SAND: One-Shot Feature Selection with Additive Noise Distortion

Pedram Pad [1] **Hadi Hammoud** [1 2] **Mohamad Dia** [1] **Nadim Maamari** [1] **L. Andrea Dunbar** [1 2]

Abstract

Feature selection is a critical step in data-driven applications, reducing input dimensionality to enhance learning accuracy, computational efficiency, and interpretability. Existing state-of-the-art methods often require post-selection retraining and extensive hyperparameter tuning, complicating their adoption. We introduce a novel, non-intrusive feature selection layer that, given a target feature count k, automatically identifies and selects the k most informative features during neural network training. Our method is uniquely simple, requiring no alterations to the loss function, network architecture, or post-selection retraining. The layer is mathematically elegant and can be fully described by:

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

where x_i is the input feature, $\tilde{x}_i$ the output, z_i a Gaussian noise, and a_i trainable gain such that $\sum_i a_i^2 = k$. This formulation induces an automatic clustering effect, driving k of the a_i gains to 1 (selecting informative features) and the rest to 0 (discarding redundant ones) via weighted noise distortion and gain normalization. Despite its extreme simplicity, our method achieves competitive performance on standard benchmark datasets and a novel real-world dataset, often matching or exceeding existing approaches without requiring hyperparameter search for k or retraining. Theoretical analysis in the context of linear regression further validates its efficacy. Our work demonstrates that simplicity and performance are not mutually exclusive, offering a powerful yet straightforward tool for feature selection in machine learning.

[1]CSEM, Neuchâtel, Switzerland [2]EPFL, Lausanne, Switzerland. Correspondence to: Pedram Pad <pedram.pad@csem.ch>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

1. Introduction

Feature selection is a fundamental problem in high-dimensional statistics and machine learning (Guyon & Elisseeff, 2003; Li et al., 2017). 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 (Guyon et al., 2002). 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 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 (He et al., 2005), matrix factorization (Wang et al., 2015), and the use of autoencoder neural networks (Balın et al., 2019). These methods are 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 these 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

(Yang & Pedersen, 1997; Chandrashekar & Sahin, 2014). 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 (Kohavi & John, 1997) involve a search process guided by the final performance of a learning model, such as classifier accuracy. Examples include greedy sequential feature selection (Das & Kempe, 2011), SHAP (SHapley Additive exPlanations) values calculation (Lundberg & Lee, 2017), in addition to combinatorial optimization and metaheuristic search algorithms (Zadeh et al., 2017; Dokeroglu et al., 2022). 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 (Breiman, 2001), Recursive Feature Elimination for Support Vector Machine (RFE-SVM) (Guyon et al., 2002), sparsity-promoting models (Tibshirani, 1996), and deep learning techniques (Simonyan et al., 2014; Wang et al., 2014). 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 (Luo & Chen, 2014; Zhao et al., 2015; Li et al., 2016; Lemhadri et al., 2021; Cancela et al., 2023), the addition of stochastic gates (Srinivas et al., 2017; Borisov et al., 2019; Yamada et al., 2020), the use of attention mechanisms (Liao et al., 2021; Yasuda et al., 2023), and the application of saliency maps (Cancela et al., 2020) to solve the feature selection problem on non-linear models. For instance, Sequential LASSO (Luo & Chen, 2014) provides an efficient implementation of greedy LASSO to recursively select input features, while Group LASSO (Zhao et al., 2015; Scardapane et al., 2017) further modifies the objective function to encourage sparsity at the group level. In LassoNet (Lemhadri et al., 2021), 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, (Yamada et al., 2020) 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. While (Yamada et al., 2020) employs a similar stochastic approach as ours, it modifies the loss function by adding an additional term and lacks direct control over the number of selected features, requiring a grid search over a model hyperparameter. In contrast, our stochastic framework does not alter or rely on the loss function, eliminating the need for retraining or hyperparameter tuning. Additionally, our method allows direct specification of the desired number of features as an input. 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 (Yasuda et al., 2023).

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 (Yamada et al., 2020; Lemhadri et al., 2021; Yasuda et al., 2023). 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 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 and a novel real-world dataset, showcasing our method's effective competition against existing approaches

while highlighting its practicality and application-driven nature. 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 $\underline{\boldsymbol{x}} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n)^\top$ (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 $\underline{\boldsymbol{y}}$[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 $\underline{a} = (a_1, a_2, \dots, a_n)^\top$ 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:

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

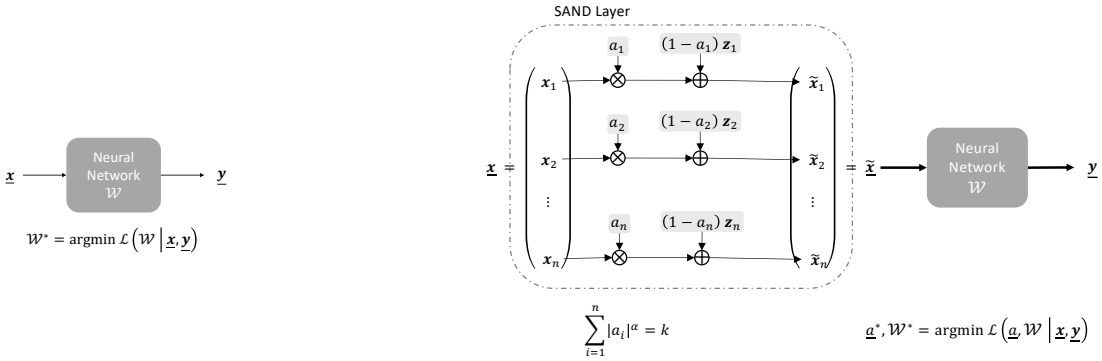
Notice that if there is no noise $\underline{\boldsymbol{z}}$ (i.e., $\sigma = 0$), the a_i's would be absorbed in the weights of the first layer of the neural network. Additionally, if $k = n$, all a_i's can become 1 and then $\tilde{\underline{\boldsymbol{x}}}$ will be identical to $\underline{\boldsymbol{x}}$ 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., $\underline{a} = (a_1, a_2, \dots, a_n)^\top$ that have

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

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 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

[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



(a) Vanilla Neural Network

(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.

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, effectively removing those features. Notably, features with small gains are noisy and hence inherently ignored by other parts of the neural network.

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

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

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

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

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 (9), we obtain

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

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

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

which leads to

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

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

$$\begin{aligned}
\sigma^2 \sum_{j=1}^{n} \overline{w}_j^2 \left(\frac{1}{a_j} - 1\right)^2 &= \frac{\lambda}{2}\alpha \sum_{j=1}^{n} |a_j|^\alpha \left(1 - a_j\right)\\
&= \frac{\lambda}{2}\alpha \left(k - \sum_{j=1}^{n} a_j\, |a_j|^\alpha\right) \qquad (13)
\end{aligned}$$

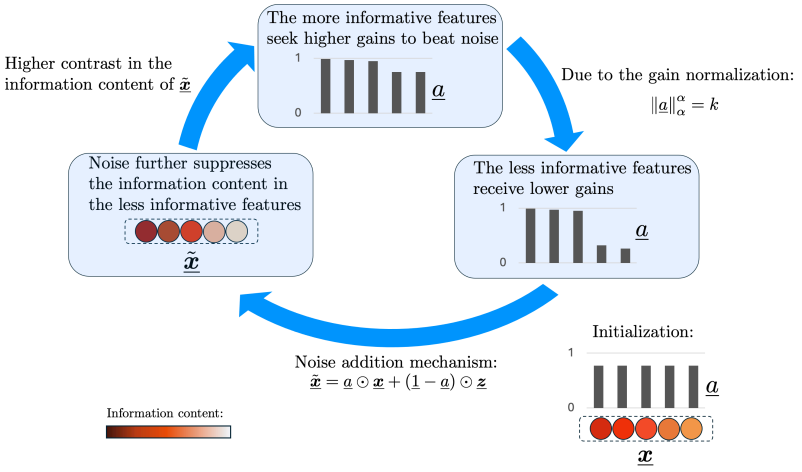


Figure 2. Reinforcing loop that results in polarization of the gains with $n = 5$ and $k = 3$.

Combining (13) and (10), we obtain

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

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. Moreover, to ensure stable training, we apply clipping to the a_i values throughout the training process, keeping them within the range $[0, 1]$.

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 nine datasets, seven of which were used in previous studies by (Balın et al., 2019; Lemhadri et al., 2021; Yamada et al., 2020; Yasuda et al., 2023). The additional real and synthetic datasets were California Housing (Torgo, 1997) and HAR70 (Logacjov & Ustad, 2023). California (CA) Housing is a real dataset consisting of 20640 samples with 8 interpretable features, with the task of regressing the price of the house with the least features. Additionally, HAR70 is a large synthetic dataset comprising 2.3 million samples. It includes 6 informative features, which we augment with 100 nuisance features sampled from $\mathcal{N}(0, 0.1)$, and the task involves classifying the activity being performed. The testing metric for all datasets was accuracy, except for the CA Housing dataset, where the mean absolute error (MAE) was used. The datasets were normalized to have zero mean and unit standard deviation for each feature. Furthermore, we evaluated SAND on a novel multi-spectral image dataset, introduced in this paper for the first time, which has never been used before for feature selection. Details of this dataset and our evaluation are provided in Appendix C.

We implemented a neural network with one hidden layer and a ReLU activation. 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 nine datasets, the corresponding number of

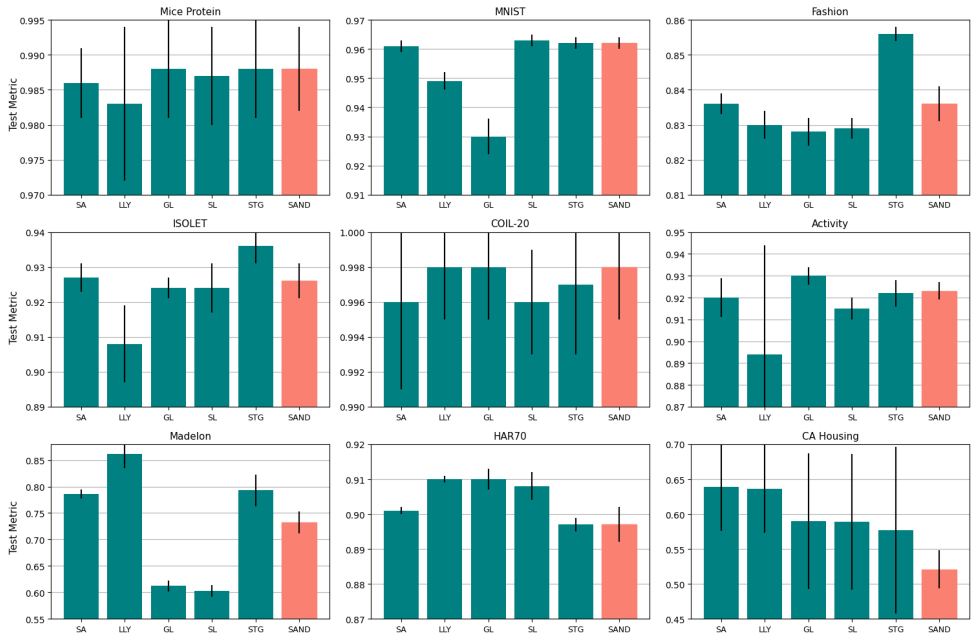


Figure 3. Test metrics on 9 datasets over 10 trials. The metric is accuracy (↑) for all except MAE (↓) for CA Housing being a regression problem. SA = Sequential Attention, LLY = Batch-wise Attenuation, GL = Group LASSO, SL = Sequential LASSO, and STG = Stochastic Gates.

epochs and batch size used for training, along with the mean accuracy/absolute-error of the model with all features.[2]

Our evaluation included a comparison between SAND and five established feature selection algorithms, namely Sequential Attention (SA) (Yasuda et al., 2023), Sequential LASSO (SL) (Luo & Chen, 2014), Btach-wise Attenuation (LLY) (Liao et al., 2021), Group LASSO (GL) (Zhao et al., 2015), and Stochastic Gates (STG) (Yamada et al., 2020).[3] For all methods except ours, training comprised a feature selection phase followed by a fitting phase wherein the neural network was retrained on the selected features. Although STG often eliminates the need for retraining thanks to its weight polarization, on some datasets (e.g. MICE Protein) the weights fail to polarize when $k = 60$, even after an exhaustive search over the regularization parameter which indirectly governs the number of selected features, and so we resort to retraining in some cases. As for SAND, the fitting phase was omitted and the weights learned during the selection phase were directly utilized for inference. 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 hyperparameters of the SAND layer, we used $\sigma = 1.5$ and $\alpha = 2$ consistently. Unless otherwise specified, we selected $k = 60$ features for all datasets by default, except for the following: $k = 5$ for the Madelon dataset, $k = 3$ for the CA Housing dataset, and $k = 6$ for the Har70 dataset. 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 calculated using the standard deviation over 10 trials. Drawing from the results in Figure 3, SAND competes effectively with other feature selection methods, while offering a streamlined pipeline that requires fewer iterations. It also exhibits a very consistent behavior as shown in Table 5 of Appendix B.

To provide additional insights, we varied the number of selected features k under consistent settings and evaluated 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

[2]The code to reproduce our experiments is available at https://github.com/csem/SAND

[3]These algorithms represent the top-performing methods reported in the literature. A comparison with other approaches (Atashgahi et al., 2022; Lemhadri et al., 2021; Sokar et al., 2022) demonstrates that SAND achieves better performance.

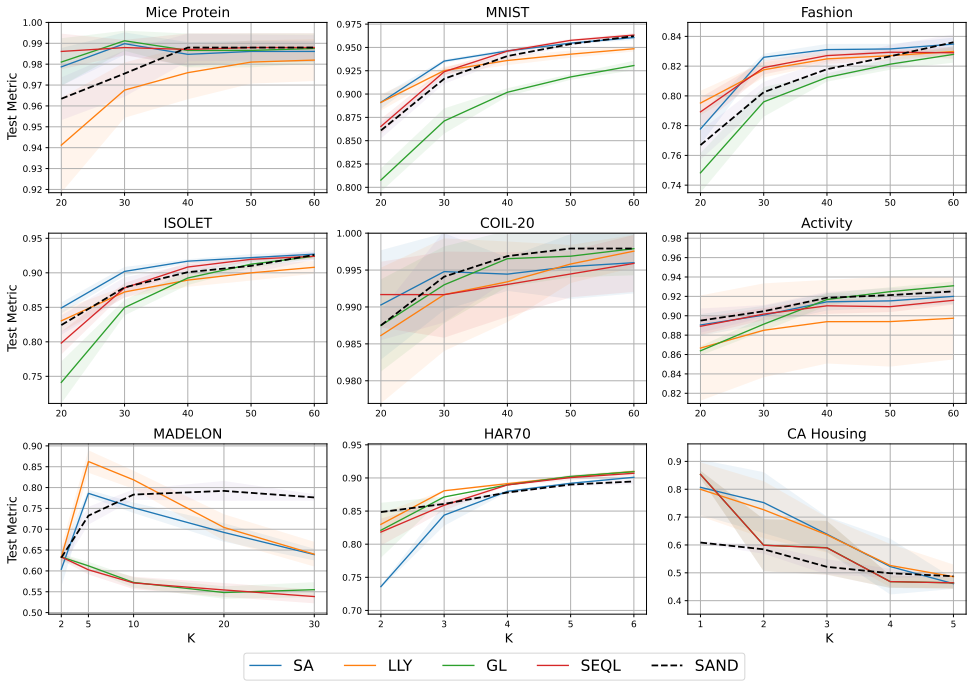


Figure 4. Test metrics for different k's. Accuracy ($\uparrow$) for all except MAE ($\downarrow$) for CA Housing.

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—qualities that our method exemplifies. It is important to note that all SAND experiments were conducted using fixed values of $\sigma = 1.5$ and $\alpha = 2$, without any fine-tuning. We omitted STG from Figure 4 due to the high computational cost of performing an exhaustive search over the regularization parameter λ for each k. Moreover, on the MICE Protein dataset (77 features), STG fails to identify a clear feature subset once k exceeds 37. For $k \leq 37$, STG produces a polarized solution—exactly k features attain importance weight at almost 1, while the remainder are almost zero. However, for $k > 37$, the selection process stalls: irrespective of λ, only the same 37 features remain at weight 1, and the other 40 features converge to an almost uniform, nonzero weight. Adjusting λ cannot induce the selection of additional features; it only uniformly increases the weights of these 40 unselected features above zero.

Role of σ

As discussed in Remark 2.2, the parameter σ influences the rate of gain polarization. To demonstrate this, we trained SAND model on MICE dataset for 2000 epochs, using σ values of 1.0 and 2.0, while keeping other settings fixed. We recorded the gains every 10 epochs. Figure 5 presents the sorted gains for selected epochs.We 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 initial experiment while varying $\sigma \in \{1.0, 1.5, 2.0, 2.5, 3.0\}$. Results are presented in Table 1. As evident in the table, our approach demonstrates high robustness to the selection of σ, which underscores a positive aspect of the proposed method.

Effect of the choice of α in ℓ_α-Normalization

To have an insight of the effect of α, we conducted a duplicate experiment, this time employing $\alpha = 1.0$ (with $\sigma \in \{0.5, 1.5\}$). The outcomes are presented in Table 2. As shown in the table, the performance remains highly consistent, despite variations 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-art performance: Through extensive experiments, we showed that the proposed method has effectively state-of-the-art performance.

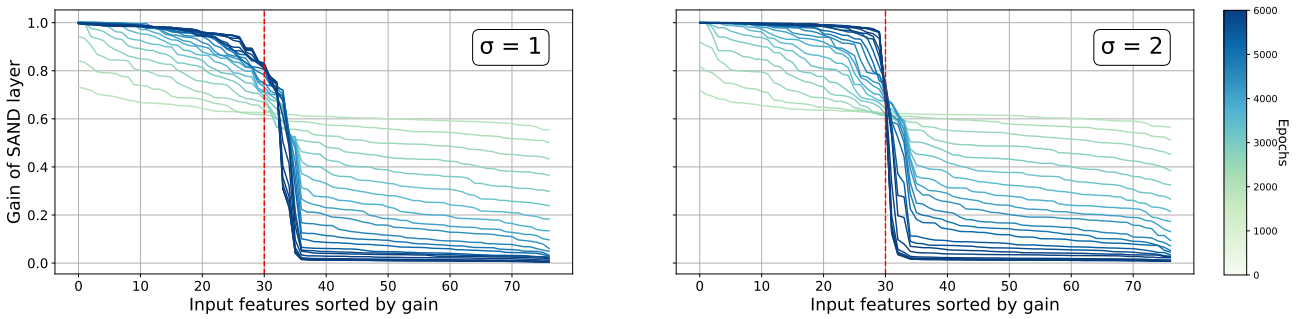


Figure 5. Polarization of the feature gains in SAND layer for $k = 30$.

Table 1. Test metrics for feature selection 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
Madelon	0.741 ± 0.022	0.732 ± 0.021	0.708 ± 0.023	0.689 ± 0.017	0.646 ± 0.025
HAR70	0.901 ± 0.002	0.897 ± 0.005	0.889 ± 0.002	0.881 ± 0.005	0.873 ± 0.006
CA Housing	0.514 ± 0.026	0.521 ± 0.027	0.532 ± 0.030	0.557 ± 0.022	0.566 ± 0.007

- 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 is 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 SAND, 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 features relation structure during the selection is another valuable avenue to explore.

Table 2. Effect of α on SAND layer.

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
Madelon	0.756 ± 0.059	0.675 ± 0.029	0.732 ± 0.021
HAR70	0.854 ± 0.036	0.814 ± 0.038	0.897 ± 0.005
CA Housing	0.538 ± 0.063	0.560 ± 0.023	0.521 ± 0.027

Acknowledgements

The present work was developed within the AGRARSENSE Project that has received Chips JU funding (Grant Agreement No. 101095835). It has received funding from the Swiss State Secretariat for Education, Research and lnnovation (SERI) and is co-funded by the Innosuisse – Swiss Innovation Agency. More information: info@agrarsense.eu.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Atashgahi, Z., Sokar, G., van der Lee, T., Mocanu, E., Mocanu, D., Veldhuis, R., and Pechenizkiy, M. Quick and robust feature selection: the strength of energy-efficient sparse training for autoencoders. *Machine Learning*, 111: 377–414, 2022.

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

Borisov, V., Haug, J., and Kasneci, G. Cancelout: A layer for feature selection in deep neural networks. In *28th International Conference on Artificial Neural Networks,*, 2019.

Breiman, L. Random forests. *Machine Learning*, 45(1): 5–32, 2001.

Cancela, B., Bolón-Canedo, V., Alonso-Betanzos, A., and Gama, J. A scalable saliency-based feature selection method with instance-level information. *Knowledge-Based Systems*, 192:105326, 2020.

Cancela, B., Bolón-Canedo, V., and Alonso-Betanzos, A. E2e-fs: An end-to-end feature selection method for neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(7):8311–8323, 2023.

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

Das, A. and Kempe, D. 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*, pp. 1057–1064, 2011.

Dokeroglu, T., Deniz, A., and Kiziloz, H. E. A comprehensive survey on recent metaheuristics for feature selection. *Neurocomputing*, 494:269–296, 2022.

Dunbar, A. L., Blanc, S., Pad, P., Beuchat, P.-A., Papathanasiou, A., Nikitakis, A., and Makantasis, K. A fast simple-to-use and inexpensive multispectral camera to detect skin conditions (Conference Presentation). In *Photonic Instrumentation Engineering VII*, volume 11287. SPIE, 2020.

Dupont, M. F., Elbourne, A., Cozzolino, D., Chapman, J., Truong, V. K., Crawford, R. J., and Latham, K. Chemometrics for environmental monitoring: a review. *Analytical Methods*, 12(38):4597–4620, 2020.

ElMasry, G., Mandour, N., Al-Rejaie, S., Belin, E., and Rousseau, D. Recent applications of multispectral imaging in seed phenotyping and quality monitoring—an overview. *Sensors*, 19(5), 2019.

Guyon, I. and Elisseeff, A. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3(null): 1157–1182, mar 2003. ISSN 1532-4435.

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. Gene selection for cancer classification using support vector machines. *Mach. Learn.*, 46(1–3):389–422, mar 2002. ISSN 0885-6125.

He, X., Cai, D., and Niyogi, P. Laplacian score for feature selection. In *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005.

Kohavi, R. and John, G. H. Wrappers for feature subset selection. *Artif. Intell.*, 97:273–324, 1997.

Lemhadri, I., Ruan, F., and Tibshirani, R. Lassonet: Neural networks with feature sparsity. In *International Conference on Artificial Intelligence and Statistics*, pp. 10–18. PMLR, 2021.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., and Liu, H. Feature selection: A data perspective. *ACM Comput. Surv.*, 50(6), dec 2017. ISSN 0360-0300.

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

Liao, Y., Latty, R., and Yang, B. Feature selection using batch-wise attenuation and feature mask normalization. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9. IEEE, 2021.

Logacjov, A. and Ustad, A. Har70+ [dataset]. UCI Machine Learning Repository, 2023. URL https://doi.org/10.24432/C5CW3D. Accessed: 2023-10-10.

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

Luo, S. and Chen, Z. Sequential Lasso cum EBIC for feature selection with ultra-high dimensional feature space. *Journal of the American Statistical Association*, 109(507): 1229–1240, 2014.

Ortega, S., Halicek, M., Fabelo, H., Callico, G. M., and Fei, B. Hyperspectral and multispectral imaging in digital and computational pathology: a systematic review. *Biomed. Opt. Express*, 11(6):3195–3233, Jun 2020.

Scardapane, S., Comminiello, D., Hussain, A., and Uncini, A. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at International Conference on Learning Representations (ICLR)*, 2014.

Sokar, G., Atashgahi, Z., Pechenizkiy, M., and Mocanu, D. C. Where to pay attention in sparse training for feature selection? In *Advances in Neural Information Processing Systems*, 2022.

Spigulis, J. *Ultra-Narrowband Multispectral Imaging: Techniques and Applications*. CRC Press, 2024.

Srinivas, S., Subramanya, A., and Babu, R. V. Training sparse neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 455–462, 2017.

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

Torgo, L. California housing dataset, 1997. URL https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html. Accessed: 2023-10-10.

Wang, Q., Zhang, J., Song, S., and Zhang, Z. Attentional neural network: Feature selection using cognitive feedback. In *Neural Information Processing Systems*, 2014.

Wang, S., Pedrycz, W., Zhu, Q., and Zhu, W. Subspace learning for unsupervised feature selection via matrix factorization. *Pattern Recognition*, 48(1):10–19, 2015.

Yamada, Y., Lindenbaum, O., Negahban, S., and Kluger, Y. Feature selection using stochastic gates. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10648–10659. PMLR, 13–18 Jul 2020.

Yang, Y. and Pedersen, J. O. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pp. 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

Yasuda, T., Bateni, M., Chen, L., Fahrbach, M., Fu, G., and Mirrokni, V. Sequential attention for feature selection. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

Zadeh, S., Ghadiri, M., Mirrokni, V., and Zadimoghaddam, M. Scalable feature selection via distributed diversity maximization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Zhao, L., Hu, Q., and Wang, W. Heterogeneous feature selection with multi-modal deep neural networks and sparse group lasso. *IEEE Transactions on Multimedia*, 17(11):1936–1948, 2015.

A. Experimental set-up

We provide Table 3 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 performance metrics.[5]

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
Madelon	(2,600, 500)	2	500	250	64	0.575 ± 0.017
HAR70	(2,259,597, 106)	8	6	3	64	0.890 ± 0.002
CA Housing	(20,640, 8)	N/A	200	100	64	0.440 ± 0.011

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. The code to reproduce our experiments is available at `https://github.com/csem/SAND`.

B. Experimental results

We present in Table 4 the benchmarking results of SAND alongside other methods on the nine datasets discussed in the experiments (Section 3). The intervals were calculated using the standard deviation across 10 trials.

Table 4. Test metrics over 10 trials (mean $\pm$ standard deviation): Accuracy ($\uparrow$) for all except MAE ($\downarrow$) for CA Housing.

Dataset	SA	LLY	GL	SL	STG	SAND
Mice Protein	0.986 ± 0.005	0.983 ± 0.011	0.988 ± 0.007	0.987 ± 0.007	0.988 ± 0.007	$\mathbf{0.988} \pm 0.006$
MNIST	0.961 ± 0.002	0.949 ± 0.003	0.930 ± 0.006	$\mathbf{0.963} \pm 0.002$	0.962 ± 0.002	0.962 ± 0.002
MNIST-Fashion	0.836 ± 0.003	0.830 ± 0.004	0.828 ± 0.004	0.829 ± 0.003	$\mathbf{0.856} \pm 0.002$	0.836 ± 0.005
ISOLET	0.927 ± 0.004	0.908 ± 0.011	0.924 ± 0.003	0.924 ± 0.007	$\mathbf{0.936} \pm 0.005$	0.926 ± 0.005
COIL-20	0.996 ± 0.005	0.998 ± 0.003	0.998 ± 0.003	0.996 ± 0.003	0.997 ± 0.004	$\mathbf{0.998} \pm 0.003$
Activity	0.920 ± 0.009	0.894 ± 0.050	$\mathbf{0.930} \pm 0.004$	0.915 ± 0.005	0.922 ± 0.006	0.923 ± 0.004
Madelon	0.786 ± 0.008	$\mathbf{0.862} \pm 0.027$	0.612 ± 0.010	0.603 ± 0.011	0.793 ± 0.03	0.732 ± 0.021
HAR70	0.901 ± 0.001	$\mathbf{0.910} \pm 0.001$	0.910 ± 0.003	0.908 ± 0.004	0.897 ± 0.002	0.897 ± 0.005
CA Housing	0.639 ± 0.063	0.636 ± 0.063	0.590 ± 0.097	0.589 ± 0.097	0.577 ± 0.119	$\mathbf{0.521} \pm 0.027$

Table 5 presents the consistency analysis for the HAR70 dataset, which comprises 6 informative features augmented with 100 synthetically generated noisy ones. This dataset was specifically chosen because SAND demonstrates its lowest relative performance here compared to other methods, as shown in Table 4 (with SAND's classification accuracy being 0.013% lower than the best-performing method, LLY). Nevertheless, SAND exhibits high consistency, consistently selecting 5 out of the 6 informative features and misidentifying the remaining feature only 20% of the time. Compared to SA and SL, SAND is more reliable in selecting the informative features. However, its slightly lower performance can be attributed to the inherent randomness of the experiments and the fact that other methods benefit from retraining after feature selection, unlike SAND, which operates without this additional training phase.

[5]The metric is classification accuracy ($\uparrow$) for all except MAE ($\downarrow$) for CA Housing. Our study utilizes the entire MNIST and Fashion datasets, unlike related works. Additionally, the Activity dataset sourced from (Lemhadri et al., 2021)'s Google Drive and (Yasuda et al., 2023)'s repository contains 10,299 samples, as opposed to the 5,744 samples reported in the referenced papers.

Table 5. Selection consistency on HAR70: feature selection frequency of the useful features $(0-5)$ and the misselected ones over 10 runs for $k = 6$.

Feature	SA	SL	**SAND**	STG	GL	LLY
0	10	10	10	10	10	10
1	8	9	8	8	9	10
2	10	10	10	10	10	10
3	10	10	10	10	10	10
4	10	7	10	10	10	10
5	2	10	10	10	10	10
Misselected	10	4	2	2	1	0

C. Benchmarking on real-world multi-spectral image dataset

In this section, we introduce a novel real-world multi-spectral image dataset, where feature selection is crucial for developing efficient and cost-effective acquisition hardware. We apply SAND to this dataset and demonstrate that it outperforms state-of-the-art methods.

Multi-spectral imaging is a powerful technology that captures images across multiple wavelengths, unlike conventional RGB imaging, which may fail to reveal critical spectral information. This approach has shown significant promise in various applications, including cancer detection in medical imaging (Ortega et al., 2020), precision agriculture (ElMasry et al., 2019), and chemometrics (Dupont et al., 2020). There are multiple ways to acquire multi-spectral data, including snapshot cameras, push-broom scanners, and filter wheel systems (Spigulis, 2024). However, a more cost-effective and flexible approach involves actively controlling the illumination. This method uses a monochromatic camera while sequentially activating lights at different wavelengths, enabling multi-spectral acquisition without requiring specialized optical components (Dunbar et al., 2020).

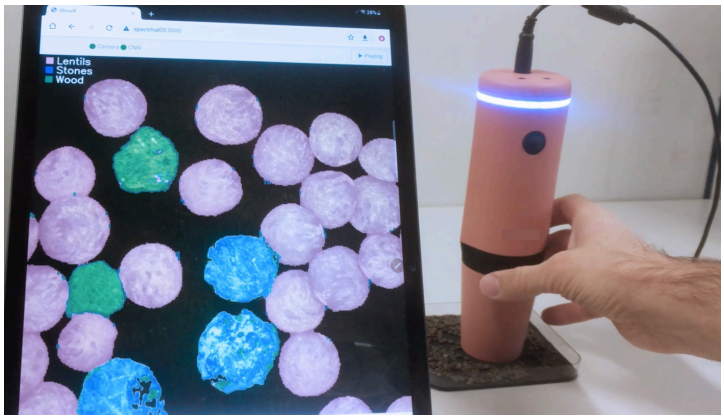


Figure 6. Demo of real-time segmentation on the MSI Grain dataset using multi-spectral imaging.

The "MSI Grain" dataset was acquired using active LED light control across 15 narrow spectral bands ranging from 360 nm to 940 nm. It contains spectral data for four distinct classes: Lentils, Wood, Stones, and Background. The goal is to leverage multi-spectral imaging to classify these visually similar classes, which are difficult to distinguish with the naked eye or conventional imaging. The dataset consists of 50,000 spectra per class, extracted from several multi-spectral image cubes (see Figure 6). Each spectrum corresponds to a 15-dimensional pixel within the image, representing its spectral signature

across the measured wavelengths. The dataset intentionally includes only spectral information, excluding spatial context. This design choice ensures that classification relies solely on the material's spectral response rather than its shape. The dataset is available in the code repository at https://github.com/csem/SAND.

A key challenge in multi-spectral imaging is reducing the number of spectral bands, or features, to optimize hardware efficiency. Fewer bands help minimize heat dissipation, lower hardware costs for commercial adoption, and enable fast real-time image acquisition. As a result, feature selection is not just a theoretical challenge but a practical necessity in this domain. SAND was developed to address this real-world need by jointly optimizing machine learning-based spectral detection and hardware design. Conventional feature selection methods often fall short in terms of performance and consistency, while state-of-the-art approaches tend to be complex and less user-friendly. In contrast, SAND provides a practical, application-driven solution inspired by these challenges, reinforcing its effectiveness in real-world scenarios.

We present in Table 6 the benchmarking results of SAND alongside other methods on the MSI Grain dataset. The target number of selected features is $k = 9$ and the experiments were run across 5 trials. All other experimental set-ups and pre-processing steps are the same as discussed in Section 3. Along with its remarkable simplicity compared to other methods, SAND achieves the highest performance, as demonstrated below.

Table 6. Test accuracy over 5 trials on MSI Grain (mean $\pm$ standard deviation).

Method	SA	LLY	GL	SL	STG	SAND
Accuracy	0.917 $\pm$ 0.003	0.917 $\pm$ 0.001	0.916 $\pm$ 0.002	0.918 $\pm$ 0.002	0.911 $\pm$ 0.005	**0.919** $\pm$ 0.002
All Features Accuracy	0.924 $\pm$ 0.001					

It is important to note that the segmentation in Figure 6 occurs in real-time, with the trained neural network running directly on the handheld edge device capturing the image. No cloud computing is involved—the tablet is used solely for visualization. Real-time edge processing was made possible by feature selection, which reduced the number of relevant spectral bands to nine, significantly decreasing both the model size and the number of spectral acquisitions. This optimization enabled the algorithm to be deployed on the edge device and operate in real-time.