SAND: One-Shot Feature Selection with Additive Noise Distortion

Anonymous Author(s)
Affiliation
Address
email

Abstract

In data-driven applications, feature selection reduces the input dimension by partitioning features into the most informative and least informative subsets. This process leads to higher learning accuracy, lower computational costs, and improved interpretability. Current state-of-the-art methods require post-selection retraining to ensure that the model can learn and generalize effectively based on the reduced feature set. We introduce a non-intrusive layer that, given a number k, selects the k most informative features during the training phase of a neural network. The selection process does not impose any alteration to the loss function or the original network architecture. Our proposed layer takes the following simple form for each feature:

$$\tilde{x}_i = a_i x_i + (1 - a_i) z_i$$

where x_i is the input feature, $\tilde{x}_i$ the output, z_i a Gaussian noise, and a_i trainable gain such that $\sum_i a_i^2 = k$. Our approach, despite its apparent simplicity in both theory and implementation, provides a one-shot solution that enables simultaneous feature selection and network training by automatically clustering k of the a_i gains around 1 and the rest around 0. This clustering effect is achieved through the weighted additive noise distortion together with the gain normalization operation linked to k. Our method was tested on different datasets and benchmarked against established feature selection algorithms. Results show that our method consistently outperforms or matches existing methods, and notably, does not require hyperparameter search to control the number of selected features or any retraining phase after feature selection. Furthermore, we provide theoretical insights of our method in the context of linear regression.

1 Introduction

Feature selection is a fundamental problem in high-dimensional statistics and machine learning [8, 13]. Unlike feature extraction techniques that alter features' semantics by creating new ones in a lower dimensional space, feature selection involves the identification and retention of the most informative features while discarding irrelevant or redundant ones. This preservation enhances the interpretability and explainability of predictive models, particularly critical in domains like medicine and biology where gene selection is a focal application [9]. By retaining the original features, researchers can directly relate model outputs to the underlying data, facilitating insights and hypothesis generation. Furthermore, feature selection not only contributes to storage reduction by eliminating unnecessary data points, optimizing memory usage, and enhancing computational efficiency, but also aids in reducing model size and complexity. By selecting a subset of input features, models can improve performance and generalization capabilities crucial for mitigating overfitting and addressing the curse of dimensionality. Moreover, in applications where sensing hardware costs or energy

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

consumption are major concerns, such as in IoT devices or sensor-based systems, feature selection can inform the design of simpler and more cost-effective hardware by ensuring that only relevant features are sensed or measured, thereby conserving resources without compromising performance.

Feature selection methods can be broadly categorized into two main groups: unsupervised and supervised. Unsupervised methods often involve an analysis of the relations between input features through methods like clustering [10], matrix factorization [23], and the use of autoencoder neural networks [1]. These methods are particularly useful when labeled data is scarce or unavailable, allowing for the exploration of inherent data structures and patterns. On the other hand, supervised methods leverage the availability of labeled data to guide the selection process. Within the realm of supervised methods, there exist model-independent and model-dependent approaches. Model-independent methods, also known as filter-based, rely on statistical tests and information-theoretic metrics to evaluate feature relevance with respect to the target variable, irrespective of the underlying machine learning model [25, 5]. While these methods are computationally efficient and can handle high-dimensional data, they may overlook complex interactions between features. Model-dependent methods, on the other hand, tailor feature selection to specific machine learning models or architectures. This category can be further divided into wrapper and embedded methods. Wrapper methods [11] involve a search process guided by the final performance of a learning model, such as classifier accuracy. Examples include greedy sequential feature selection via forward or backward search [6], SHAP (SHapley Additive exPlanations) values calculation [16], in addition to combinatorial optimization and metaheuristic search algorithms [27, 7]. Wrapper methods offer the advantage of considering feature interactions but may suffer from high computational costs due to intensive search over the input space, which is highly impractical for complex models and large feature dimensions. In contrast, embedded methods rank features based on metrics intrinsically learned during model training, seamlessly integrating feature selection into the learning process. Examples include feature importance for tree-based algorithms [2], Recursive Feature Elimination for Support Vector Machine (RFE-SVM) [9], sparsity-promoting models [21], and other deep learning techniques [19, 22]. Such methods enable an automatic selection of relevant features during training and can effectively handle non-trivial relationships in data.

Related works

Given the pervasive adoption of deep learning in recent years, this work concentrates on embedded feature selection techniques tailored to neural networks. Within this domain, a multitude of approaches have emerged, predominantly centered around various adaptations of LASSO-based regularization [17, 28, 14, 12, 3], the addition of stochastic gates [20, 24], the use of attention mechanisms [15, 26], and the application of saliency maps [4] to solve the feature selection problem on non-linear models. For instance, Sequential LASSO [17] provides an efficient implementation of greedy LASSO to recursively select input features, while Group LASSO [28, 18] further modifies the objective function to encourage sparsity at the group level. In LassoNet [12], a skip linear connection is added to the neural network with two types of regularization parameters. A continuous search is then applied using a hierarchical proximity algorithm, which combines a proximal gradient descent method with a hierarchical feature selection. Alternatively, to impose sparsity and overcome the limitations of applying gradient descent on ℓ_1 regularized objective functions, [24] introduces a continuous relaxation of Bernoulli gates that are attached to the input features. A Gaussian-based regularization is then added to the objective function and grid-search over the regularization parameter is applied to select the required number of features. Lately, the attention mechanism is being employed to relate a trainable softmax mask to feature importance, and hence perform embedded feature selection by adaptively estimating marginal feature gains over multiple rounds [26].

Contributions

The existing methods mentioned above typically necessitate alterations to the objective function or significant modifications to the neural network architecture involving the addition of new connections. Consequently, feature selection is often a separate phase followed by a retraining phase on the selected features, or it requires some kind of hyperparameter tuning to control the number of selected features [24, 12, 26]. In this work, we propose a novel, yet exceptionally simple, method for one-shot feature selection. It involves the integration of a simple constrained weighted additive noise layer at the neural network's input. The constrained stochasticity helps the network generate a polarized input space and effectively select the desired number of features during training. As a result,

the network architecture inherently converges to its final form, which can be directly used for inference without necessitating any additional retraining. The constraint on the weights is imposed by construction through a normalization operation and requires no regularization terms in the objective function. The proposed layer imposes negligible computational overhead and can be seamlessly incorporated, akin to the addition of Dropout or Batch Normalization layers. Through this layer, direct control over the number of selected features is enabled without the need for additional grid search or further tuning of regularization terms. The simplicity of our method does not compromise the final prediction performance of the neural network. In this work, we conduct an extensive benchmarking study against state-of-the-art feature selection methods using common datasets, showcasing our method's effective competition in classification accuracy against existing approaches. Furthermore, we provide theoretical insights by demonstrating that our method, when applied to linear regression, promotes the selection of a predefined number of features on an equivalent problem.

2 Selection with Additive Noise Distortion (SAND)

In a typical supervised learning problem, we are tasked to map a set of input vectors to predefined outputs. These input vectors consist of various features. However, not all features are equally important in determining the output. Some may be irrelevant, while others might contain redundant information. This leads us to the concept of feature selection: the quest to identify the subset of features that provide sufficient information to determine the output accurately. In real-world applications, the number of features to select is typically pre-defined due to constraints on data acquisition burden, computation cost, or memory footprint.

Consider an n-dimensional feature vector $\underline{\boldsymbol{x}} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)^\top$ (which can be of any shape; but for the sake of simplicity in notations, we assume it to be $n \times 1$) to be mapped to the output vector $\boldsymbol{y}$[1]. Figure 1(a) depicts a typical neural network solution to this problem. Now, assume we are interested in finding the k dimensions that yield the highest performance, with $k \leq n$.

Our idea is to multiply each feature $\boldsymbol{x}_i$ with a gain a_i and add a zero-mean Gaussian noise with the standard deviation of $|(1 - a_i)\sigma|$ to it before feeding it to the neural network. Here, σ is a fixed scalar. Moreover, we constrain the vector $\underline{a} = (a_1, a_2, \ldots, a_n)^\top$ to have the ℓ_α-norm equal to $k^{\frac{1}{\alpha}}$ for a pre-selected $\alpha > 0$. Thus, we define

$$\tilde{\underline{\boldsymbol{x}}} = \underline{a} \odot \underline{\boldsymbol{x}} + (\underline{1} - \underline{a}) \odot \underline{\boldsymbol{z}} \tag{1}$$

where

$$\|\underline{a}\|_\alpha^\alpha = k. \tag{2}$$

We then feed $\tilde{\underline{\boldsymbol{x}}}$ to the neural network during the training phase instead of $\underline{\boldsymbol{x}}$ as illustrated in Figure 1(b)). Here, $\underline{\boldsymbol{z}}$ is a Gaussian vector with i.i.d. entries with zero-mean and standard deviation σ. In this setting, when a_i is close to 1, $\tilde{\boldsymbol{x}}_i$ is close to noiseless $\boldsymbol{x}_i$, and when a_i is close to 0, $\tilde{\boldsymbol{x}}_i$ becomes almost pure noise (the signal-to-noise ratio is proportional to $\frac{a_i^2}{(1-a_i)^2\sigma^2}$). During the training phase, we allow the a_i's to be trained alongside the other parameters of the network. The architecture of the neural network and the loss function remain unchanged; the only difference is the addition of n extra parameters (a_i's) to optimize. As training progresses, we observe that k of the a_i's cluster around 1, indicating the selected features, while the remaining a_i's cluster around 0, indicating the neglected features. We refer to this approach as SAND, which stands for Selection with Additive Noise Distortion.

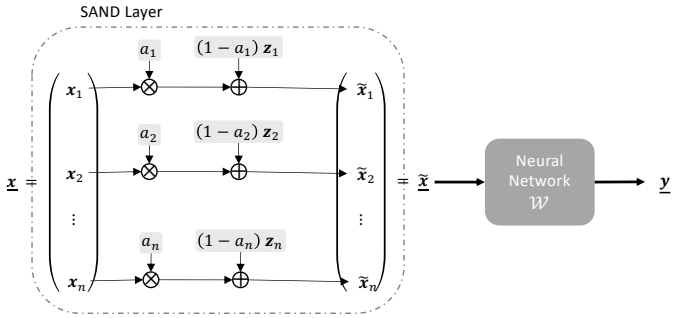
Remark 2.1 *The two operations of the SAND layer in (1) and (2) are implemented together. This means the constraint (2) is enforced by construction where we normalize the a_i's by their ℓ_α-norm inside the layer, without adding any regularization term to the loss function. Hence, the SAND layer takes this form in practice:*

$$\tilde{\underline{\boldsymbol{x}}} = \frac{\underline{a}}{\|\underline{a}\|_\alpha} k^{\frac{1}{\alpha}} \odot \underline{\boldsymbol{x}} + \left(\underline{1} - \frac{\underline{a}}{\|\underline{a}\|_\alpha} k^{\frac{1}{\alpha}}\right) \odot \underline{\boldsymbol{z}}. \tag{3}$$

[1]Throughout the paper, we use small characters to denote scalars, underlined characters to indicate vectors, capital characters for matrices and bold font for random objects

$$\mathcal{W}^* = \text{argmin}\, \mathcal{L}\left(\mathcal{W} \,\middle|\, \underline{x}, \underline{y}\right)$$

(a) Vanilla Neural Network



$$\sum_{i=1}^{n} |a_i|^{\alpha} = k \qquad \underline{a}^*, \mathcal{W}^* = \text{argmin}\, \mathcal{L}\left(\underline{a}, \mathcal{W} \,\middle|\, \underline{x}, \underline{y}\right)$$

(b) Neural Network with the SAND layer

Figure 1: Neural network architecture and the loss function before and after adding the SAND layer. Here, $\mathcal{L}$ and $\mathcal{W}$ indicate the loss function and the trainable parameters of the neural network respectively.

Notice that if there is no noise $\underline{z}$ (i.e., $\sigma = 0$), the a_i's would be absorbed in the weights of the first layer of the neural network. Additionally, if $k = n$, all a_i's can become 1 and then $\underline{\tilde{x}}$ will be identical to $\underline{x}$ without any noise. Given that the noise is independent of the data and lacks information about the output, the network naturally adjusts to mitigate its impact during training.

The first non-trivial property is that there is always an optimal $\underline{a}$ whose entries are between 0 and 1. Here, optimal means with respect to the loss function of the network. To prove that, assume there is an optimal $\underline{a}$ that has an $a_j < 0$. By replacing a_j by $-a_j$, while the constraint (2) still holds, $\tilde{\boldsymbol{x}}_j$ is a less noisy version of $\boldsymbol{x}_j$. On the other hand, if there is an optimal $\underline{a}$ that has an $a_j > 1$, we can decrease a_j to 1, which results in $\tilde{\boldsymbol{x}}_j$ becoming a noiseless copy of $\boldsymbol{x}_j$, and increase other a_i's that are less than 1 towards 1 to satisfy the condition (2); it decreases the noise added to those features as well. Therefore, we can confine the search space of $\underline{a}$ to the vectors that have all entries between 0 and 1, i.e., $\underline{a} = (a_1, a_2, \ldots, a_n)^\top$ that have

$$0 \leq a_i \leq 1 \quad \text{for } \; i = 1, \ldots, n. \tag{4}$$

Now, we analyze what is happening during the training. Intuitively, a more informative feature x_j will get a gain a_j closer to 1 so that it will be passed to the neural network with less noise. Due to the constraint (2), this automatically yields smaller gains $a_{j'}$ for other features which are less informative. Consequently, the less informative features have now become noisier, which makes them even less informative and pushes them to get even smaller gains. This leaves more room, due to (2), for the more informative features to get their gains closer to 1 and become less noisy. This reinforcing loop, summarized in Figure 2, results in polarization of the gains around 1 and 0. Ideally, we will end up with a vector $\underline{a}$ which has k entries equal to 1, indicating the selected features, and the rest equal to 0, indicating the neglected features. However, since in practice the smallest values have not necessarily converged to absolute zero, at the end of the training phase, we keep the top k gains intact and manually set the $n - k$ smallest gains to 0, which is equivalent to removing the corresponding features. Notice that the features with small gains are such noisy that they are already implicitly neglected through other parts of the neural network.

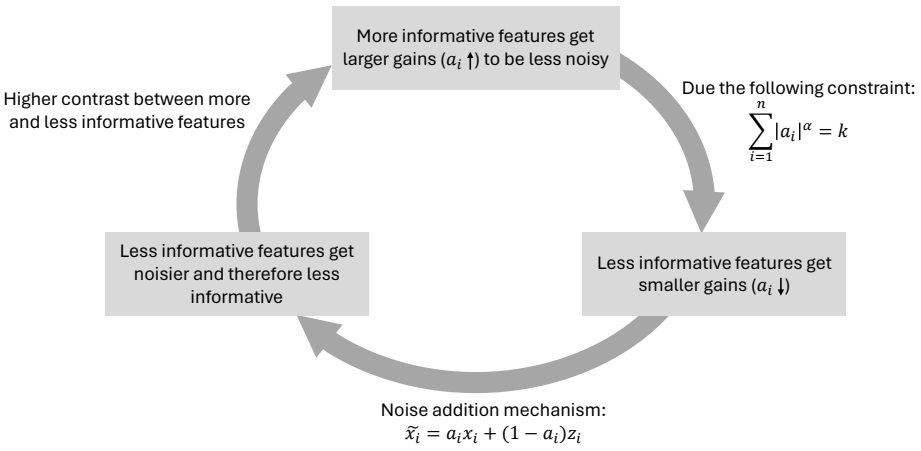


Figure 2: Reinforcing loop that results in polarization of the gains.

Linear Regression

Here, we mathematically show that in the case of linear regression, adding the SAND layer introduced above is equivalent to adding a term in the loss function that promotes selection of k features.

In linear regression problem, the loss function is

$$\mathcal{L}\left(\mathrm{W}, \underline{b}\right) = \mathbb{E}_{\underline{\boldsymbol{x}}, \underline{\boldsymbol{y}}} \left\{ \left\| \underline{\boldsymbol{y}} - \mathrm{W}\underline{\boldsymbol{x}} - \underline{b} \right\|_2^2 \right\}, \tag{5}$$

where $\mathbb{E}$ denotes the expected value, W is the coefficient matrix and $\underline{b}$ is the bias vector. Thus, optimal solution is

$$\mathrm{W}^*, \underline{b}^* = \underset{\mathrm{W}, \underline{b}}{\operatorname{argmin}} \, \mathcal{L}(\mathrm{W}, \underline{b}). \tag{6}$$

Now, we add the SAND layer in the beginning, i.e.,

$$\tilde{\underline{\boldsymbol{x}}} = \underline{a} \odot \underline{\boldsymbol{x}} + (\underline{1} - \underline{a}) \odot \underline{\boldsymbol{z}} \quad \text{ such that } \quad \|\underline{a}\|_{\alpha}^{\alpha} = k. \tag{7}$$

We get

$$\begin{aligned} \mathcal{L}\left(\underline{a}, \mathrm{W}, \underline{b}\right) &= \mathbb{E}_{\tilde{\underline{\boldsymbol{x}}}, \underline{\boldsymbol{y}}} \left\{ \left\| \underline{\boldsymbol{y}} - \mathrm{W}\tilde{\underline{\boldsymbol{x}}} - \underline{b} \right\|_2^2 \right\} \\ &= \mathbb{E}_{\underline{\boldsymbol{x}}, \underline{\boldsymbol{y}}, \underline{\boldsymbol{z}}} \left\{ \left\| \underline{\boldsymbol{y}} - \mathrm{W}\left(\underline{a} \odot \underline{\boldsymbol{x}} + (\underline{1} - \underline{a}) \odot \underline{\boldsymbol{z}}\right) - \underline{b} \right\|_2^2 \right\} \\ &= \mathbb{E}_{\underline{\boldsymbol{x}}, \underline{\boldsymbol{y}}} \left\{ \left\| \underline{\boldsymbol{y}} - \mathrm{W}\left(\underline{a} \odot \underline{\boldsymbol{x}}\right) - \underline{b} \right\|_2^2 \right\} + \sum_{i=1}^{n} w_i^2 \left(1 - a_i\right)^2 \sigma^2 \end{aligned} \tag{8}$$

where w_i is the ℓ_2-norm of the i^{th} column of W. Define the matrix $\overline{\mathrm{W}}$ to be the matrix W that its i^{th} column is multiplied by a_i for $i = 1, \ldots, n$. Rewriting (8), we obtain

$$\mathcal{L}\left(\underline{a}, \overline{\mathrm{W}}, \underline{b}\right) = \mathbb{E}_{\underline{\boldsymbol{x}}, \underline{\boldsymbol{y}}} \left\{ \left\| \underline{\boldsymbol{y}} - \overline{\mathrm{W}}\underline{\boldsymbol{x}} - \underline{b} \right\|_2^2 \right\} + \sigma^2 \sum_{i=1}^{n} \overline{w}_i^2 \left(\frac{1}{a_i} - 1 \right)^2 \tag{9}$$

where $\overline{w}_i$ is the ℓ_2-norm of the i^{th} column of $\overline{\mathrm{W}}$. Using the Lagrange multiplier method for constrained optimization, we obtain

$$\frac{\partial}{\partial a_j} \mathcal{L}\left(\underline{a}, \overline{\mathrm{W}}, \underline{b}\right) = -\lambda \frac{\partial}{\partial a_j} \|\underline{a}\|_{\alpha}^{\alpha} \tag{10}$$

where λ is a scalar and the right side of the equation is from the constraint in (7). Thus, we have

$$2\sigma^2 \overline{w}_j^2 \frac{1}{a_j^2} \left(\frac{1}{a_j} - 1 \right) = \lambda \, \alpha \, \operatorname{sgn}\left(a_j\right) \left|a_j\right|^{\alpha - 1} \tag{11}$$

which leads to

$$\sigma^2 \overline{w}_j^2 \left(\frac{1}{a_j} - 1 \right)^2 = \frac{\lambda}{2} \alpha \left| a_j \right|^\alpha (1 - a_j) . \tag{12}$$

By summing over j's and incorporating the constraint in (7), we get

$$\sigma^2 \sum_{j=1}^{n} \overline{w}_j^2 \left(\frac{1}{a_j} - 1 \right)^2 = \frac{\lambda}{2} \alpha \sum_{j=1}^{n} \left| a_j \right|^\alpha (1 - a_j) = \frac{\lambda}{2} \alpha \left(k - \sum_{j=1}^{n} a_j \left| a_j \right|^\alpha \right) \tag{13}$$

Combining (13) and (9), we obtain

$$\mathcal{L} \left(\underline{a}, \overline{\mathrm{W}}, \underline{b} \right) = \mathbb{E}_{\underline{\boldsymbol{x}}, \underline{\boldsymbol{y}}} \left\{ \left\| \underline{\boldsymbol{y}} - \overline{\mathrm{W}} \underline{\boldsymbol{x}} - \underline{b} \right\|_2^2 \right\} + \frac{\lambda}{2} \alpha k - \frac{\lambda}{2} \alpha \sum_{i=1}^{n} a_i \left| a_i \right|^\alpha \tag{14}$$

Remember that we can confine the search space to the a_i's between 0 and 1. Hence, according to (12), we have $\lambda \geq 0$. Therefore, the term at the end of (14) achieves its minima when there are k of a_i's equal to 1 and $n - k$ of them equal to 0, which completes the proof.

Remark 2.2 *There are three hyper parameters in the SAND layer, k, σ and α:*

- *k is the number of features to be selected. It will be initially set straightforwardly.*
- *σ indicates how firmly we would like to restrict the number of features to k. A higher value of $sigma$ places greater emphasis on precisely achieving k features, resulting in faster binarization (polarization toward 0 and 1) of the gains (a_i's).*
- *α indicates which norm to be used to normalize the gain vector $\underline{a}$ during training.*

We will see in the experiments that the method is not sensitive to the choice of σ and α. In fact, setting σ within the range of standard deviation of the input features, and $\alpha = 2$, yields nearly optimal results across all datasets. Thus, there is no need to fine-tune σ and α.

3 Experiments

Feature Selection for Neural Networks

We explored the performance of SAND through experiments on standard benchmark datasets used for feature selection in neural networks. Specifically, we utilized the same six standard datasets used in previous studies by [12, 1, 26]. Also, we normalize the datasets to have zero mean and unit standard deviation for each feature. We implemented a neural network with one hidden layer and a ReLU activation, while selecting $k = 60$ features. Given the variation in hidden layer widths across cited works, we opted for a width equal to $n/3$, where n represents the dimensionality of the input data. Please refer to Table 3 of Appendix A for a comprehensive overview of the six datasets, the corresponding number of epochs and batch size used for training, along with the mean accuracy of the model with all features.[2]

Our evaluation included a comparison between SAND and four established feature selection algorithms, namely Sequential Attention [26], Sequential LASSO [17], LLY [15], and Group LASSO [28]. For all methods except ours, training comprised a feature selection phase followed by a fitting phase wherein the neural network was retrained from scratch on the selected features. As for SAND, the fitting phase was omitted and the weights learned during the selection phase were directly utilized. In other words, the gains corresponding to the non-selected features where set to zero while keeping all other weights of the model intact. Hence, from this point of view, our method offers two key benefits. Firstly, it demands fewer epochs (33% fewer epochs in our experiments). Secondly, it provides a streamlined pipeline where both selection and inference are handled by the same model.

Across all experiments, we employed the Adam optimizer with a learning rate of 10^{-3}, and we partitioned the datasets into 70-10-20 splits for training, validation, and testing, respectively. For the

[2]The code to reproduce our experiments is available at `https://anonymous.4open.science/r/SAND-6BB1`

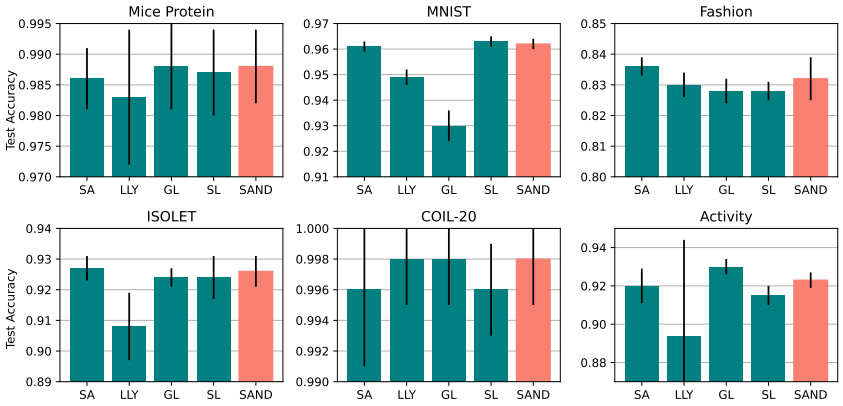


Figure 3: Test accuracies on 6 datasets after selecting 60 features over 10 trials. SA = Sequential Attention, GL = Group LASSO, and SL = Sequential LASSO. SAND method is applied with $\sigma = 1.5$.

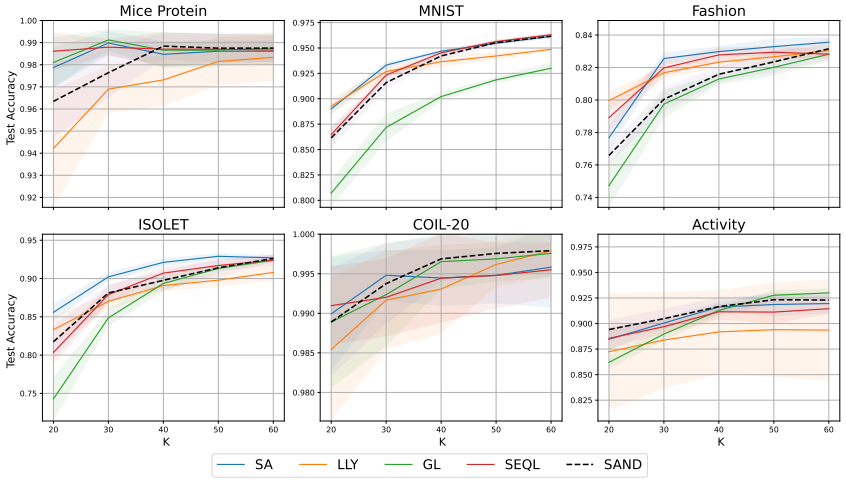


Figure 4: Test accuracies when selecting $k \in \{20, 30, 40, 50, 60\}$ features.

hyper paramaters of the SAND layer, we use $\sigma = 1.5$ and $\alpha = 2$ for all datasets. The comparative results are summarized in Figure 3, and the exact numerical values are shared in Table 4 of Appendix B. The error bars were computed using the standard deviation over 10 trials. Drawing from the results in Figure 3, we note that SAND competes effectively with other feature selection methods; it is either the best or the second best with the performance very close to the best one.

To provide additional insights, we varied the number of selected features $k \in \{20, 30, 40, 50\}$, using the same settings, and assessed performance on the test set. Results are shown in Figure 4. Notably, SAND demonstrates its strength in feature selection, showcasing results that outperform or are comparable to other methods across different feature counts. This advantage is particularly significant given the fact that the best method is changing from dataset to dataset. Thus, there is a need for algorithms that deliver value beyond marginal accuracy improvements, prioritizing enhancements in computational demand and simplicity—and this is precisely what our method accomplishes. Please notice that all of these experiments are done for fixed values of $\sigma = 1.5$ and $\alpha = 2$ without any fine-tuning.

Role of σ

As discussed in Remark 2.2, the parameter σ influences the rate of gain polarization. To demonstrate this, we trained the SAND model on the MICE dataset for 2000 epochs, using σ values of 1.0 and

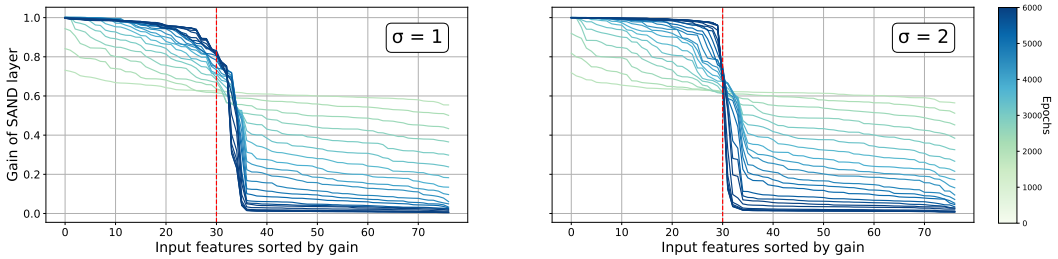


Figure 5: Polarization of the feature gains in SAND layer for $k = 30$.

Table 1: Test accuracies when selecting $k = 60$ features with SAND using different σ's

Dataset	$\sigma = 1.0$	$\sigma = 1.5$	$\sigma = 2.0$	$\sigma = 2.5$	$\sigma = 3.0$
Mice Protein	0.988 ± 0.006	0.988 ± 0.006	0.988 ± 0.007	0.988 ± 0.006	0.987 ± 0.006
MNIST	0.953 ± 0.007	0.962 ± 0.002	0.958 ± 0.002	0.953 ± 0.002	0.948 ± 0.004
MNIST-Fashion	0.830 ± 0.007	0.832 ± 0.007	0.833 ± 0.003	0.831 ± 0.004	0.825 ± 0.003
ISOLET	0.913 ± 0.009	0.926 ± 0.005	0.922 ± 0.007	0.921 ± 0.006	0.916 ± 0.006
COIL-20	0.991 ± 0.005	0.998 ± 0.003	0.998 ± 0.002	0.998 ± 0.003	0.996 ± 0.004
Activity	0.929 ± 0.006	0.923 ± 0.004	0.922 ± 0.006	0.924 ± 0.004	0.922 ± 0.005

2.0, while keeping other settings fixed. We recorded the gains every 10 epochs. Figure 5 presents the sorted gains for selected epochs.We can observe that while the gains tend to cluster around 1 and 0 in both plots, this clustering occurs at a higher rate for a larger σ. Moreover, to assess SAND's sensitivity to σ, we replicated the experiment conducted for selecting 60 features while varying $\sigma \in \{1.0, 1.5, 2.0, 2.5, 3.0\}$. Results are presented in Table 1. As evident from the table, our approach demonstrates insensitivity to the selection of σ, which underscores a positive aspect of the proposed method.

Effect of ℓ_α-Norm

To have an insight of the effect of α, we conducted a duplicate experiment, previously performed to select 60 features, this time employing $\alpha = 1.0$ (with $\sigma \in \{0.5, 1.5\}$). The outcomes are presented in Table 2. In accordance with the table, it can be noted that consistent performance was observed despite the variation in α.

4 Summary and Future Works

In this paper, we introduced a novel feature selection method. Specifically, we presented a new layer (SAND) that integrates into a neural network, enabling automatic feature selection during the training phase. The benefits of this approach include:

- On par with the state-of-the-are performance: Through extensive experiments, we showed that the proposed method has effectively state-of-the-art performance.

Table 2: Test accuracies when selecting $k = 60$ features with SAND for $\alpha \in \{1.0, 2.0\}$

Dataset	$\alpha = 1.0$		$\alpha = 2.0$
	$\sigma = 0.5$	$\sigma = 1.5$	
Mice Protein	0.987 ± 0.005	0.988 ± 0.007	0.988 ± 0.006
MNIST	0.959 ± 0.002	0.956 ± 0.002	0.962 ± 0.002
MNIST-Fashion	0.828 ± 0.007	0.830 ± 0.006	0.832 ± 0.007
ISOLET	0.922 ± 0.008	0.910 ± 0.010	0.926 ± 0.005
COIL-20	0.997 ± 0.003	0.996 ± 0.005	0.998 ± 0.003
Activity	0.922 ± 0.004	0.907 ± 0.008	0.923 ± 0.004

- Low computational and memory burden: The layer introduces only n trainable parameters, along with n multiplication-additions and a single n-dimensional ℓ_2-normalization, where n represents the number of features.
- One-shot feature selection and network training: There is no need for selecting the features in one phase of the training and then retrain the network with the selected features. Once the training phase has finished, the features are selected and the neural network is trained for the selected features.
- Control on the number of selected features: The number of features can be directly set in the algorithm in contrast to the main stream methods which require sweeping over a hyper parameter to be able to obtain the desired number of features.
- Considerably faster: As there is no need for the retraining phase, and due to the low computational overload, the method is considerably faster than the competitors.
- Handy Integration of Feature Selection in Neural Networks: Our feature selection method seamlessly integrates as an additional layer at the outset of the neural network, preserving the original architecture and loss function. With only input gradients required to train the layer gains, the network architecture or loss function can be treated as a black box.
- Tailored features to the application and the neural network architecture: Since SAND layer is an integral component of the base model, the features selected are automatically adapted for the specific application at hand and the chosen model architecture.
- Remarkably simple both conceptually and practically: the mathematical model of our method involves only entrywise multiplication, addition with Gaussian noise, followed by ℓ_2-norm normalization, rendering it remarkably simple in theory and in practice.

It is worth mentioning that the proposed SAND layer works in a very similar way to the Dropout layer but with an opposing effect. In the Dropout layer, randomization leads to an even distribution of information across all neurons. Conversely, randomization in the SAND layer, due to weights' constraint, selects only neurons with the highest information content.

A straightforward continuation of this work is to explore SAND layer's performance for network pruning by incorporating it into intermediate layers, akin to how Dropout and Batch Normalization layers are utilized. Additionally, studying the effect of different noise distributions and rigorous understanding of the effect of α and σ for different network architectures (dense, convolutional, transformers, etc.) are interesting lines for future researches. Furthermore, considering feature's relation structure during the selection is another valuable avenue to explore.

References

[1] Muhammed Fatih Balın, Abubakar Abid, and James Zou. "Concrete autoencoders: Differentiable feature selection and reconstruction". In: *International conference on machine learning*. PMLR. 2019, pp. 444–453.

[2] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32.

[3] Brais Cancela, Verónica Bolón-Canedo, and Amparo Alonso-Betanzos. "E2E-FS: An End-to-End Feature Selection Method for Neural Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.7 (2023), pp. 8311–8323.

[4] Brais Cancela et al. "A scalable saliency-based feature selection method with instance-level information". In: *Knowledge-Based Systems* 192 (2020), p. 105326.

[5] Girish Chandrashekar and Ferat Sahin. "A survey on feature selection methods". In: *Computers Electrical Engineering* 40.1 (2014), pp. 16–28.

[6] Abhimanyu Das and David Kempe. "Submodular meets Spectral: Greedy Algorithms for Subset Selection, Sparse Approximation and Dictionary Selection". In: *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*. 2011, pp. 1057–1064.

[7] Tansel Dokeroglu, Ayça Deniz, and Hakan Ezgi Kiziloz. "A comprehensive survey on recent metaheuristics for feature selection". In: *Neurocomputing* 494 (2022), pp. 269–296.

[8] Isabelle Guyon and André Elisseeff. "An introduction to variable and feature selection". In: *J. Mach. Learn. Res.* 3.null (Mar. 2003), pp. 1157–1182. ISSN: 1532-4435.

[9] Isabelle Guyon et al. "Gene Selection for Cancer Classification using Support Vector Machines". In: *Mach. Learn.* 46.1–3 (Mar. 2002), pp. 389–422. ISSN: 0885-6125.

[10] Xiaofei He, Deng Cai, and Partha Niyogi. "Laplacian Score for Feature Selection". In: *Advances in Neural Information Processing Systems*. Vol. 18. MIT Press, 2005.

[11] Ron Kohavi and George H. John. "Wrappers for Feature Subset Selection". In: *Artif. Intell.* 97 (1997), pp. 273–324.

[12] Ismael Lemhadri, Feng Ruan, and Rob Tibshirani. "Lassonet: Neural Networks with Feature Sparsity". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 10–18.

[13] Jundong Li et al. "Feature Selection: A Data Perspective". In: *ACM Comput. Surv.* 50.6 (Dec. 2017). ISSN: 0360-0300.

[14] Yifeng Li, Chih-Yu Chen, and Wyeth W. Wasserman. "Deep Feature Selection: Theory and Application to Identify Enhancers and Promoters". In: *Journal of Computational Biology* 23.5 (2016), pp. 322–336.

[15] Yiwen Liao, Raphael Latty, and Bin Yang. "Feature Selection Using Batch-wise Attenuation and Feature Mask Normalization". In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2021, pp. 1–9.

[16] Scott M. Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. 2017, pp. 4768–4777.

[17] Shan Luo and Zehua Chen. "Sequential Lasso cum EBIC for feature selection with ultra-high dimensional feature space". In: *Journal of the American Statistical Association* 109.507 (2014), pp. 1229–1240.

[18] Simone Scardapane et al. "Group sparse regularization for deep neural networks". In: *Neurocomputing* 241 (2017), pp. 81–89.

[19] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". In: *Workshop at International Conference on Learning Representations (ICLR)*. 2014.

[20] Suraj Srinivas, Akshayvarun Subramanya, and R. Venkatesh Babu. "Training Sparse Neural Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2017, pp. 455–462.

[21] Robert Tibshirani. "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), pp. 267–288.

[22] Qian Wang et al. "Attentional Neural Network: Feature Selection Using Cognitive Feedback". In: *Neural Information Processing Systems*. 2014.

[23] Shiping Wang et al. “Subspace learning for unsupervised feature selection via matrix factorization”. In: *Pattern Recognition* 48.1 (2015), pp. 10–19.

[24] Yutaro Yamada et al. “Feature Selection using Stochastic Gates”. In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 10648–10659.

[25] Yiming Yang and Jan O. Pedersen. “A Comparative Study on Feature Selection in Text Categorization”. In: *Proceedings of the Fourteenth International Conference on Machine Learning*. ICML ’97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 412–420.

[26] Taisuke Yasuda et al. “Sequential Attention for Feature Selection”. In: *Proceedings of the 11th International Conference on Learning Representations*. 2023.

[27] Sepehr Abbasi Zadeh et al. “Scalable Feature Selection via Distributed Diversity Maximization”. In: 2017, pp. 2876–2883.

[28] Lei Zhao, Qinghua Hu, and Wenwu Wang. “Heterogeneous Feature Selection with Multimodal Deep Neural Networks and Sparse Group Lasso”. In: *IEEE Transactions on Multimedia* 17.11 (2015), pp. 1936–1948.

A Experimental Set-up

We provide a table containing details about all datasets utilized in the feature selection experiments. Additionally, the table includes the epochs employed during training for each dataset, identifying the ones used for feature selection and the ones used to retrain/fit the model on the selected features. As indicated in the experiments (Section 3), fitting epochs are only utilized by models other than SAND, whereas SAND employs only the 'Select Epochs'. The table also presents the test accuracy of the base model trained using all features.

Table 3: Dataset Characteristics, Experiment Parameters, and All-Features Accuracy[3]

Dataset	(n, d)	# Classes	Select Epochs	Fit Epochs	Batch Size	All Features
Mice Protein	(1,080, 77)	8	400	200	64	0.987 ± 0.006
MNIST	(70,000, 784)	10	100	50	64	0.978 ± 0.001
MNIST-Fashion	(70,000, 784)	10	200	100	64	0.878 ± 0.003
ISOLET	(7,797, 617)	26	400	200	64	0.958 ± 0.002
COIL-20	(1,440, 400)	20	1000	500	64	0.996 ± 0.003
Activity	(10,299, 561)	6	200	100	64	0.941 ± 0.002

Moreover, the experiments were executed on a machine equipped with an NVIDIA GeForce RTX 4090 GPU with 24GB of RAM, paired with an AMD Ryzen 9 5900X 12-Core Processor featuring 24 threads.

B Experimental Results

We present the benchmarking results of SAND alongside other methods on the six datasets discussed in the experiments (Section 3). The intervals were calculated using the standard deviation across 10 trials.

Table 4: Test accuracies over 10 trials: mean $\pm$ standard deviation

Dataset	SA	LLY	GL	SL	SAND
Mice Protein	0.986 ± 0.005	0.983 ± 0.011	0.988 ± 0.007	0.987 ± 0.007	0.988 ± 0.006
MNIST	0.961 ± 0.002	0.949 ± 0.003	0.930 ± 0.006	0.963 ± 0.002	0.962 ± 0.002
MNIST-Fashion	0.836 ± 0.003	0.830 ± 0.004	0.828 ± 0.004	0.828 ± 0.003	0.832 ± 0.007
ISOLET	0.927 ± 0.004	0.908 ± 0.011	0.924 ± 0.003	0.924 ± 0.007	0.926 ± 0.005
COIL-20	0.996 ± 0.005	0.998 ± 0.003	0.998 ± 0.003	0.996 ± 0.003	0.998 ± 0.003
Activity	0.920 ± 0.009	0.894 ± 0.050	0.930 ± 0.004	0.915 ± 0.005	0.923 ± 0.004

[3]Our study utilizes the entire MNIST and Fashion datasets, unlike related works. Additionally, the Activity dataset sourced from [12]'s Google Drive and [26]'s repository contains 10,299 samples, as opposed to the 5,744 samples reported in the referenced papers.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and intro shed light on the novelty of the solution and the numerical as well as theoretical analysis done in the paper. Furthermore, due to the simplicity of the solution that does not compromise the performance, which is the main selling point of the paper, we provide the exact mathematical formulation in the abstract (to give the reader a sense of how simple the solution is, to the extent it can be explained at the high-level in the abstract).

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [NA]

   Justification: To the best of our knowledge, we did not identify any limitation of our method. The solution is application-driven and it first originated from a real-life feature selection problem in multispectral imaging. After it showed remarkable success, it was then formalised and benchmarked against state-of-the-art solutions and datasets.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Refer to *Linear Regression* 2

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The method's architecture is detailed in Section 2 and a link to the code to reproduce the experiments is provided in Section 3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: A link to the GitHub repository hosting the code and the data is provided in the Section 3.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Experimental set-up and a link to the code is provided in Section 3 and Appendix A

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Refer to Section 3

   Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Although the topic of the paper is not hardware dependent, we provided the specifications of the machine used in Appendix A. The total compute time for all the reported results is less than a day.

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The contribution is mainly conceptual. Numerical benchmarking was done on public datasets commonly used for feature selection problem in all recent literature.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

Justification: Our method aids in the task of feature selection and does not have direct societal impacts.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our method does not pose such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: This work is licensed under the CC-BY-NC 4.0 license.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Documentation and courtesy provided in the anonymized github repo.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.