMAC	1943	3289	2
	RELATHE	1427	4322	2
Biological	Prostate-GE	102	5966	2
	SMK-CAN-187	187	19993	2
	CLL-SUB-111	111	11340	3
	lymphoma	96	4026	9
	Carcinom	174	9182	11
Time Series	HAR	10299	561	6
Mass Spectrometry	Arcene	200	10000	2
Speech	Isolet	1560	617	26
Artificial (Noisy)	Madelon	2600	500	2

$$S_{itr}(X_j) = \sum_{i=0}^{C-1}[\frac{1}{m}\sum_{k=0}^{m-1}|a_{ij}^{L-1}(x_k)|], \tag{4}$$

where $S_{itr}(X_j)$ is the importance of the jth feature at iteration itr, C is the number of output neurons (classes), m is the number of samples, and $a_{ij}^{L-1}(x_k)$ is the neuron attribution of the jth input neuron for the ith output neuron and sample x_k. To compute neuron importance in Equation 4 for each input neuron, we sum the neuron attribution for all output neurons (averaged over all samples); this choice is driven by empirical results. In other words, a feature is important if it is highly relevant for most output features and input samples. For computing the overall importance of the neurons during training, we sum $S_{itr}(X_j)$ over all the training iterations. While attribution methods have been studied only in dense neural networks, in this paper, we study these methods for sparse neural networks. A toy example of this process is visualized in Figure 3.

Feature Selection. After deriving the input neurons importance based in Equation 4, we select the features corresponding to the neurons with the highest neuron attribution-based importance as the final feature set.

4 Results

In this section, we first describe the experimental settings (4.1). Then, we present the results of the feature selection comparison among sparse and dense models in Section 4.2, and among standard feature selection baselines using sparsity in Section 4.4. Finally, we visualize the neuron importance in Section 4.3. Additionally, we perform an analysis on a synthetic dataset in Appendix C.

4.1 Experimental Setup

In the following, we describe datasets, baselines, and the evaluation approach. More experimental details, including the hyperparameter settings and model architecture, can be found in Appendix A.

4.1.1 Datasets

The datasets, outlined in Table 2, include a diverse collection of 18 datasets, varying in size and type. This selection allows for a comprehensive analysis of each method across different domains. More than half of these datasets are high-dimensional making them challenging benchmarks for the models.

4.1.2 Baselines

DST for Feature Selection. The main focus of this paper is to analyze how sparse neural networks trained with DST perform in feature selection compared to dense networks. Therefore, we consider dense and sparse baselines. Secondly, for the sparse models, we want to study how the DST algorithm for training the sparse neural network affects the feature selection performance. We consider two standard DST approaches in the literature, SET [32] and RigL [13], that are frequently used in evaluating the DST framework. We describe SET and RigL in Appendix A.2. Finally, we want to assess the efficacy of the neuron importance metric. We consider the neuron strength metric from QuickSelection [4], which is also used as a part of neuron importance estimation in [42] or directly used in [5] to rank the features in a supervised setting; we call this metric as "QS" in the experiments. We compare the neuron strength metric with our proposed neuron attribution metric, called "Attr". These metrics combined with the three considered models, dense network (*Dense*), SNN trained with SET (*SET*), and SNN trained with RigL (*RigL*), result in 6 baselines for the experiments: *Dense-QS*, *Dense-Attr*, *SET-QS*, *SET-Attr*, *RigL-QS*, *RigL-Attr*. For QuickSelection, as we also do for Attr, to compute the input neuron/feature importance, we sum the importance during all training epochs. As shown in Appendix E, we observed that looking at the history of importance during all training epochs improves the results for these models.

Sparsity for Feature Selection. We consider three feature selection methods that exploit sparsity to perform feature selection including STG [47], LassoNet [28], and Lasso [45].

4.1.3 Evaluation

For evaluating each method, we first perform feature selection to derive K most important features (K is a hyperparameter set by the user). Then, we train an SVM classifier on the subset of K features of the original data and report the test accuracy on a hold-out test set.

4.2 Feature Selection Comparison

Settings. In this experiment, we compare the two feature ranking criteria, neuron strength from QuickSelection (QS), and neuron attribution (Attr). We consider dense and sparse MLPs to perform feature selection on. The sparse models are trained with DST; we consider SET and RigL as the training algorithms. The comparison results are summarized in Tables 3. For ease of comparison, we considered pairwise comparison. However, comparisons of all baselines referred to in Section 4.1.2, are brought in Appendix D. Additionally, we analyze the performance of each method across various K values in $\{25, 50, 75, 100, 200\}$ in terms of average ranking in Appendix B.

Dense vs Sparse. Comparing dense and sparse models, we consider Dense-Attr and SET-Attr. In Table 3a, we can observe that while on high-dimensional biological datasets and the noisy dataset (Madelon) Dense-Attr performs better, on the rest of the datasets (11 out of 18 datasets) SET-Attr excels the dense model. Additionally, SET-Attr exploits significantly fewer parameters (memory) and FLOPs (computational costs). As shown in Table 3, on average over all datasets, SET achieves 54.2% sparsity and RigL achieves 66.4% sparsity. In Table 5, we present the FLOPs (Floating-Point Operations) count for

Table 3: Feature selection results in terms of test classification accuracy [%] of an SVM classifier on the selected subset of (K=100). The values in the parenthesis show the sparsity level.

	Dense-Attr	**SET-Attr**
MNIST	96.23 ± 0.08 (0.00)	**96.24** ± **0.13** (0.25)
USPS	**96.83** ± **0.13** (0.00)	96.77 ± 0.13 (0.50)
Gisette	97.03 ± 0.26 (0.00)	**97.10** ± **0.23** (0.50)
Coil20	**98.82** ± **1.17** (0.00)	98.02 ± 1.85 (0.50)
ORL	89.38 ± 3.80 (0.00)	**89.50** ± **2.45** (0.25)
Yale	63.94 ± 7.48 (0.00)	**65.76** ± **7.05** (0.25)
BASEHOCK	92.83 ± 1.51 (0.00)	**94.34** ± **0.66** (0.95)
PCMAC	84.96 ± 1.40 (0.00)	**87.89** ± **1.26** (0.95)
RELATHE	80.24 ± 2.38 (0.00)	**82.10** ± **1.02** (0.95)
Prostate_GE	**89.05** ± **2.18** (0.00)	88.57 ± 2.33 (0.50)
SMK	**81.05** ± **3.07** (0.00)	80.53 ± 2.41 (0.50)
CLL	**83.48** ± **5.43** (0.00)	78.26 ± 8.02 (0.25)
Carcinom	**84.57** ± **4.64** (0.00)	83.43 ± 5.83 (0.50)
Lymphoma	64.50 ± 5.68 (0.00)	**66.00** ± **7.35** (0.50)
Arcene	65.25 ± 7.62 (0.00)	**67.75** ± **7.11** (0.25)
HAR	94.78 ± 0.37 (0.00)	**94.90** ± **0.39** (0.25)
Isolet	94.33 ± 1.31 (0.00)	**95.53** ± **0.42** (0.95)
Madelon	**81.52** ± **2.55** (0.00)	76.63 ± 2.36 (0.95)

(a) Dense-Attr vs. SET-Attr.

	RigL-Attr	**SET-Attr**
MNIST	96.22 ± 0.10 (0.50)	**96.24** ± **0.13** (0.25)
USPS	**96.87** ± **0.17** (0.80)	96.77 ± 0.13 (0.50)
Gisette	97.07 ± 0.27 (0.50)	**97.10** ± **0.23** (0.50)
Coil20	**98.06** ± **1.87** (0.80)	98.02 ± 1.85 (0.50)
ORL	**90.88** ± **3.26** (0.25)	89.50 ± 2.45 (0.25)
Yale	**73.94** ± **5.94** (0.50)	65.76 ± 7.05 (0.25)
BASEHOCK	86.69 ± 2.20 (0.95)	**94.34** ± **0.66** (0.95)
PCMAC	81.70 ± 2.29 (0.95)	**87.89** ± **1.26** (0.95)
RELATHE	74.69 ± 2.96 (0.98)	**82.10** ± **1.02** (0.95)
Prostate_GE	88.09 ± 3.19 (0.80)	**88.57** ± **2.33** (0.50)
SMK	80.26 ± 3.95 (0.50)	**80.53** ± **2.41** (0.50)
CLL	**78.70** ± **7.64** (0.25)	78.26 ± 8.02 (0.25)
Carcinom	80.00 ± 6.26 (0.50)	**83.43** ± **5.83** (0.50)
Lymphoma	65.50 ± 7.57 (0.50)	**66.00** ± **7.35** (0.50)
Arcene	57.00 ± 6.40 (0.80)	**67.75** ± **7.11** (0.25)
HAR	94.88 ± 0.40 (0.50)	**94.90** ± **0.39** (0.25)
Isolet	94.72 ± 0.58 (0.90)	**95.53** ± **0.42** (0.95)
Madelon	75.75 ± 3.50 (0.98)	**76.63** ± **2.36** (0.95)

(b) RigL-Attr vs. SET-Attr.

	SET-QS	**SET-Attr**
MNIST	96.02 ± 0.10 (0.25)	**96.24** ± **0.13** (0.25)
USPS	96.64 ± 0.13 (0.50)	**96.77** ± **0.13** (0.50)
Gisette	96.65 ± 0.45 (0.50)	**97.10** ± **0.23** (0.50)
Coil20	**98.72** ± **0.31** (0.50)	98.02 ± 1.85 (0.50)
ORL	**92.25** ± **2.15** (0.25)	89.50 ± 2.45 (0.25)
Yale	**74.55** ± **3.64** (0.25)	65.76 ± 7.05 (0.25)
BASEHOCK	93.78 ± 1.01 (0.95)	**94.34** ± **0.66** (0.95)
PCMAC	86.76 ± 1.41 (0.95)	**87.89** ± **1.26** (0.95)
RELATHE	80.77 ± 1.35 (0.95)	**82.10** ± **1.02** (0.95)
Prostate_GE	87.62 ± 4.86 (0.50)	**88.57** ± **2.33** (0.50)
SMK	79.47 ± 1.97 (0.50)	**80.53** ± **2.41** (0.50)
CLL	76.09 ± 8.53 (0.25)	**78.26** ± **8.02** (0.25)
Carcinom	**89.14** ± **2.49** (0.50)	83.43 ± 5.83 (0.50)
Lymphoma	**70.50** ± **6.10** (0.50)	66.00 ± 7.35 (0.50)
Arcene	66.25 ± 8.31 (0.25)	**67.75** ± **7.11** (0.25)
HAR	94.59 ± 0.41 (0.25)	**94.90** ± **0.39** (0.25)
Isolet	95.22 ± 0.35 (0.95)	**95.53** ± **0.42** (0.95)
Madelon	**78.40** ± **1.71** (0.95)	76.63 ± 2.36 (0.95)

(c) SET-QS vs. SET-Attr.

SET-Attr and Dense-Attr; SET significantly reduces the number of FLOPs, and as a result, computational costs, in all cases compared to the dense model.

DST Algorithm. Table 3b compares the results of RigL-Attr and SET-Attr. While on the Image datasets (Coil20, ORL, and Yale), RigL outperforms SET in the quality of the selected features on the rest of the datasets SET outperforms RigL or has comparable performance. On average, RigL achieves 66.4% sparsity, while SET achieves 54.2% sparsity.

Neuron Attribution vs Neuron Strength. In Table 3c, we compare the two neuron strength (QS) and neuron attribution (Attr) metrics, when the network is trained using the SET algorithm. Across the hand-written datasets (MNIST, USPS, Gisette), text datasets (BASEHOCK, PCMAC, RELATHE), and HAR, SET-Attr consistently yield better results than SET-QS on each of the models. This pattern is similar when the network is trained as Dense or with RigL (please see Table 7 in Appendix D). In image datasets (Coil20, ORL, Yale) and Madelon, SET-QS outperforms SET-Attr. The Biological datasets (Prostate_GE, SMK, CLL, Carcinom, Lymphoma) showcase varying degrees of performance for different methods. SET-Attr performs better in some cases, such as Prostate_GE, SMK, and CLL, while in others, such as Carcinom and Lymphoma, SET-QS shows superiority. This indicates the complexity and diversity of biological data, suggesting that the choice of feature selection method might depend heavily on the specific dataset characteristics.

Conclusions. The results show that neuron attribution generally outperforms neuron strength, sparse models outperform dense ones, and SET is generally a better choice when training the sparse neural networks for feature selection than RigL. However, selecting an appropriate feature selection metric and training algorithm based on dataset characteristics and domain requirements is of great importance.

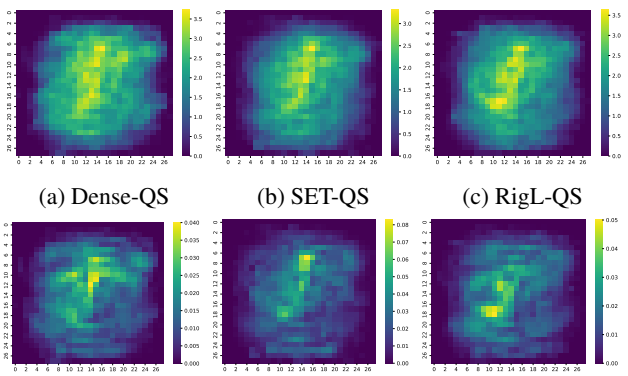
4.3 Neuron Importance Visualization

Settings. In this section, we visualize the neuron importance on the MNIST dataset for each method in Table 3. We plot the neuron importance of input features as a 2D heatmap in Figure 4.

Results. As we can observe in the first row of Figure 4, the neuron strength metric in QuickSelection is mostly similar for all three models, showing that they all can detect the most important features which mostly appear in the center of the image as we see in the MNIST dataset. The pattern formed by the neuron attribution metric in the second row of Figure 4, is close to that of neuron strength, where important features appear in the center of the image. However, the main difference is that neuron attribution reaches a more sparse feature importance pattern. This shows that neuron attribution focuses mostly on the most important features.

4.4 Comparsion with Baselines

Settings. We compare SET-Attr (achieving the highest ranking among the considered methods when $K = 100$) with three popular feature selection methods in the literature that exploit sparsity to perform feature selection: Lasso, STG [47], and LassoNet [28]. We compare



(a) Dense-QS (b) SET-QS (c) RigL-QS

(d) Dense-Attr (e) SET-Attr (f) RigL-Attr

Figure 4: Neuron importance visualization on the MNIST dataset as 2d-heat maps. The lighter area in the center of the Figures shows more important features which is in-line with the pattern of digits in the MNIST dataset.

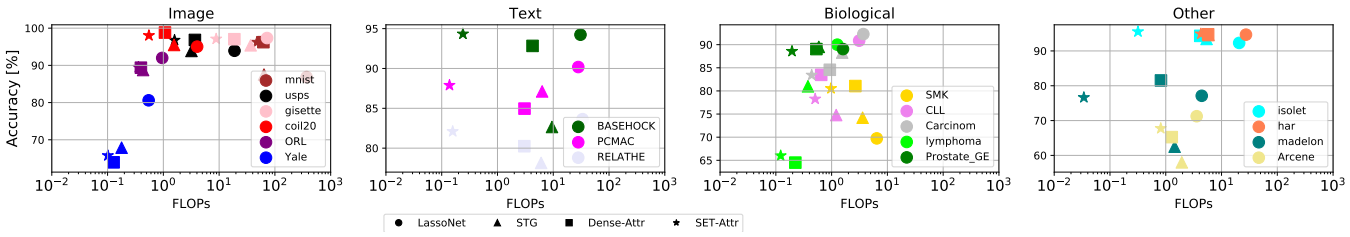


Figure 5: Accuracy vs. FLOPs comparison among various neural network-based feature selection methods inducing sparsity in the network. The FLOPs values are divided by 10^{12}

the feature selection performance of these methods when selecting $K = 100$ features.

Feature Selection Performance. The feature selection results are summarized in Table 4. SET-Attr and LassoNet are the best performers among the considered methods, each being 7 and 10 times the best performers, respectively. STG and Lasso consistently fall behind both SET-Attr and LassoNet in most datasets. By looking at the theoretical estimation of FLOPs count in Table 5, we can observe that SET-Attr has the lowest number of FLOPs in all datasets, having on average 88.2% less flops than LassoNet. This indicates the efficiency of SET-Attr while achieving a competitive feature selection performance to the SOTA feature selection algorithms.

Computational Costs Comparison. To compare the computational efficiency of these methods, we have computed the theoretical estimation of FLOPs count in Table 5.

Examining the efficiency gains achieved by a sparse neural network in comparison to its dense counterpart commonly involves assessing the FLOPs (floating-point operations), as highlighted in prior works [13, 41]. The evaluation of *FLOPs* serves to estimate the time complexity of an algorithm irrespective of its specific implementation. Given that current deep learning hardware is not tailored for sparse matrix computations, many methods for generating sparse neural networks resort to simulating sparsity through binary weight masks. Consequently, the execution time of such approaches may not accurately reflect their efficiency. Furthermore, the community is actively engaged in exploring dedicated sparse implementations for these networks [22]. In line with the theoretical nature of our paper, we defer consideration of these engineering aspects to future research. Therefore, our analysis of efficiency also incorporates FLOPs count.

We only show these numbers for neural network-based feature selection methods, as the computational efficiency of Lasso is very low, and it falls behind the performance of the competitors on most datasets. LassoNet sparsifies the input layer by dropping features during training; however, other layers of the network are dense. Moreover, due to the long training time and convergence, it needs much higher FLOPs than other methods. For example, it requires 88.2% more FLOPs than SET-Attr. STG exploits a probabilistic gating mechanism in the input layer to select features; if a gate is converged to 0, it is removed from the network; if it is converged to 1, it is selected. when the gate probabilities are not converged to 0/1, a cutoff can be set on the gate/feature probabilities to select features. We observed in our experiments that while some gate probabilities might converge to small values, they do not converge to exact 0. As a result, all input neurons contribute to the network's output. Therefore, the model is dense during training and the FLOPs are much higher than SET-Attr in all cases considered.

Table 4: Feature selection results in terms of classification accuracy of an SVM classifier on the selected subset of features (K=100). **Bold** entries denote the best performers.

Dataset	Lasso	LassoNet	STG	SET-Attr
MNIST	78.27 ± 0.00	86.83 ± 0.11	87.54 ± 0.55	**96.24 ± 0.13**
USPS	93.55 ± 0.00	93.92 ± 0.16	93.88 ± 0.19	**96.77 ± 0.13**
Gisette	93.70 ± 0.00	**97.34 ± 0.21**	95.39 ± 0.60	97.10 ± 0.23
Coil20	94.10 ± 0.00	95.03 ± 0.79	95.49 ± 1.10	**98.02 ± 1.85**
ORL	81.25 ± 0.00	**92.00 ± 2.92**	88.75 ± 3.06	89.50 ± 2.45
Yale	72.73 ± 0.00	**80.61 ± 2.78**	67.88 ± 5.45	65.76 ± 7.05
BASEHOCK	87.72 ± 0.00	94.24 ± 0.59	82.66 ± 9.23	**94.34 ± 0.66**
PCMAC	66.07 ± 0.00	**90.18 ± 0.99**	87.12 ± 3.74	87.89 ± 1.26
RELATHE	80.77 ± 0.00	**83.64 ± 2.37**	78.15 ± 5.32	82.10 ± 1.02
Prostate_GE	**90.48 ± 0.00**	89.05 ± 2.18	89.52 ± 3.56	88.57 ± 2.33
SMK	73.68 ± 0.00	69.74 ± 4.44	74.21 ± 8.39	**80.53 ± 2.41**
CLL	69.56 ± 0.00	**90.87 ± 5.65**	74.78 ± 8.65	78.26 ± 8.02
Carcinom	85.71 ± 0.00	**92.29 ± 1.83**	88.29 ± 5.18	83.43 ± 5.83
Lymphoma	65.00 ± 0.00	**90.00 ± 2.67**	81.00 ± 6.63	66.00 ± 7.35
Arcene	65.00 ± 0.00	**71.25 ± 3.58**	58.00 ± 7.14	67.75 ± 7.11
HAR	92.87 ± 0.00	94.68 ± 0.20	94.56 ± 0.39	**94.90 ± 0.39**
Isolet	91.47 ± 0.00	92.30 ± 0.44	93.44 ± 0.26	**95.53 ± 0.42**
Madelon	58.67 ± 0.00	**77.12 ± 1.31**	62.50 ± 6.34	76.63 ± 2.36

5 Analysis of Effects of Sparsity for Feature Selection

Accuracy vs FLOPs Tradeoff. To summarize the results of Tables 4 and 5, the tradeoff between accuracy and FLOPs is shown in Figure 5. As we can observe, SET-Attr has in most cases in order of magnitude lower number of FLOPs than LassoNet while achieving competitive performance. STG and Dense-Attr fall behind in terms of both accuracy and FLOPs count in most cases considered.

In Section 4.2, we optimize the sparsity level, selecting the value yielding minimal loss on the validation set. However, our focus now shifts to a comprehensive analysis of the impact of sparsity on feature selection performance, exploring a range of sparsity levels (i.e., $0.25, 0.50, 0.75, 0.90, 0.95, 0.98$) for each dataset. The results are visualized in Figure 6.

Studying image datasets (Coil-20, ORL, Yale), we observe that sparsity levels up to 80% generally enhance or maintain performance, with notable improvement on the Yale dataset. Beyond 80%, the feature selection method with SET (using any of the two importance metrics), remains stable or even exhibits improvement, whereas RigL experiences degradation in the high sparsity regime.

For hand-written digits datasets (MNIST, USPS, Gisette), sparsity consistently leads to performance improvement. Notably, the neuron attribution metric frequently outperforms neuron strength. At very high sparsity (98%), feature selection with SET demonstrates significant performance gains, while RigL's performance either

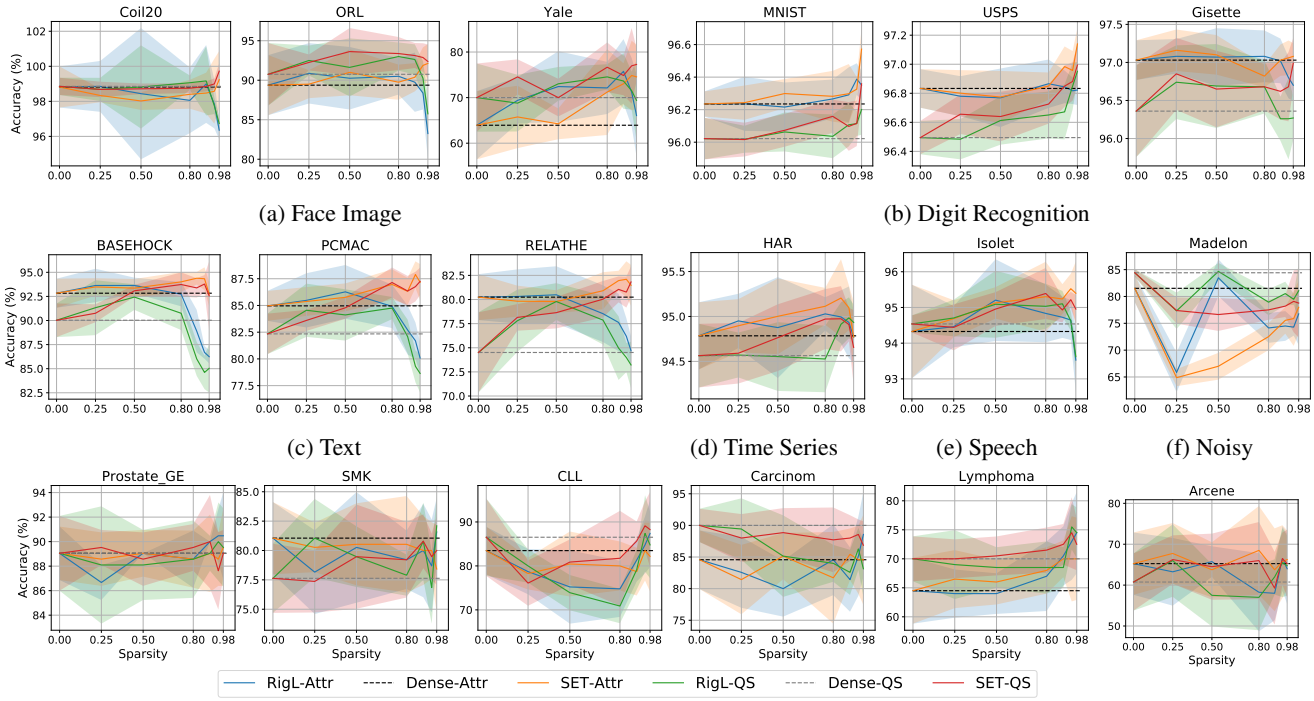


Figure 6: Sparsity effect of feature selection performance.

slightly improves or even declines.

On text datasets (BASEHOCK, PCMAC, REALTHE), performance improvement occurs until 50% sparsity for all methods. However, RigL's feature selection performance, using both importance metrics, start to decline beyond 50% sparsity, while SET continues to improve. Neuron attribution consistently outperforms neuron strength across most cases.

Across other high-dimensional datasets, the observed patterns vary. SET-QS often benefits from high sparsity ($> 90\%$), outperforming dense networks in many cases. The Lymphoma dataset achieves maximum performance at 95% sparsity for all methods. SMK, Prostate_GE, Carcinom, and Arcene can be highly sparsified without significant performance drop, providing computational efficiency.

On HAR and Isolet datasets, sparsity enhances performance up to 90% sparsity level. However, on the highly noisy Madelon dataset, sparsity generally deteriorates performance, with RigL-QS demonstrating superior performance among sparse models. Notably, RigL outperforms SET overall on the Madelon dataset.

This study emphasizes the importance of considering dataset characteristics when selecting an appropriate sparsity level for feature selection.

Table 5: FLOPs comparison among neural network feature selection methods (All numbers are divided by 10^{12}). **Bold** entries denote the best performers.

Dataset	LassoNet	STG	Dense-Attr	SET-Attr
MNIST	370.10	63.76	62.14	**47.67**
USPS	18.87	3.19	3.68	**1.59**
Gisette	72.29	36.73	18.75	**8.85**
Coil20	4.04	1.56	1.07	**0.55**
ORL	0.95	0.43	0.40	**0.36**
Yale	0.55	0.18	0.13	**0.10**
BASEHOCK	30.76	9.49	4.25	**0.24**
PCMAC	28.09	6.32	3.05	**0.14**
RELATHE	33.53	6.06	3.02	**0.16**
Prostate_GE	1.60	0.59	0.53	**0.19**
SMK	6.45	3.59	2.66	**0.97**
CLL	3.12	1.21	0.65	**0.51**
Carcinom	3.70	1.55	0.93	**0.44**
Lymphoma	1.26	0.38	0.22	**0.12**
Arcene	3.58	1.94	1.28	**0.81**
HAR	27.44	5.84	5.76	**4.27**
Isolet	20.72	5.39	4.15	**0.32**
Madelon	4.42	1.44	0.82	**0.03**

6 Conclusions

In conclusion, our work contributes significantly to the understanding of feature selection with sparse neural networks within the dynamic sparse training framework. Through systematic analysis, we demonstrate the efficacy of sparse neural networks in feature selection from diverse datasets, comparing them against dense neural networks and other sparsity-inducing methods. Our findings reveal that SNNs trained with DST algorithms achieve remarkable memory and computational savings, exceeding 50% and 55% respectively compared to dense networks, while maintaining superior feature selection quality in 13 out of 18 cases. One promising direction to continue this research is to consider neuron importance metrics to improve the training of sparse neural networks in the DST framework to guide the weight addition process.

References

[1] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Sy21R9JAW.

[2] S. Ö. Arik and T. Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6679–6687, 2021.

[3] Z. Atashgahi, J. Pieterse, S. Liu, D. C. Mocanu, R. Veldhuis, and M. Pechenizkiy. A brain-inspired algorithm for training highly sparse neural networks. *Machine Learning*, 111(12):4411–4452, 2022.

[4] Z. Atashgahi, G. Sokar, T. van der Lee, E. Mocanu, D. C. Mocanu, R. Veldhuis, and M. Pechenizkiy. Quick and robust feature selection: the strength of energy-efficient sparse training for autoencoders. *Machine Learning*, pages 1–38, 2022.

[5] Z. Atashgahi, X. Zhang, N. Kichler, S. Liu, L. Yin, M. Pechenizkiy, R. Veldhuis, and D. C. Mocanu. Supervised feature selection with neuron evolution in sparse neural networks. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=GcO6ugrLKp.

[6] M. F. Balın, A. Abid, and J. Zou. Concrete autoencoders: Differentiable feature selection and reconstruction. In *International conference on machine learning*, pages 444–453. PMLR, 2019.

[7] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on neural networks*, 5(4):537–550, 1994.

[8] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[9] B. Chandra and R. K. Sharma. Exploring autoencoders for unsupervised feature selection. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2015.

[10] G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[11] G. Doquet and M. Sebag. Agnostic feature selection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 343–358. Springer, 2019.

[12] R. Espinosa, F. Jiménez, and J. Palma. Embedded feature selection in lstm networks with multi-objective evolutionary ensemble learning for time series forecasting. *arXiv preprint arXiv:2312.17517*, 2023.

[13] U. Evci, T. Gale, J. Menick, P. S. Castro, and E. Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR, 2020.

[14] Z. Gharibshah and X. Zhu. Local contrastive feature learning for tabular data. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 3963–3967, 2022.

[15] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[16] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[17] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1):389–422, 2002.

[18] K. Han, Y. Wang, C. Zhang, C. Li, and C. Xu. Autoencoder inspired unsupervised feature selection. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2941–2945. IEEE, 2018.

[19] X. He, D. Cai, and P. Niyogi. Laplacian score for feature selection. In *Advances in neural information processing systems*, pages 507–514, 2006.

[20] X. He, K. Zhao, and X. Chu. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

[21] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *The Journal of Machine Learning Research*, 22(1): 10882–11005, 2021.

[22] S. Hooker. The hardware lottery. *Communications of the ACM*, 64(12): 58–65, 2021.

[23] F. Imrie, A. Norcliffe, P. Liò, and M. van der Schaar. Composite feature selection using deep ensembles. *Advances in Neural Information Processing Systems*, 35:36142–36160, 2022.

[24] S. Jayakumar, R. Pascanu, J. Rae, S. Osindero, and E. Elsen. Top-kast: Top-k always sparse training. *Advances in Neural Information Processing Systems*, 33:20744–20754, 2020.

[25] A. Jeffares, T. Liu, J. Crabbé, F. Imrie, and M. van der Schaar. TANGOS: Regularizing tabular neural networks through gradient orthogonalization and specialization. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=n6H86gW8u0d.

[26] K. Jia and M. Rinard. Effective neural network l_0 regularization with binmask. *arXiv preprint arXiv:2304.11237*, 2023.

[27] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.

[28] I. Lemhadri, F. Ruan, L. Abraham, and R. Tibshirani. Lassonet: A neural network with feature sparsity. *The Journal of Machine Learning Research*, 22(1):5633–5661, 2021.

[29] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu. Feature selection: A data perspective. *ACM computing surveys (CSUR)*, 50(6):1–45, 2017.

[30] H. Liu, R. Setiono, et al. A probabilistic approach to feature selection-a filter solution. In *ICML*, volume 96, pages 319–327. Citeseer, 1996.

[31] Y. Lu, Y. Fan, J. Lv, and W. S. Noble. Deeppink: reproducible feature selection in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 8676–8686, 2018.

[32] D. C. Mocanu, E. Mocanu, P. Stone, P. H. Nguyen, M. Gibescu, and A. Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018.

[33] D. C. Mocanu, E. Mocanu, T. Pinto, S. Curci, P. H. Nguyen, M. Gibescu, D. Ernst, and Z. A. Vale. Sparse training theory for scalable and efficient agents. *arXiv preprint arXiv:2103.01636*, 2021.

[34] S.-M. Moosavi-Dezfooli, A. Fawzi, J. Uesato, and P. Frossard. Robustness via curvature regularization, and vice versa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9078–9086, 2019.

[35] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[36] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8): 1226–1238, 2005.

[37] H. Peng, G. Fang, and P. Li. Copula for instance-wise feature selection and rank. In *Uncertainty in Artificial Intelligence*, pages 1651–1661. PMLR, 2023.

[38] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on international conference on machine learning*, pages 833–840, 2011.

[39] R. Setiono and H. Liu. Neural-network feature selector. *IEEE transactions on neural networks*, 8(3):654–662, 1997.

[40] D. Singh, H. Climente-González, M. Petrovich, E. Kawakami, and M. Yamada. Fsnet: Feature selection network on high-dimensional biological data. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2023.

[41] G. Sokar, E. Mocanu, D. C. Mocanu, M. Pechenizkiy, and P. Stone. Dynamic sparse training for deep reinforcement learning. *arXiv preprint arXiv:2106.04217*, 2021.

[42] G. Sokar, Z. Atashgahi, M. Pechenizkiy, and D. C. Mocanu. Where to pay attention in sparse training for feature selection? *Advances in Neural Information Processing Systems*, 35:1627–1642, 2022.

[43] G. Somepalli, M. Goldblum, A. Schwarzschild, C. B. Bruss, and T. Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*, 2021.

[44] E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for modern deep learning research. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13693–13696, 2020.

[45] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[46] M. Wojtas and K. Chen. Feature importance ranking for deep learning. *Advances in Neural Information Processing Systems*, 33:5105–5114, 2020.

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

[48] G. Yuan, X. Ma, W. Niu, Z. Li, Z. Kong, N. Liu, Y. Gong, Z. Zhan, C. He, Q. Jin, et al. Mest: Accurate and fast memory-economic sparse training framework on the edge. *Advances in Neural Information Processing Systems*, 34:20838–20850, 2021.

[49] R. Zhang, F. Nie, X. Li, and X. Wei. Feature selection with multi-view data: A survey. *Information Fusion*, 50:158–167, 2019.

Appendices

A Experimental Setup

A.1 Hyperparameters & Model Architecture

The architecture used in the experiments is an MLP with two hidden layers with 1000 and 100 hidden neurons in each layer, respectively. The hidden layer activation function is ReLU, and softmax is used for the output layer. We used a learning rate of 0.001 for all datasets. The L2 regularization term and sparsity level are optimized based on the validation loss in $[5e^{-5}, 0.0001, 0.001]$ and $[0.25, 0.5, 0.80, 0.9, 0.95, 0.98]$, respectively. ζ is set to 0.3. All datasets have been scaled to have zero mean and unit variance. The batch size for the datasets with less than 200 samples is set to 32 and for the rest of the datasets is set to 100. Adam optimizer is used for training the model. The maximum number of training epochs is 200; if the validation loss does not improve within 50 epochs, the training will end. The datasets are split into train, validation, and test sets with a split ratio $[65\%, 15\%, 20\%]$. The code is implemented in Python and using the PyTorch library [35]. The start of the code is *GraNet*[1] and *TANGOS*[2].

A.2 DST Algorithms

In Section 2.3.1, we explain the dynamic sparse training framework. In the experiments, we consider two dynamic sparse training algorithms: SET and RigL. These two algorithms are among the commonly used DST approaches in the literature to study the DST framework.

- **SET.** Sparse Evolutionary Training (SET) [32] is a pioneering approach that introduced the Dynamic Sparse Training framework. This method starts with initializing a sparse network with the desired sparsity level. Then, at each epoch or every few iterations, it updates the connectivity of the sparse neural network by dropping a fraction ζ of weight with the smallest magnitude and adding the same number of random weights back to the network. This dynamic process serves as a means to continuously refine and update the network's topology, to evolve its structure over time.
- **RigL.** The Rigged Lottery (RigL) [13] is another dynamic sparse training that evolves the topology of the sparse neural network by dynamically updating the connectivity. However, the difference compared to SET is that RigL adds the new weight based on the gradient information. It adds the non-existing weights having a large gradient magnitude to the network.

B Performance for Various K Values

While the results of the experiments in Section 4.2 are for when we select $K = 100$ features, we measure the performance when selecting $K \in \{25, 50, 75, 100, 200\}$. To summarize the performance of each method when selecting different numbers of K, we compute the average ranking score. To compute this score, in each experiment (each dataset and value of K), we rank the methods in terms of accuracy and give a score depending on their rank, where the best-performing method receives a score of 6 and the worst-performing method receives 1. Then, we average these scores for all the experiments for each method. The average ranking score for each K value is presented in Figure 7.

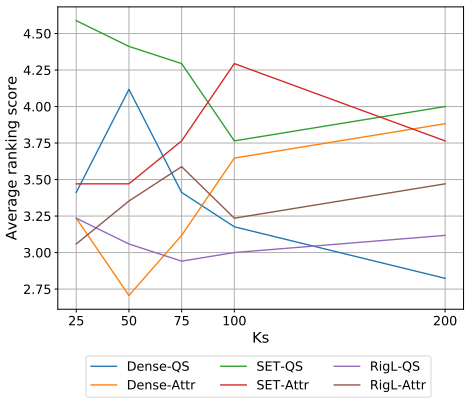


Figure 7: Average ranking score between the methods in Table 6 for various K values.

C Experiment on Synthetic Dataset

To evaluate the performance of the methods in a controlled environment, we design an experiment on an artificial dataset. We generate an artificial dataset with 200 features where only 100 of these features are informative and the rest of the features are noise. The task is a binary classification. We vary the number of samples in $\{100, 500, 1000, 10000\}$. We plot the fraction of the relevant features that each method can find (coverage) in Figure 8.

[1] https://github.com/VITA-Group/GraNet
[2] https://github.com/alanjeffares/TANGOS

As shown in Figure 8, as the number of samples increases the coverage also increases where the maximum value for most methods at 10000 samples. Neuron strength metric outperforms neuron attribution metric for higher number of samples (1000 and 10000) in dense and sparse networks. We observed this also in some cases on the madelon dataset in Table 3 in Section 4.2. Lasso is the worst performers among all methods. LassoNet and STG outperform the other methods in medium size of samples (500 and 1000), while at the high number of samples LassoNet achieves on par performance with QuickSelection. We should also highlight again that the computational costs of sparse neural network-based methods are much fewer than the competitors as shown in Section 4.4. However, the can achieve comparable performance to the competitors and find informative features.

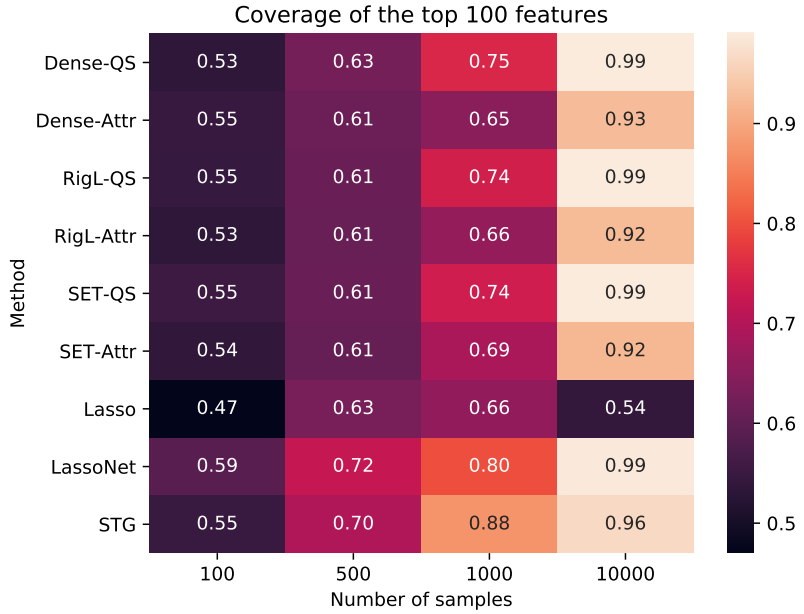


Figure 8: Coverage of the top 100 features for various methods considered in the experiments.

D Feature Selection Comparison

The overall comparison among the methods in shown in Table 6. In addition, more pairwise comparison (as shown in Table 3), are brought in Table 7.

Table 6: Feature selection results in terms of test classification accuracy [%] of an SVM classifier on the selected subset of (K=100). The values in the parenthesis show the sparsity level.

	Dataset	Dense-QS	Dense-Attr	SET-QS	SET-Attr	RigL-QS	RigL-Attr
Hand-written	MNIST	96.02 ± 0.13 (0.00)	96.23 ± 0.08 (0.00)	96.02 ± 0.10 (0.25)	**96.24 ± 0.13** (0.25)	96.06 ± 0.08 (0.50)	96.22 ± 0.10 (0.50)
	USPS	96.49 ± 0.11 (0.00)	96.83 ± 0.13 (0.00)	96.64 ± 0.13 (0.50)	96.77 ± 0.13 (0.50)	96.65 ± 0.16 (0.80)	**96.87 ± 0.17** (0.80)
	Gisette	96.36 ± 0.60 (0.00)	97.03 ± 0.26 (0.00)	96.65 ± 0.45 (0.50)	**97.10 ± 0.23** (0.50)	96.69 ± 0.53 (0.50)	97.07 ± 0.27 (0.50)
Image	Coil20	98.85 ± 0.49 (0.00)	98.82 ± 1.17 (0.00)	98.72 ± 0.31 (0.50)	98.02 ± 1.85 (0.50)	**99.06 ± 0.41** (0.80)	98.06 ± 1.87 (0.80)
	ORL	90.75 ± 3.96 (0.00)	89.38 ± 3.80 (0.00)	92.25 ± 2.15 (0.25)	89.50 ± 2.45 (0.25)	**92.50 ± 2.30** (0.25)	90.88 ± 3.26 (0.25)
	Yale	70.00 ± 7.48 (0.00)	63.94 ± 7.48 (0.00)	**74.55 ± 3.64** (0.25)	65.76 ± 7.05 (0.25)	71.52 ± 5.28 (0.50)	73.94 ± 5.94 (0.50)
Text	BASEHOCK	90.03 ± 1.75 (0.00)	92.83 ± 1.51 (0.00)	93.78 ± 1.01 (0.95)	**94.34 ± 0.66** (0.95)	84.61 ± 1.77 (0.95)	86.69 ± 2.20 (0.95)
	PCMAC	82.34 ± 1.90 (0.00)	84.96 ± 1.40 (0.00)	86.76 ± 1.41 (0.95)	**87.89 ± 1.26** (0.95)	79.31 ± 1.59 (0.95)	81.70 ± 2.29 (0.95)
	RELATHE	74.51 ± 4.16 (0.00)	80.24 ± 2.38 (0.00)	80.77 ± 1.35 (0.95)	**82.10 ± 1.02** (0.95)	73.22 ± 1.90 (0.98)	74.69 ± 2.96 (0.98)
Biological	Prostate_GE	**89.05 ± 3.05** (0.00)	**89.05 ± 2.18** (0.00)	87.62 ± 4.86 (0.50)	88.57 ± 2.33 (0.50)	85.71 ± 5.63 (0.80)	88.09 ± 3.19 (0.80)
	SMK	77.63 ± 2.94 (0.00)	**81.05 ± 3.07** (0.00)	79.47 ± 1.97 (0.50)	80.53 ± 2.41 (0.50)	79.47 ± 3.07 (0.50)	80.26 ± 3.95 (0.50)
	CLL	**86.52 ± 8.79** (0.00)	83.48 ± 5.43 (0.00)	76.09 ± 8.53 (0.25)	78.26 ± 8.02 (0.25)	80.00 ± 7.58 (0.25)	78.70 ± 7.64 (0.25)
	Carcinom	**90.00 ± 2.63** (0.00)	84.57 ± 4.64 (0.00)	89.14 ± 2.49 (0.50)	83.43 ± 5.83 (0.50)	85.14 ± 3.33 (0.50)	80.00 ± 6.26 (0.50)
	Lymphoma	70.00 ± 3.87 (0.00)	64.50 ± 5.68 (0.00)	**70.50 ± 6.10** (0.50)	66.00 ± 7.35 (0.50)	67.00 ± 4.58 (0.50)	65.50 ± 7.57 (0.50)
Mass Spectrometry	Arcene	60.75 ± 6.99 (0.00)	65.25 ± 7.62 (0.00)	66.25 ± 8.31 (0.25)	**67.75 ± 7.11** (0.25)	58.00 ± 6.00 (0.80)	57.00 ± 6.40 (0.80)
Time-series	HAR	94.56 ± 0.35 (0.00)	94.78 ± 0.37 (0.00)	94.59 ± 0.41 (0.25)	**94.90 ± 0.39** (0.25)	94.55 ± 0.23 (0.50)	94.88 ± 0.40 (0.50)
Speech	Isolet	94.54 ± 0.24 (0.00)	94.33 ± 1.31 (0.00)	95.22 ± 0.35 (0.95)	**95.53 ± 0.42** (0.95)	95.10 ± 0.51 (0.90)	94.72 ± 0.58 (0.90)
Noisy	Madelon	**84.42 ± 0.69** (0.00)	81.52 ± 2.55 (0.00)	78.40 ± 1.71 (0.95)	76.63 ± 2.36 (0.95)	79.80 ± 1.99 (0.98)	75.75 ± 3.50 (0.98)

Table 7: Feature selection results in terms of test classification accuracy [%] of an SVM classifier on the selected subset of (K=100). The values in the parenthesis show the sparsity level.

	Dense-QS	**Dense-Attr**
MNIST	96.02 ± 0.13 (0.00)	**96.23 ± 0.08** (0.00)
USPS	96.49 ± 0.11 (0.00)	**96.83 ± 0.13** (0.00)
Gisette	96.36 ± 0.60 (0.00)	**97.03 ± 0.26** (0.00)
Coil20	**98.85 ± 0.49** (0.00)	98.82 ± 1.17 (0.00)
ORL	**90.75 ± 3.96** (0.00)	89.38 ± 3.80 (0.00)
Yale	**70.00 ± 7.48** (0.00)	63.94 ± 7.48 (0.00)
BASEHOCK	90.03 ± 1.75 (0.00)	**92.83 ± 1.51** (0.00)
PCMAC	82.34 ± 1.90 (0.00)	**84.96 ± 1.40** (0.00)
RELATHE	74.51 ± 4.16 (0.00)	**80.24 ± 2.38** (0.00)
Prostate_GE	**89.05 ± 3.05** (0.00)	**89.05 ± 2.18** (0.00)
SMK	77.63 ± 2.94 (0.00)	**81.05 ± 3.07** (0.00)
CLL	**86.52 ± 8.79** (0.00)	83.48 ± 5.43 (0.00)
Carcinom	**90.00 ± 2.63** (0.00)	84.57 ± 4.64 (0.00)
Lymphoma	**70.00 ± 3.87** (0.00)	64.50 ± 5.68 (0.00)
Arcene	60.75 ± 6.99 (0.00)	**65.25 ± 7.62** (0.00)
HAR	94.56 ± 0.35 (0.00)	**94.78 ± 0.37** (0.00)
Isolet	**94.54 ± 0.24** (0.00)	94.33 ± 1.31 (0.00)
Madelon	**84.42 ± 0.69** (0.00)	81.52 ± 2.55 (0.00)

(a) Dense-QS vs. Dense-Attr.

	SET-QS	**RigL-QS**
MNIST	96.02 ± 0.10 (0.25)	**96.06 ± 0.08** (0.50)
USPS	96.64 ± 0.13 (0.50)	**96.65 ± 0.16** (0.80)
Gisette	96.65 ± 0.45 (0.50)	**96.69 ± 0.53** (0.50)
Coil20	98.72 ± 0.31 (0.50)	**99.06 ± 0.41** (0.80)
ORL	92.25 ± 2.15 (0.25)	**92.50 ± 2.30** (0.25)
Yale	**74.55 ± 3.64** (0.25)	71.52 ± 5.28 (0.50)
BASEHOCK	**93.78 ± 1.01** (0.95)	84.61 ± 1.77 (0.95)
PCMAC	**86.76 ± 1.41** (0.95)	79.31 ± 1.59 (0.95)
RELATHE	**80.77 ± 1.35** (0.95)	73.22 ± 1.90 (0.98)
Prostate_GE	**87.62 ± 4.86** (0.50)	85.71 ± 5.63 (0.80)
SMK	**79.47 ± 1.97** (0.50)	**79.47 ± 3.07** (0.50)
CLL	76.09 ± 8.53 (0.25)	**80.00 ± 7.58** (0.25)
Carcinom	**89.14 ± 2.49** (0.50)	85.14 ± 3.33 (0.50)
Lymphoma	**70.50 ± 6.10** (0.50)	67.00 ± 4.58 (0.50)
Arcene	**66.25 ± 8.31** (0.25)	58.00 ± 6.00 (0.80)
HAR	**94.59 ± 0.41** (0.25)	94.55 ± 0.23 (0.50)
Isolet	**95.22 ± 0.35** (0.95)	95.10 ± 0.51 (0.90)
Madelon	78.40 ± 1.71 (0.95)	**79.80 ± 1.99** (0.98)

(b) RigL-QS vs. SET-QS.

	RigL-Attr	**RigL-QS**
MNIST	**96.22 ± 0.10** (0.50)	96.06 ± 0.08 (0.50)
USPS	**96.87 ± 0.17** (0.80)	96.65 ± 0.16 (0.80)
Gisette	**97.07 ± 0.27** (0.50)	96.69 ± 0.53 (0.50)
Coil20	98.06 ± 1.87 (0.80)	**99.06 ± 0.41** (0.80)
ORL	90.88 ± 3.26 (0.25)	**92.50 ± 2.30** (0.25)
Yale	**73.94 ± 5.94** (0.50)	71.52 ± 5.28 (0.50)
BASEHOCK	**86.69 ± 2.20** (0.95)	84.61 ± 1.77 (0.95)
PCMAC	**81.70 ± 2.29** (0.95)	79.31 ± 1.59 (0.95)
RELATHE	**74.69 ± 2.96** (0.98)	73.22 ± 1.90 (0.98)
Prostate_GE	**88.09 ± 3.19** (0.80)	85.71 ± 5.63 (0.80)
SMK	**80.26 ± 3.95** (0.50)	79.47 ± 3.07 (0.50)
CLL	78.70 ± 7.64 (0.25)	**80.00 ± 7.58** (0.25)
Carcinom	80.00 ± 6.26 (0.50)	**85.14 ± 3.33** (0.50)
Lymphoma	65.50 ± 7.57 (0.50)	**67.00 ± 4.58** (0.50)
Arcene	57.00 ± 6.40 (0.80)	**58.00 ± 6.00** (0.80)
HAR	**94.88 ± 0.40** (0.50)	94.55 ± 0.23 (0.50)
Isolet	94.72 ± 0.58 (0.90)	**95.10 ± 0.51** (0.90)
Madelon	75.75 ± 3.50 (0.98)	**79.80 ± 1.99** (0.98)

(c) SET-QS vs. SET-Attr.

E Feature Importance Metric Analysis

In the experiments in the paper, we calculate the neuron importance based on the summation of the importance within all training epochs. However, we considered two other ways to compute the importance: the summation of importance within last training epoch, and the importance in the last iteration. The results of these two approaches are summarized in Tables 8 and 9. As can be seen from this Table, considering the importance within all training epochs, as done in Section 4.2 leads to better results. This shows that the history of the importance, even when measured at the first epochs is also beneficial to the performance.

Table 8: Feature selection results in terms of test classification accuracy [%] of an SVM classifier on the selected subset of (K=100). The importance metric is computed based on the summation of importance values (for all iterations) within the last training epoch for all methods. The values in the parenthesis show the sparsity level.

	Dataset	**Dense-QS**	**Dense-Attr**	**SET-QS**	**SET-Attr**	**RigL-QS**	**RigL-Attr**
Hand-written	MNIST	95.97 ± 0.24 (0.00)	96.21 ± 0.17 (0.00)	95.89 ± 0.22 (0.25)	**96.23 ± 0.17** (0.25)	95.90 ± 0.21 (0.50)	96.19 ± 0.20 (0.50)
	USPS	96.60 ± 0.20 (0.00)	96.81 ± 0.19 (0.00)	96.67 ± 0.26 (0.50)	**96.91 ± 0.12** (0.50)	96.58 ± 0.14 (0.80)	96.86 ± 0.14 (0.80)
	Gisette	94.52 ± 1.26 (0.00)	96.66 ± 0.40 (0.00)	93.91 ± 1.79 (0.50)	**96.99 ± 0.40** (0.50)	95.40 ± 0.60 (0.50)	96.80 ± 0.36 (0.50)
Image	Coil20	98.82 ± 0.59 (0.00)	98.82 ± 1.20 (0.00)	98.75 ± 0.32 (0.50)	98.75 ± 0.35 (0.50)	**99.03 ± 0.43** (0.80)	98.78 ± 0.61 (0.80)
	ORL	**90.88 ± 2.56** (0.00)	89.25 ± 3.92 (0.00)	90.62 ± 2.32 (0.25)	88.62 ± 2.12 (0.25)	90.75 ± 1.79 (0.25)	90.38 ± 3.40 (0.25)
	Yale	68.48 ± 5.94 (0.00)	59.39 ± 8.48 (0.00)	**72.73 ± 7.17** (0.25)	63.33 ± 7.95 (0.25)	68.49 ± 5.94 (0.50)	71.82 ± 6.78 (0.50)
Text	BASEHOCK	75.59 ± 6.68 (0.00)	86.69 ± 2.44 (0.00)	92.00 ± 1.60 (0.95)	90.85 ± 1.84 (0.95)	87.84 ± 1.95 (0.95)	**92.06 ± 1.81** (0.95)
	PCMAC	78.20 ± 2.98 (0.00)	80.69 ± 1.28 (0.00)	**85.50 ± 1.21** (0.95)	84.73 ± 1.47 (0.95)	81.65 ± 2.24 (0.95)	84.04 ± 2.05 (0.95)
	RELATHE	64.97 ± 6.84 (0.00)	73.53 ± 3.14 (0.00)	**80.42 ± 1.71** (0.95)	78.57 ± 2.97 (0.95)	73.36 ± 1.83 (0.98)	78.08 ± 2.62 (0.98)
Biological	Prostate_GE	89.05 ± 4.29 (0.00)	**90.00 ± 1.43** (0.00)	89.52 ± 1.90 (0.50)	88.09 ± 3.19 (0.50)	87.62 ± 3.16 (0.80)	87.62 ± 2.33 (0.80)
	SMK	75.53 ± 4.25 (0.00)	79.74 ± 3.13 (0.00)	80.53 ± 4.11 (0.50)	**81.05 ± 3.29** (0.50)	78.95 ± 4.24 (0.50)	80.26 ± 5.16 (0.50)
	CLL	**79.57 ± 14.43** (0.00)	77.39 ± 7.97 (0.00)	73.91 ± 16.38 (0.25)	74.78 ± 13.72 (0.25)	71.74 ± 7.84 (0.25)	72.61 ± 7.79 (0.25)
	Carcinom	**84.57 ± 5.74** (0.00)	77.14 ± 7.23 (0.00)	**84.57 ± 6.29** (0.50)	75.71 ± 4.09 (0.50)	80.57 ± 6.49 (0.50)	78.29 ± 5.14 (0.50)
	Lymphoma	70.00 ± 5.00 (0.00)	64.00 ± 4.36 (0.00)	**71.50 ± 7.43** (0.50)	63.00 ± 5.57 (0.50)	65.50 ± 4.15 (0.50)	63.00 ± 6.00 (0.50)
Mass Spectrometry	Arcene	65.50 ± 9.07 (0.00)	**65.75 ± 7.59** (0.00)	63.50 ± 8.08 (0.25)	63.75 ± 6.64 (0.25)	56.75 ± 5.60 (0.80)	63.25 ± 5.60 (0.80)
Time-series	HAR	94.72 ± 0.30 (0.00)	94.85 ± 0.25 (0.00)	94.30 ± 0.33 (0.25)	**94.88 ± 0.30** (0.25)	94.64 ± 0.33 (0.50)	94.63 ± 0.34 (0.50)
Speech	Isolet	94.11 ± 0.46 (0.00)	94.23 ± 0.69 (0.00)	95.57 ± 0.21 (0.95)	**95.64 ± 0.25** (0.95)	95.04 ± 0.42 (0.90)	95.01 ± 0.42 (0.90)
Noisy	Madelon	**84.30 ± 0.96** (0.00)	82.25 ± 2.20 (0.00)	78.45 ± 2.44 (0.95)	75.87 ± 2.94 (0.95)	79.13 ± 2.20 (0.98)	72.30 ± 3.86 (0.98)

Table 9: Feature selection results in terms of test classification accuracy [%] of an SVM classifier on the selected subset of (K=100). The importance metric is computed at the last training iteration for all methods. The values in the parenthesis show the sparsity level.

	Dataset	Dense-QS	Dense-Attr	SET-QS	SET-Attr	RigL-QS	RigL-Attr
Hand-written	MNIST	96.10 ± 0.09 (0.00)	96.13 ± 0.20 (0.00)	95.97 ± 0.22 (0.25)	96.06 ± 0.20 (0.25)	95.99 ± 0.20 (0.50)	**96.16 ± 0.34** (0.50)
	USPS	96.55 ± 0.24 (0.00)	**96.81 ± 0.24** (0.00)	96.60 ± 0.17 (0.50)	96.76 ± 0.19 (0.50)	96.58 ± 0.19 (0.80)	96.77 ± 0.18 (0.80)
	Gisette	93.36 ± 2.76 (0.00)	96.70 ± 0.71 (0.00)	94.37 ± 1.54 (0.50)	96.82 ± 0.37 (0.50)	95.17 ± 0.76 (0.50)	**96.86 ± 0.52** (0.50)
Image	Coil20	98.82 ± 0.59 (0.00)	98.61 ± 1.51 (0.00)	98.78 ± 0.28 (0.50)	98.68 ± 0.43 (0.50)	**99.03 ± 0.43** (0.80)	98.78 ± 0.68 (0.80)
	ORL	**91.00 ± 2.61** (0.00)	87.75 ± 4.36 (0.00)	90.75 ± 2.18 (0.25)	90.00 ± 2.17 (0.25)	90.75 ± 1.79 (0.25)	89.00 ± 3.00 (0.25)
	Yale	68.48 ± 5.62 (0.00)	63.64 ± 7.30 (0.00)	**71.82 ± 7.55** (0.25)	61.82 ± 9.79 (0.25)	68.18 ± 5.63 (0.50)	65.15 ± 7.81 (0.50)
Text	BASEHOCK	74.26 ± 7.06 (0.00)	85.31 ± 2.63 (0.00)	**91.80 ± 1.53** (0.95)	90.20 ± 1.78 (0.95)	88.22 ± 2.34 (0.95)	91.13 ± 2.32 (0.95)
	PCMAC	77.22 ± 4.17 (0.00)	80.57 ± 1.17 (0.00)	**85.06 ± 1.65** (0.95)	84.55 ± 1.38 (0.95)	82.29 ± 1.61 (0.95)	83.73 ± 2.08 (0.95)
	RELATHE	66.64 ± 4.95 (0.00)	72.87 ± 2.46 (0.00)	**80.10 ± 1.97** (0.95)	78.74 ± 3.62 (0.95)	72.69 ± 2.39 (0.98)	78.15 ± 2.50 (0.98)
Biological	Prostate_GE	89.05 ± 4.29 (0.00)	**90.00 ± 1.43** (0.00)	89.52 ± 1.90 (0.50)	89.05 ± 3.05 (0.50)	87.14 ± 3.05 (0.80)	86.67 ± 3.56 (0.80)
	SMK	76.32 ± 2.88 (0.00)	78.68 ± 3.62 (0.00)	**81.32 ± 4.47** (0.50)	80.26 ± 3.95 (0.50)	79.21 ± 3.62 (0.50)	80.26 ± 4.74 (0.50)
	CLL	**79.57 ± 11.35** (0.00)	70.43 ± 8.65 (0.00)	77.39 ± 14.39 (0.25)	71.30 ± 13.07 (0.25)	72.61 ± 7.54 (0.25)	71.74 ± 12.34 (0.25)
	Carcinom	84.00 ± 6.41 (0.00)	76.00 ± 6.54 (0.00)	**84.86 ± 6.13** (0.50)	73.14 ± 3.88 (0.50)	81.43 ± 6.42 (0.50)	72.57 ± 6.79 (0.50)
	Lymphoma	69.50 ± 5.22 (0.00)	64.00 ± 3.74 (0.00)	**71.50 ± 7.43** (0.50)	65.00 ± 5.92 (0.50)	65.00 ± 4.47 (0.50)	61.50 ± 6.34 (0.50)
Mass Spectrometry	Arcene	**66.50 ± 8.00** (0.00)	66.00 ± 7.84 (0.00)	62.75 ± 6.75 (0.25)	62.00 ± 7.31 (0.25)	54.25 ± 6.71 (0.80)	60.75 ± 5.71 (0.80)
Time-series	HAR	94.35 ± 0.32 (0.00)	**94.72 ± 0.46** (0.00)	94.29 ± 0.48 (0.25)	94.70 ± 0.63 (0.25)	94.62 ± 0.34 (0.50)	94.72 ± 0.30 (0.50)
Speech	Isolet	94.08 ± 0.45 (0.00)	94.45 ± 0.56 (0.00)	**95.56 ± 0.30** (0.95)	95.42 ± 0.42 (0.95)	95.04 ± 0.37 (0.90)	94.31 ± 1.27 (0.90)
Noisy	Madelon	**83.68 ± 1.89** (0.00)	81.68 ± 3.00 (0.00)	78.13 ± 2.41 (0.95)	76.40 ± 2.15 (0.95)	79.30 ± 1.61 (0.98)	72.02 ± 2.94 (0.98)