Embedded feature selection using dual-network architecture

Abderrahim Abbassi[a,*], Arved Dörpinghaus[a], Niklas Römgens[a], Tanja Grießmann[a], Raimund Rolfes[a]

[a]*Leibniz University Hannover, Institute of Structural Analysis, Appelstraße 9A, 30167 Hanover, Germany*

Abstract

Feature selection is essential for eliminating noise, reducing redundancy, simplifying computational complexity, and lowering data collection and processing costs. However, existing methods often face challenges due to the complexity of feature interdependencies, uncertainty regarding the exact number of relevant features, and the need for hyperparameter optimization, which increases methodological complexity.

This research proposes a novel dual-network architecture for feature selection that addresses these issues. The architecture consists of a task model and a selection model. First, redundant features are fed into the selection model, which generates a binary mask aligned with the input feature dimensions. This mask is applied to a shifted version of the original features, serving as input to the task model. The task model then uses the selected features to perform the target supervised task. Simultaneously, the selection model aims to minimize the cumulative value of the mask, thus selecting the most relevant features with minimal impact on the task model's performance.

The method is evaluated using benchmark and synthetic datasets across different supervised tasks. Comparative evaluation with state-of-the-art techniques demonstrates that the proposed approach exhibits superior or competitive feature selection capabilities, achieving a reduction of 90% or more in feature count. This is particularly notable in the presence of non-linear feature interdependencies. The key

*Corresponding author

Email addresses: a.abbassi@isd.uni-hannover.de (Abderrahim Abbassi), arved.doerpinghaus@stud.uni-hannover.de (Arved Dörpinghaus), n.roemgens@isd.uni-hannover.de (Niklas Römgens), t.griessmann@isd.uni-hannover.de (Tanja Grießmann), r.rolfes@isd.uni-hannover.de (Raimund Rolfes)

advantages of the proposed method are its ability to self-determine the number of relevant features needed for the supervised task and its simplicity, requiring the pre-definition of only a single hyperparameter, for which an estimation approach is suggested.

Keywords: Feature selection, dual-network architecture, feature redundancy, non-linear interdependencies

---

1. Introduction

Due to the rising versatility of machine learning applications and the increasing intricacy of the tasks they accomplish, the underlying datasets are constantly growing in size and complexity. Often, data are gathered without the specific model and job in mind, resulting in vast amounts of data being collected and stored. These abundant quantities induce demand for methods that select relevant variables for a given predictive task and filter out correlations and dependencies. These variables are referred to as *features*, and methods for selecting relevant subsets perform *feature selection* (Kira and Rendell, 1992; Li et al., 2017; Kumar and Minz, 2014). By selecting a meaningful feature subset, reduces the dataset in size, resulting in a more manageable and interpretable representation. Pruning datasets of irrelevant or redundant variables offers several benefits, like a reduction in model complexity, an improvement in computational efficiency, the mitigation of overfitting (Ying, 2019), and possible enhancements of the predictive performance by removing noisy data (Li et al., 2017). Focusing on the most informative features, which exhibit a strong relationship with the target variable, the model better captures the underlying patterns (Li et al., 2017). A smaller dataset (or model) reduces sensor and power usage, computing requirements, and storage demands, leading to cost savings, streamlined data collection processes, efficient data management, and better accessibility.

Applying sufficient feature selection has become unquestionably important as dataset sizes grow and machine learning is employed across a broadening range of applications.

2. Related work

The wide variety of existing feature selection methods can broadly be classified into three categories using the taxonomy described in (Guyon and Elisseeff, 2003a; Chandrashekar and Sahin, 2014). Methods are categorized as *filter*, *wrapper*, and

embedded methods depending on their working principles. The first evaluates the relevance of features by looking at the intrinsic properties of the data, without involving any learning algorithm. These methods typically use statistical techniques to rank features based on their correlation with the output variable. Common techniques include Pearson's correlation coefficient, the Chi-square test, mutual information, and various statistical tests (e.g., t-test, ANOVA). Those methods have the main advantage of being computationally efficient, easy to compute, and can handle high-dimensional data but suffer from a lack of specificity and overly broad assumptions, e.g., data distribution or linearity (Bommert et al., 2020). Wrapper methods involve a learning algorithm to evaluate the importance of feature subsets. These methods search through the space of possible feature subsets and use a predictive model to evaluate their performance. Techniques like recursive feature elimination (Guyon et al., 2002a), forward selection (Guyon and Elisseeff, 2003b), backward elimination (Koller and Sahami, 1996), and genetic algorithms (Leardi, 1994) are commonly used in wrapper methods. While these methods tend to provide better performance than filter methods by considering feature dependencies and interactions, they are computationally expensive, especially with large datasets and high-dimensional feature spaces (El Aboudi and Benhlima, 2016). Therefore, performance is often approximated rather than computed for each subset (El Aboudi and Benhlima, 2016; Kursa and Rudnicki, 2010; Lundberg and Lee, 2017; Mlambo et al., 2016). Embedded methods perform feature selection during the process of training a learning algorithm. These methods integrate feature selection into the model training process, often leading to more accurate and efficient feature selection. Examples of embedded methods include decision trees and tree-based methods like random forests (Ho, 1995) and gradient boosting (Xu et al., 2014), LASSO (Least Absolute Shrinkage and Selection Operator) (Tibshirani, 1996), and elastic net regularization (Zou and Hastie, 2005). Embedded methods strike a balance between the computational efficiency of filter methods and the accuracy of wrapper methods. By considering feature interactions while being less computationally intensive than wrapper methods, these methods are increasingly gaining interest. This work focuses on some of the leading approaches for feature selection within this category.

The **concrete autoencoder** (Balın et al., 2019) uses an autoencoder architecture that determines a pre-defined number of important features, which itself requires certain knowledge about the task and dataset. It employs a concrete selector layer in the encoder, which learns to select a specified number of features from the input while allowing end-to-end training through gradient-based optimization. This method is unsupervised and hence does not result in a trained model for inference. While the method is effective, it requires prior knowledge about the number of relevant features

necessitating extensive experimentation to optimize.

Shapley **A**dditive Ex**p**lanations (SHAP) values (Lundberg and Lee, 2017) are used for interpreting machine learning models by quantifying the contribution of each feature to the model's predictions. They are based on cooperative game theory and provide a consistent way to attribute the importance of features. For feature selection, SHAP values are calculated for each feature across the dataset, representing their contribution to the prediction outcomes. By aggregating these values, features can be ranked according to their importance (Samek et al., 2017). Heuristics like LinearSHAP or DeepLIFT (Lundberg and Lee, 2017; Mishra, 2021) approximate SHAP values, addressing the combinatorial explosion of subsets and making SHAP practical for feature selection (Lundberg and Lee, 2017; Marcílio and Eler, 2020). Computing SHAP values can be computationally expensive, especially for complex models and large datasets, resulting in high processing times. Additionally, the method assumes feature independence and the absence of redundancy, which might not hold in all datasets, potentially impacting the accuracy of the selection task.

Feature **I**mportance **D**eep **L**earning (FIDL)(Wojtas and Chen, 2020) is a method for measuring feature importance in Deep Neural Networks (DNNs). It uses a dual-net strategy to find optimal feature subsets via binary masks, where large values indicate important features. FIDL employs a multiphase selection process with an exploration-exploitation strategy to generate mask candidates. The selector network aims to minimize the operator network's loss, and new masks are chosen based on the input gradient size of the selector (exploitation) and random generation (exploration). Unlike other methods, FIDL operates by changing subsets of masks, making it a suitable choice for comparison.

Stochastic **G**ate feature selection (STG) (Yamada et al., 2020, 2018) employs a continuously relaxed Bernoulli variable for feature selection using stochastic gates. This method does not require specifying the number of features in advance. However, it necessitates a large number of parameters to be trained in the first layer, leading to a risk of overfitting, particularly with deep neural networks handling high-dimensional data or when the training dataset is limited (Li et al., 2023; Singh et al., 2023).

The previously discussed approaches, while effective, face several limitations such as dependency on the choice of hyper-parameters, susceptibility to redundant information, and the need for prior knowledge about the number of relevant features. This research addresses these challenges by introducing a novel embedded method for feature selection that employs a dual-net architecture. The architecture consists of a task model and a selection model. Initially, the input data are fed into the selection model, which generates a quasi-binary mask (that is a mask that ap-

proaches a binary mask) aligned with the input feature dimensions. This mask is applied to a shifted version of the original features, which serves as input to the task model. The task model then uses the selected features to perform the target supervised task. Concurrently, the selection model aims to minimize the cumulative value of the mask, thereby selecting the most relevant features with minimal impact on the task model's performance. The contributions and advantages of the proposed methodology are threefold: (1) Firstly, it effectively performs feature selection even in the presence of information redundancy and non-linear feature dependencies, achieving accuracy comparable to or better than the best of the compared state-of-the-art methods. (2) Secondly, while some state-of-the-art methods are limited by their complexity and the requirement for extensive hyper-parameter optimization, which increases the time and computational costs, this work emphasizes simplicity, requiring the optimization of only one hyper-parameter, for which a reliable estimation is suggested. (3) Lastly, selecting the minimal subset of features relevant to the supervised task often necessitates prior knowledge about the data and the number of relevant features. In the proposed approach, the number of relevant features is determined automatically, making the method more suitable for real-world applications and large, complex datasets.

This article, after having presented an introduction to the problem and existing possible solutions will explain the working principle of the new approach, present results proving its capability to achieve the stated objectives, compare the performance to the previously described methods, and discuss the determination of the hyper-parameter and possible limitations.

3. Fundamentals of dual-net feature selection

This paper proposes a dual-net approach, referred to as DNFS in the rest of this paper, which involves two jointly trained DNNs, one *task* network and one *selection* network. In the following, the task and selection networks will be denoted by Π and Γ, respectively. Π_t and Γ_t refer to the specific weights, biases, and optimizer states at a time t. $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$ represents the dataset, where $\mathcal{X} \subset \mathbb{R}^n$ denotes the set of instances as real-valued feature vectors in $\mathbb{R}^n$, and $\mathcal{Y} \subset \mathbb{R}$ represents the target variable. Furthermore, $\mathcal{V}$ refers to the set of all variables (features) of a given dataset with the feature count $n \coloneqq |\mathcal{V}|$. Obviously, we have $|\mathcal{X}| = |\mathcal{Y}|$. $\mathcal{D}' = \{\mathcal{X}', \mathcal{Y}'\}$ denotes a shuffled version of the dataset and $x' \in \mathcal{X}'$ acts as a placeholder for some random instance. To clarify the distinction, we define variables as raw data inputs or measurements obtained from the system, while features are derived or transformed representations of these variables, relevant for predictive modeling or classification

tasks. Despite this distinction, the proposed approach can be applied to the selection of either variables or features, depending on the input to the task model.

The task network's target is to accomplish the supervised learning task on a feature subset provided by the selection network. The goal of the selection network is to generate a quasi-binary mask $\mathbf{m} \in [0,1]^n$ where 0 indicates a redundant or unimportant variable, and 1 indicates an important variable. The modified dataset

$$\mathcal{D}_{\mathbf{m}} \coloneqq \mathcal{D} \otimes \mathbf{m} \coloneqq \{(x \otimes \mathbf{m}, y)\}_{x \in \mathcal{X}, y \in \mathcal{Y}}$$

(here $\otimes$ refers to the Hadamard (Horn, 1990) product) ought to maximize the performance of the task network while simultaneously ensuring that $\mathbf{m}$ is as sparse as possible. The objectives of the networks are controlled via their loss functions

$$\mathcal{L}_{task}(x, y, \mathbf{m}; \Pi) = l(\Pi(x \otimes \mathbf{m}), y), \tag{1}$$

$$\mathcal{L}_{sel}(x, y; \Pi, \Gamma) = \lambda \cdot \mathcal{L}_{task}(x, y, \mathbf{m}; \Pi) + p(\mathbf{m}), \tag{2}$$

.

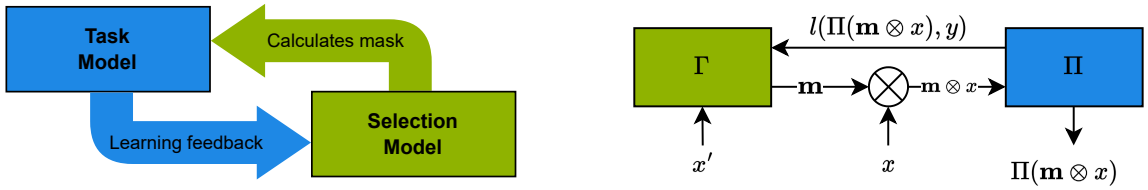
$$\text{where } \mathbf{m} \coloneqq \Gamma(x').$$

The task network's loss $\mathcal{L}_{task}$ consists of an instance-level task-dependent loss function l, e.g., mean squared error (MSE) for regression or cross-entropy for classification, respectively. The selection network's loss $\mathcal{L}_{sel}$ is a weighted sum of the task network's loss and a penalization function

$$p : [0,1]^n \to \mathbb{R}_{\geq 0}$$

that punishes the use of too many features, encouraging the selection of a smaller, more relevant subset. The parameter λ weighs the sum to achieve the desired selection aggressiveness, where a higher λ increases the penalty on feature usage, promoting sparser feature selection, and a lower λ reduces the penalty, allowing for more features to be included. The concept of weighting and summing different loss functions to achieve multiple training goals is uncommon in machine learning and can be used when multiple goals are related to the optimization task, such as regularization terms and multi-task learning like physics-informed neural networks (Abbassi et al., 2024). The novelty of DNFS lies in outsourcing the penalization to a separate model so that the training of the model performing the supervised task is not constrained by a more complex loss function. Unlike other embedded methods, such as Lasso, where the task model simultaneously handles both prediction and feature selection, our approach decouples these tasks, allowing the selection model to focus solely on

optimizing the feature subset. The working concept of the proposed approach is illustrated in Figure 1a.



(a) Concept of the working principle.

(b) Model interactions during one update.

Figure 1: Illustration of the workflow of the proposed DNFS approach.

The selection network receives feedback through the learning performance of the task model. It then generates a quasi-binary mask $\mathbf{m} \in [0, 1]^n$. A new dataset is generated via

$$\mathcal{D}_\mathbf{m} = \mathcal{D} \otimes \mathbf{m}.$$

The task model is then trained for one iteration on $D_\mathbf{m}$. Features with lower importance are eliminated by multiplication by 0, while important features are preserved by the multiplication with non-zero values (for mask entries $0 < \mathbf{m}_i < 1$, the feature undergoes a linear transformation which the task model's first layer can easily compensate). The selection model incorporates the task model's learning feedback and adjusts its output correspondingly. The interactions of the networks are illustrated in Figure 1b. Since the selection model should output a global mask, i.e., an input-independent mask, it is trained on a shuffled dataset denoted by $\mathcal{D}'$. A global mask refers to a mask that does not depend on a specific instance $x \in \mathcal{D}$ but rather remains unchanged for all $x \in \mathcal{D}$, i.e., it contains the significant features for the whole dataset. The process is described in detail in algorithm 1.

Before the training starts, both models are initialized (the specific implementation uses Glorot uniform initialization (Glorot and Bengio, 2010)). The selection model generates a quasi-binary mask $\mathbf{m}_0$ based on an instance $x' \in \mathcal{D}'$ of the shuffled dataset, which, at this point, is just a random output. The training data are multiplied by $\mathbf{m}_0$ to generate the initial dataset $\mathcal{D}_{\mathbf{m}_0}$ and passed on to the task model. This updates according to the employed optimizer (cf. Appendix F). Subsequently, the loss between the updated task model's predicted values and the actual values is recalculated. The loss of the selection model is computed as demonstrated in Eq. 2 and the selection model itself undergoes an update, initiating the process anew. The procedure is detailed in Figure 2. For efficiency reasons, in practice, the model updates happen batch-wise, unlike on the whole dataset. The training continues

Algorithm 1 Dual-Net Feature Selection

Input: dataset $\mathcal{D}$, y target variables, networks Π, Γ, networks parameter $\theta_\Pi, \theta_\Gamma$ epochs E, hyperparameter λ, p penalization function (cf. Section 6).
Output: trained task model Π and mask $\mathbf{m}$
Generate randomized dataset $\mathcal{D}' \coloneqq \text{shuffle}(\mathcal{D})$
Initialize $\mathbf{m} = \Gamma(x'), \quad x' \in \mathcal{D}'$.
repeat
  for $\mathcal{B}$ *(batch)* **in** $\mathcal{D}$ **do**
    calculate $\mathcal{L}_{task}(\mathcal{B}, y, \mathbf{m}; \Pi) = l(\Pi(\mathcal{B} \otimes \mathbf{m}), y)$
    calculate $\mathcal{L}_{sel}(\mathcal{B}, y; \Pi, \Gamma) = \lambda \cdot \mathcal{L}_{task}(\mathcal{B}, y, \mathbf{m}; \Pi) + p(\mathbf{m})$
    Update θ_Π using $\mathcal{L}_{task}$
    Update θ_Γ using $\mathcal{L}_{sel}$
    $\mathbf{m} = \Gamma(x')$.
  end for
until stopping criterion is fulfilled.

either for a predetermined number of epochs or until a termination criterion is met. Examples of such a criterion are a certain number of features being set to 0, the mask $\mathbf{m}$ converging to a certain subset, or a predefined early stopping criterion (cf. section 6.5). The method returns a feature subset $\mathcal{S} \subset \mathcal{V}$ containing the features deemed relevant. The cardinality $|\mathcal{S}|$ denotes the number of selected features.

An optimal feature selection method should be able to minimize the number of features while maintaining the same or a similar information density in the dataset, ensuring that a model can be trained on the remaining data without loss in performance. Further, it should effectively **prune highly correlated or redundant features**, even when the interdependencies are **non-linear** and the feature space is **high-dimesional**. Many existing feature selection methods require the target number of features to be specified beforehand. This requires an initial understanding of the dataset and an estimate of the necessary percentage of features, which is often not a trivial task. The proposed method, by directly incorporating learning feedback, aims to **determine the number of relevant features** autonomously. These are the primary objectives of this approach.

While the dual-network structure has been employed in other methods, their working principles differ significantly from DNFS. For instance, FIDL uses an operator net and a selector net, trained alternately. The operator net performs supervised learning, while the selector net refines feature subsets based on performance feedback. FIDL focuses on ranking feature importance and incorporates exploration-

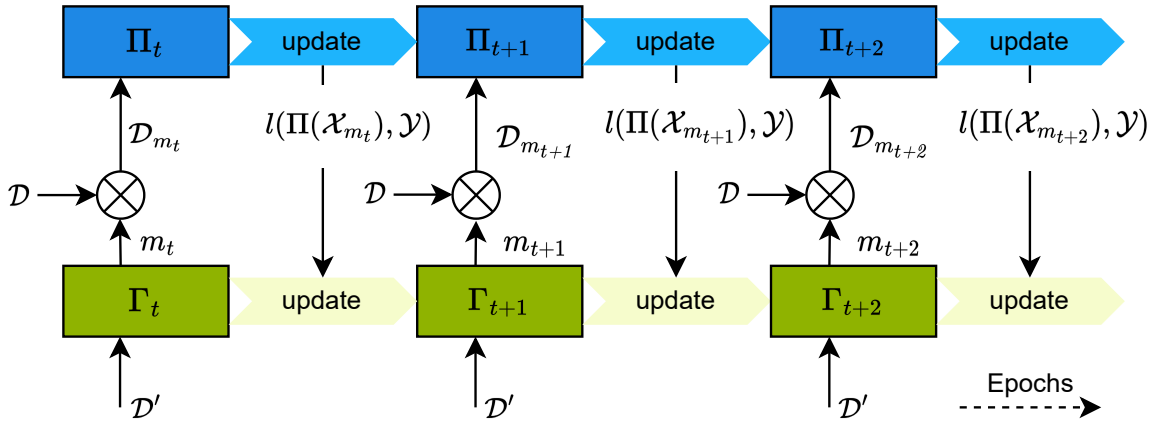


Figure 2: Model interactions over time. The concrete implementation performs batchwise instead of epoch-wise updates (cf. Algorithm 1).

exploitation with perturbation to avoid local optima. In contrast, DNFS trains its task network and selection network jointly. The selection network generates a sparse, quasi-binary mask by combining task performance loss with a sparsity penalty, enabling autonomous feature selection. This makes DNFS highly efficient for high-dimensional datasets, as it emphasizes sparse feature selection and uses global masks. Compared to CA, which employs a concrete selector layer to sample feature subsets and focuses on reconstructing input data, DNFS prioritizes task performance and sparsity. While CA excels in scenarios requiring feature interpretability and reconstruction accuracy, DNFS avoids the computational overhead of reconstruction, making it more efficient for large-scale tasks. Similarly, STG introduces a probabilistic approach, using stochastic binary gates to mask features. STG encourages sparsity by penalizing the probability of gates being active, making it suitable for tasks requiring highly sparse feature subsets. However, unlike DNFS, which uses a deterministic quasi-binary mask, STG relies on probabilistic gates. This makes DNFS more interpretable, as mask values directly indicate feature importance, while STG's probabilistic nature is advantageous for exploring uncertain feature interactions. DNFS stands out for its efficiency, interpretability, and autonomous feature selection, making it ideal for high-dimensional datasets. While FIDL ranks feature importance, CA focuses on reconstruction, and STG offers probabilistic sparsity, DNFS uniquely balances sparsity, performance, and scalability.

In the following sections, we present investigation results that demonstrate the method's ability to achieve these targets and compare its performance to other state-

of-the-art feature selection techniques.

4. Empirical evaluation

To assess the capabilities of DNFS, the method was tested on various datasets and tasks.

4.1. Datasets

To evaluate DNFS's ability to eliminate redundancies in feature sets, synthetic datasets containing redundant features were engineered. The underlying synthetic dataset contains ten features, where the vector

$$(X_0, X_1, X_2, X_3, X_4, X_9) \sim \mathcal{N}(0, I_6)$$

follows a standard normal distribution. The remaining feature set $\{X_5, X_6, X_7, X_8\}$ is populated via linear and non-linear (n.l.) dependencies, resulting in two distinct datasets. The dependencies for the linear case are introduced as follows:

$$X_5 = 6X_2, \quad X_6 = -\frac{1}{2}X_3, \quad X_7 = 5X_0, \quad X_8 = -2X_1 \tag{3}$$

For the non-linear case, the X_5 to X_8 were populated via:

$$X_5 = X_2^3, \quad X_6 = -\frac{1}{2} \cdot \frac{X_3^5}{1 + \exp(X_0)}, \quad X_7 = 5\tan^{-1}(X_0), \quad X_8 = -2\exp(X_1^3) \tag{4}$$

For each dataset, a regression (Wojtas and Chen, 2020; Chen et al., 2018) and a classification (Wojtas and Chen, 2020; J. Friedman) task were defined with targets Y_{reg}, Y_{cl} set as follows:

$$Y_{reg} = -2\sin(X_0) + \max(X_1, 0) + X_2 + \exp(-X_3) + \varepsilon \tag{5}$$

$$Y_{cl} = \begin{cases} 0 & \text{if } 4 \leq \sum_{i=0}^{3} X_i^2 \leq 16, \\ 1 & \text{else.} \end{cases} \tag{6}$$

where $\varepsilon \sim \mathcal{N}(0, 0.1)$ represents noise added to the target. The condition in Eq. 6 leads to 60%-40% distribution of ones and zeros, making for a balanced classification task.

For the benchmark tasks, the well-known MNIST (Lecun et al., 1998), Basehock, and Ames housing (Cock, 2011) datasets were chosen. The first involves an image classification task, the second a quasi-binary classification task, and the third a

regression task predicting housing prices. All these datasets include redundancies in their features and can be significantly reduced in size without degradation of the information density. These benchmarks may validate the achievement of the stated objectives by the proposed method,, even when applied to more complex, real-world tasks. For a detailed overview of the datasets and their properties, refer to Table H.10 in Section Appendix H.

4.2. Evaluation setup

Analyzing the results of the synthetic case is a comparatively simple task since these can be classified deterministically. Because the underlying dependencies and redundancies are known, it is possible to precisely quantify the quality of selected subsets. To simplify the analysis of the results obtained using the synthetic datasets, we introduce the following definitions for a subset of the set of all variables $\mathcal{S} \subseteq \mathcal{V}$:

Definition 1. *$\mathcal{S}$ is called valid if it contains all information about Y, i.e., X_0 to X_3 can be isolated via equivalence transformations and simultaneously does not contain random noise, i.e., features.*

Definition 2. *$\mathcal{S}$ is called minimal if it has size four, i.e., $N := |\mathcal{S}| = 4$. This originates from the fact that there are four unique variables and only bijective transformations.*

Definition 3. *$\mathcal{S}$ is called optimal if it is valid and minimal. A selection is called optimal when the selected subset is optimal.*

A valid subset is defined as one which contains all information about the target. For both datasets, the linear and non-linear sets, the minimal size of a valid subset is 4. If the method has selected valid subsets of minimal size (within a margin), the result is deemed optimal. Note here that if a method requires a target feature count the percentages of valid and optimal subsets are equal. Although the minimal size is well defined, the specific subsets that qualify as valid are not unique. For instance, both $\{X0, X1, X2, X3\}$ and $\{X5, X6, X7, X8\}$ are optimal subsets, yet they share no common variables. Each dataset presents 16 potential valid subsets. The concrete subset varies depending on the dataset, the model weight initialization, and several random states in the optimizer. To account for this variability, each selection process was repeated multiple times, and the results were averaged to provide a robust measure of performance.

Evaluating the performance of subsets using benchmark datasets poses a challenge due to the inherent ambiguity in both their sizes and the specific features they contain. As a result, performance assessment was conducted by comparing the

model's performance before and after subset selection. To establish a baseline, a machine learning model with an architecture identical to that of the task model was trained on the entire dataset. The validation performance of this full model was then recorded as the baseline performance for the dataset. Following the subset selection process, the model's performance was re-evaluated using the reduced subset of features, employing the same training and validation procedures. This comparative approach makes a consistent evaluation of the impact of feature selection on model performance possible.

5. Results

We performed feature selection using DNFS on the previously presented datasets employing the proposed methodology. The results of the selection of the synthetic and benchmark datasets are summarized in Tables 1 and 2.

Table 1: Results of DNFS on synthetic datasets.

Dataset	Valid	N	Optimal
synth. reg.	100%	4	100%
synth. cl.	100%	4	100%
synth. reg. n.l.	100%	4.132 ± 0.4	95%
synth. cl. n.l.	100%	4.325 ± 0.52	93%

For all four synthetic datasets, DNFS consistently produced optimal results, with a standard deviation in subset size of less than 10%. Remarkably, the method yielded valid subsets almost 100% of the time, underscoring that the performance of the task model is prioritized over the removal of features. The best performance, indicated by the highest percentage of optimal subsets, was observed in the linear regression task, where DNFS universally selected optimal subsets. The greatest deviation in subset size was seen with the non-linear classification task, with a standard deviation of 0.52.

An illustrative outcome of Γ_{200} applied to synthetic regression with linear dependencies is exemplified by the following vector:

$$\mathbf{m}_{200} \approx (0.1, 0.2, 0.1, 0, 0, 0, 0.4, 0, 0, 0)^T$$

Interpreting the non-zero entries as selected features translates to the subset

$$\mathcal{S} = \{X_0, X_1, X_2, X_6\}, \quad \text{with} \quad |\mathcal{S}| = 4$$

being selected in epoch 200.

Table 2: Results of DNFS on benchmark datasets.

Dataset	baseline	sel. perf.	N
MNIST	$96,9 \pm 0,01\%$	$95,5\% \pm 0,3\%$	50 ± 2
Ames H.	$17.000 \pm 300\$$	$15.000\$ \pm 500\$$	20 ± 3
Basehock	96%	95%	53 ± 4

Table 2 shows the results for the MNIST, Ames, and Basehock datasets. The baseline was established by training a model with the same architecture as the task model on the whole dataset and measuring its validation performance. The intention behind using the same architecture is to ensure a fair and controlled evaluation, where the impact of the feature selection process on performance can be isolated from architectural or hyperparameter influences. For the MNIST dataset, DNFS achieved $\approx$ 94% feature reduction while maintaining a high validation accuracy of 95.5%. On the Basehock dataset, the feature count was reduced by more than 99% with a stable accuracy of 95%. For the Ames housing dataset, the reduction was around 85%, accompanied by a significant improvement in validation MAE, with a decrease of approximately 12%. To ensure that the observed performance improvements are not dependent on the architecture or hyperparameter settings of the task model, we additionally evaluated the selected features using a support vector machine (SVM). The SVM was trained on both the full feature set and the subset selected by DNFS. The results, provided in Table C.6, show similar performance trends to those reported in Table 2, thereby validating the effectiveness of the selected features in a model-independent manner.

Overall, DNFS successfully identified a significant number of relevant features across various datasets and achieved reductions in size surpassing 90% depending on the specific task. Notably, DNFS demonstrated consistent results with minimal variation, and the validation performance of the task model either remained similar or even improved.

6. Discussion

To contextualize the presented results, the same feature selection tasks were performed by the methods discussed in Section 2. This section will compare the results and provide a detailed analysis of the properties, advantages, and disadvantages of DNFS.

6.1. Comparative results on synthetic datasets

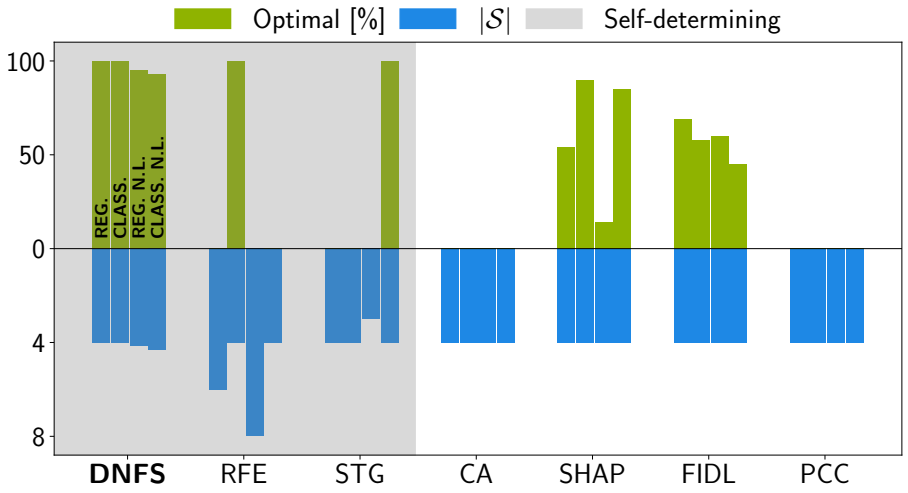
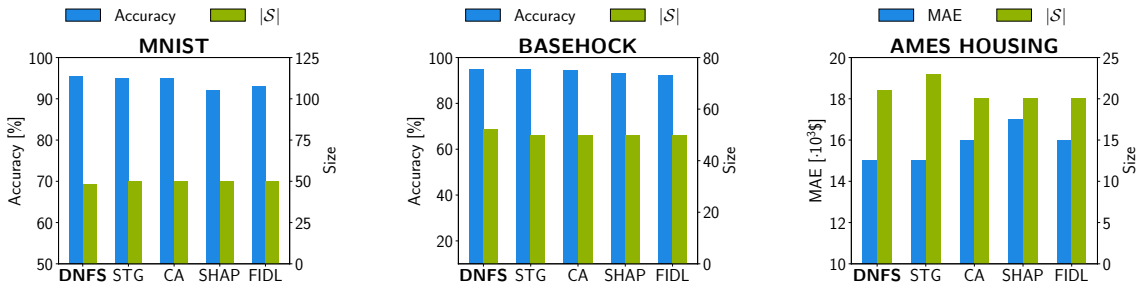


Figure 3: DNFS, RFE, STG, CA, SHAP, FIDL, and PCC results. The datasets are (from left to right): linear regression, linear classification, non-linear regression, non-linear classification. CA, SHAP, and FIDL return a fixed number, i.e., 4 features. For exact results see Appendix A.

Figure 3 shows the performance of the seven different methods on the synthetic datasets. In addition to state-of-the-art embedded methods, **P**earson **C**orrelation **C**oefficient (PCC) and **R**ecursive **F**eature **E**limination (RFE) (Guyon et al., 2002b) are included in the comparison. PCC identifies relevant features by calculating absolute correlation values, discarding those with little influence on the outcome. RFE, on the other hand, is a feature selection technique that iteratively removes the least important features to improve model performance. It works by fitting a model, ranking features by importance, and recursively eliminating the least significant ones until a desired number of features remains. RFE is commonly used with algorithms like support vector machines (SVMs) and decision trees, making it an effective wrapper method for selecting relevant features while maintaining accuracy.

The definitions for valid, minimal, and optimal results (subsets) are consistent with the criteria outlined in Section 4. The results highlight the superiority of DNFS in the feature selection task compared to all other implemented methods. DNFS successfully identifies optimal feature subsets for 100 % of the linear tasks and 94 % of the non-linear regression and classification tasks, demonstrating clear dominance over the compared methods (detailed results are provided in Table A.4). An interesting observation is that while RFE is theoretically expected to deliver optimal results, it often falls short in practice, selecting redundant features for nearly every dataset except the linear classification task. This limitation stems from the absence of deep learning models in the RFE process, which restricts its ability to capture complex, non-linear feature interactions. The use of synthetic datasets was intended to test the methods' ability to identify and prune even complex non-linear dependencies. While the other methods may also result in subsets delivering good performance in the linear tasks, they often struggle with either selecting the minimal number of features or, if so, the correct combinations for optimal performance that don't contain redundant information. Specifically, CA selects redundant features, and STG doesn't succeed in selecting all relevant features for the fulfillment of the task or selects redundant ones as well. FIDL and SHAP produce mediocre results, frequently failing on multiple datasets to meet one of the objectives by selecting redundant features in many trials. Although these methods can achieve optimal results, they do not consistently reproduce them. PCC fails to remove redundancies and instead consistently selects the redundant features with the highest correlation to the target in all cases.

6.2. Comparison of Embedded Methods on Benchmark Datasets



(a) (b) (c)

Figure 4: Comparative results on the benchmark datasets.

After analyzing a multitude of different methods, this section focuses on comparing DNFS to other embedded methods on benchmark datasets. Since DNFS is itself

an embedded method, we limit our analysis to this category to ensure a fair and meaningful comparison. On MNIST, DNFS achieves the best results with the best validation performance and the highest validation accuracy among all methods for the given feature count. The validation performance with DNFS is nearly identical to the baseline, showing no significant deviation and remaining well within the margin of error, which is markedly better than the declines observed with the other methods. The targeted feature count was 50 features, as this was the average number selected by DNFS, and further shrinking the target did not result in improved performance of the other methods.

On the Basehock dataset, DNFS demonstrates the capability to perform competitively in a high-dimensional setting, matching the performance of methods such as STG, CA, FIDL, and SHAP. Although DNFS selected more features than the other methods, it also achieved the highest accuracy among them. This may be due to the elimination of relevant features by the other methods, which, as observed with STG in non-linear tasks, results in smaller selected feature subsets but at the cost of performance.

Regarding the Ames Housing dataset, DNFS successfully identified the optimal subset of features, leading to superior performance in the regression task compared to subsets selected by other methods. Notably, when compared to STG-the only other method capable of autonomously determining the number of relevant features—DNFS not only selected fewer features but also achieved better results. This dataset is particularly challenging due to its high proportion of categorical features, which introduces significant sparsity, especially after one-hot encoding.

Table 3: Comparison of study contents with other works. *While the authors outline how to apply their method in a supervised setting (Balın et al., 2019), no implementation is provided, therefore it hasn't been tested.

	DNFS	RFE	STG	CA	SHAP	FIDL	PCC
Self-Determining	✓	✓	✓	✗	✗	✗	✓
Redundancy-Elimination	✓	✓	–	✗	✗	✗	✗
Few Hyperparameters	✓	✓	✗	✓	✓	✗	✓
Supervised	✓	✓	✓	–*	✓	✓	✓
Simplicity	✓	✗	✗	✓	✓	✗	✓

The comparison results demonstrated the superiority of DNFS in selecting the optimal subset of features, particularly in the presence of non-linear dependencies, as observed in the synthetic data, and in high-dimensional datasets like MNIST and Ames Housing. DNFS excels in selecting all relevant features for the target, avoiding redundant features, and maintaining or improving task model performance. A key advantage of DNFS is its ability to determine the optimal number of features without requiring any prior specification, setting it apart from methods such as CA, FIDL, SHAP, and other state-of-the-art feature selection techniques. In practical applications, identifying the optimal feature count often requires extensive optimization. However, DNFS inherently addresses this challenge without the need for predefined targets. Moreover, DNFS offers simplicity, requiring minimal hyperparameter tuning. This simplicity is further exemplified by the use of a single hyperparameter, λ, whose optimal value is suggested and discussed in section 6.3. The training duration of DNFS increases by approximately $1.2\times$ to $2.6\times$, depending on the dataset characteristics and whether the task is regression or classification, compared to training the baseline model (task model without the selection task). This increase in training time can be attributed to the additional computational complexity introduced by the selection process, which requires the model to evaluate feature relevance during training. Despite this increase, the enhanced interpretability and improved performance obtained through feature selection justify the additional computational effort. Further details, including runtime comparisons and hardware specifications, are provided in Appendix E.4 and Table E.9. STG is particularly sensitive to a range of hyperparameters, which can result in substantially different outcomes. For instance, modifications to the optimizer or number of epochs lead to significant variations in performance. To the best of our knowledge, the authors have not explicitly addressed this issue. While other methods exhibit similar limitations, they are generally less pronounced compared to STG. A high number of sensitive hyperparameters diminishes the practicality of a method, as it necessitates extensive computational resources to identify optimal parameters through grid search or comparable algorithms, thereby significantly increasing runtime and complexity.

6.3. The impact of λ

The hyperparameter λ, which weights the sum in the selection network's objective function (cf. Eq. 2), is the only hyperparameter which directly influences the selection. Consequently, variations in λ can potentially significantly impact the outcomes. Other parameters are set indirectly, like the model architectures, the employed optimizer, and the method of weight initialization. Compared to λ, we have found none of these to be nearly as impactful, or impactful at all.

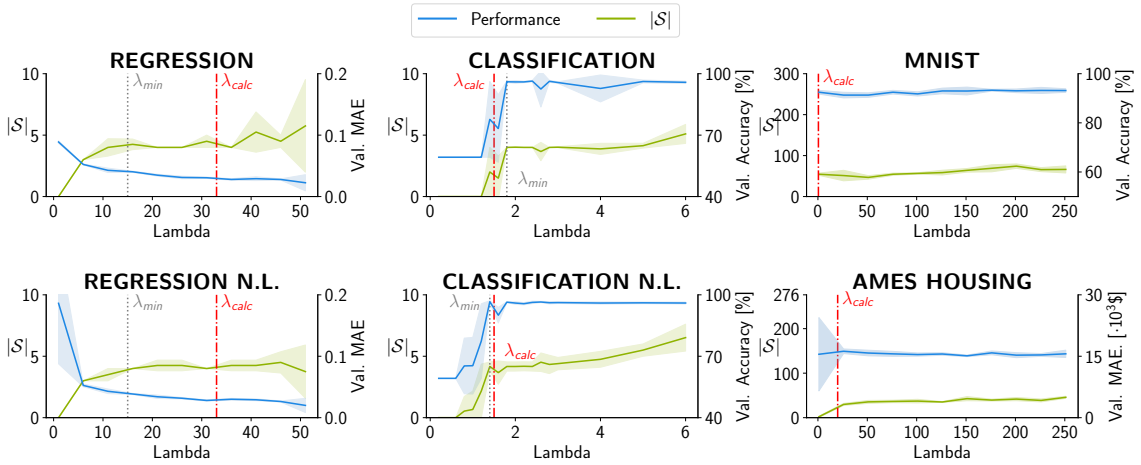


Figure 5: Selection behavior of DNFS for the synthetic datasets for increasing values of λ. Depending on λ's magnitude, the summands of (2) are weighted differently, resulting in an in- or decreased number of selected features. λ_{min} marks the first values at which DNFS converges reliably to the optimal solution. λ_{calc} represents the values calculated via the approximations presented in section 6.3. The regressions were performed on normalized target (cf. section 6.3) and, for Ames, scaled back to the original scale.

The aforementioned dependency on λ makes it the only interesting parameter to analyze. To better visualize its impact, Figure 5 shows the size $|\mathcal{S}|$ of the outputted subsets, their yielded performance, and the standard deviation of those metrics for all presented datasets. For all synthetic datasets, there exists a threshold value λ_{min} beyond which DNFS exhibits a stable convergence to optimal subsets. On the benchmark datasets, the intervals of convergence are even larger, spanning over one order of magnitude in length. It is important to note that no method converges at $\lambda = 0$, even if some of the graphs might suggest otherwise. This is because, for some datasets, DNFS still converges for small values.

Choosing λ is crucial for a successful selection. It depends on the dataset, the loss function $\mathcal{L}_{task}$, and the penalization function p. We employed categorical cross-entropy (resp. MSE) for classification (resp. regression) as loss function l in Eq. 1. Both losses can theoretically take values in whole $\mathbb{R}_{\geq 0}$ but can be approximated if one makes certain reasonable assumptions. The first is that the task model is reasonably well suited for the predictive task and, in the case of a regression, that the target Y is normalized, which can always be achieved via min-max transformation:

$$Y_{norm} = \frac{Y - \min(Y)}{\max(Y) - \min(Y)}$$

For classification, it is then reasonable to approximate l as

$$l \leq -\log(\frac{1}{|\text{Classes}|}) = \log(|\text{Classes}|),$$

which is the loss resulting from the task model uniformly guessing ($|\text{Classes}|$ denotes the number of classes of a given classification task). In the case of regression, we assume that the mse deviates approximately 10% around the mean $\overline{Y}_{norm}$ which, especially for later epochs, is fairly generous. Choosing λ such that $\lambda \cdot \mathcal{L}_T$ and p are of the same order of magnitude (for $p(\mathbf{m}) = \overline{\mathbf{m}}$ we have $0 \leq p \leq 1$) leads to:

$$\lambda_{class} \geq \frac{1}{\log(|\text{Classes}|)}, \quad \lambda_{reg} \geq \frac{1}{0.1 \cdot \overline{Y}_{norm}} \tag{7}$$

These are not strict boundaries but result from, in this context, justified generalizations. The calculated values λ_{calc} are shown in Figure 5 (cf. Appendix E.2 Table E.7). All lie in the area of convergence except for the linear classification dataset, for which the calculation is slightly too small. This further emphasizes that, for optimal selection of λ, an initial value can be derived based on the Eq. 7, with further possible improvement of results through trials with other λ values. Additionally, this suggests that the model's uniform guessing approximation might be overly pessimistic, which is likely the case.

6.4. Convergence behavior

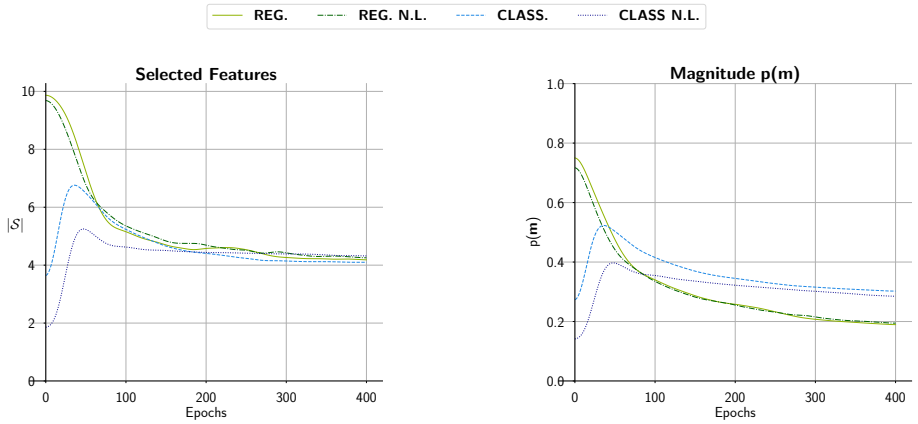


Figure 6: Convergence behavior of the subset size DNFS selects (left) and the magnitude of the penalization function p (right) on the synthetic datasets. The values were averaged over multiple trials.

DNFS exhibited the desired performance on all datasets and converged to optimal solutions. Figure 6 examines further the convergence behavior of DNFS on the synthetic datasets for a specific λ (cf. Appendix E Table E.7). The left graphs in Figure 6 show the number of features being selected in every epoch, i.e., the number of non-zero entries in $\mathbf{m}$, while the right show the magnitude of the penalization function p. In the concrete implementation, we chose $p(\mathbf{m}) = \overline{\mathbf{m}}$ as the arithmetic mean of $\mathbf{m}$. Comparing the two plots reveals that while $|\mathcal{S}|$ converges, p seems not to. The cause lies in the continuity of p. Since p can be reduced by linearly decreasing all non-zero values, its magnitude decreases while simultaneously the number of selected features stays constant. This holds true for every continuous function p.

Proposition 1. *For every non-constant continuous function $f \in \mathcal{C}(\mathbb{R}^n)$ and $\varepsilon > 0$ with $f(x) = 0$ for some $x \in \mathbb{R}^n$ there exist $y \in \mathbb{R}^n$ such that $f(y) \neq 0$ and $|f(y)| < \varepsilon$.*

Proof. This is an immediate deduction from the intermediate values theorem for multivariate functions (Shimamoto, 2019) (cf. Appendix B). □

The result shows that the previously described problem can arise with any non-constant continuous penalization function. The following two masks are an example of the phenomenon on the synthetic datasets. After 200 epochs of training the selection network's output is:

$$\mathbf{m}_{200} \approx (0.2, 0, 0.2, 0.5, 0, 0, 0, 0, 0.2, 0)^T$$

After 400 epochs Γ_{400} generated

$$\mathbf{m}_{400} \approx (0.1, 0, 0.1, 0.3, 0, 0, 0, 0, 0.1, 0)^T \approx \frac{1}{2}\mathbf{m}_{200}$$

which corresponds to a halving of $\mathbf{m}_{200}$. Notably, no new feature has been set to zero, insinuating that the feature count has stabilized. Therefore, convergence criteria that solely depend on p are unfeasible.

6.5. Classifying selections and stopping criteria

Determining which features should be classified as selected and when to stop the selection is a non-trivial task in and of itself. In practice, $\mathbf{m}$ approaches gradually zero at indices whose corresponding features are being deemed unimportant. Nevertheless, this convergence can be slow and tends to decrease in pace as the neuron values inch toward 0. To address this, a classification threshold $\delta > 0$ is introduced to determine whether an entry should be classified as selected or not. Moreover,

the dropout layers (cf. Table E.8) help prevent the method from converging prematurely to a suboptimal subset but also introduce volatility into $\mathbf{m}$, causing mask entries to occasionally spike even after extended training if the feature is deemed unimportant. This problem can be mitigated using different strategies. A possible way is to calculate the moving average of the last t_{av} epochs and perform the classification on this averaged mask. This method alleviates the volatility problem but is still prone to the threshold δ beyond which a feature is classified as selected. The current implementation employs a more sophisticated statistical approach. Starting at epoch t_{start}, which gives the process time to stabilize, it records whether a feature is selected in each subsequent epoch. We refer to the interval $[t_{start}, t_{end}]$ as the selection interval. Post training, features deemed important are those selected in at least $p\%$ of the epochs. This generally is a stronger condition than the moving average method and yields better results. Determining the selected features in each epoch t requires setting a threshold δ. The hyperparameters p and δ are robust and generally independent of the dataset (cf. Appendix D Figures D.8 and D.7).

Deciding when to stop the selection can be achieved either by training for a fixed number of epochs or by using a classic early stopping criterion like validation loss monitoring of the task model's performance. The latter has been implemented in the current version.

6.6. Limitations

While DNFS introduces an innovative approach and demonstrates promising results in various scenarios, it is important to acknowledge its potential limitations. Like all approaches, it encounters certain constraints that may impact its performance and applicability in specific contexts. It necessitates a certain level of understanding, if not of the relationship between the features and targets, then of the task itself. First and foremost, the current implementation and theoretical foundation are designed for multilayer perceptrons (MLPs) and were not tested on different model architectures. Therefore, we cannot make any assertions about performance beyond classical MLPs. The specific architectures vary from task to task and need to be determined by a prior investigation. The models used by us are stated in Table E.8. Furthermore, requiring the training of deep neural networks DNFS is inherently more computing intensive than statistical methods or methods that employ simpler networks, e.g., linear models. The increase in the necessary time per epoch varies from 40% for the synthetic datasets to 160% for MNIST (for used hardware and precise runtimes see Appendix E.4).

To guarantee a comparable and fair evaluation, we trained a new network on the selected subsets. In praxis, though, it is recommendable to use the already

trained task network to perform the predictive task. This, however, is perhaps not always feasible since task model performance tends to lag behind a newly trained model. Convergence issues may arise with DNFS when variables in a dataset exhibit extremely uneven information density. In such cases, DNFS tends to select too few variables, failing to include those that, despite offering less information about the target compared to the already selected variables, still contribute to increases in performance, even if small. This problem may also arise if a lot of categorical features are present.

7. Conclusion

This study presents a new dual-network architecture for embedded feature selection. The main innovation is its ability to identify a subset of features that are essential for achieving the target variable, effectively addressing issues of redundancy and irrelevance. This method also tackles the challenge of not having prior knowledge about the number of features. Through evaluation and comparison with state-of-the-art methods using various synthetic and benchmark datasets in different supervised tasks, this approach demonstrates its potential to achieve high accuracy for the intended tasks while using a minimal feature set. The effectiveness of the selection process depends on the choice of the hyperparameter λ, which balances accuracy and the number of selected features.

Acknowledgments

The authors gratefully acknowledge the financial support provided by the Federal Ministry for Economic Affairs and Climate Action of the Federal Republic of Germany through the project "SONYA - Increasing the reliability of segmented rotor blades through hybrid condition monitoring" (FKZ 03EE3026B).

Declaration of conflicting interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used ChatGPT in order to improve the language. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

Abbassi, A., Römgens, N., Grießmann, T., Rolfes, R., 2024. Data augmentation with data-based simulation in structural health monitoring using knowledge embedded generative adversarial network. Available at SSRN 4687651 .

Balın, M.F., Abid, A., Zou, J., 2019. Concrete autoencoders: Differentiable feature selection and reconstruction, in: Chaudhuri, K., Salakhutdinov, R. (Eds.), Proceedings of the 36th International Conference on Machine Learning, PMLR. pp. 444–453. URL: https://proceedings.mlr.press/v97/balin19a.html.

Basodi, S., Ji, C., Zhang, H., Pan, Y., 2020. Gradient amplification: An efficient way to train deep neural networks. Big Data Mining and Analytics 3, 196–207. doi:10.26599/BDMA.2020.9020004.

Bommert, A., Sun, X., Bischl, B., Rahnenführer, J., Lang, M., 2020. Benchmark for filter methods for feature selection in high-dimensional classification data. Computational Statistics & Data Analysis 143, 106839. URL: https://www.sciencedirect.com/science/article/pii/S016794731930194X, doi:https://doi.org/10.1016/j.csda.2019.106839.

Chandrashekar, G., Sahin, F., 2014. A survey on feature selection methods. Computers & electrical engineering 40, 16–28.

Chen, J., Stern, M., Wainwright, M.J., Jordan, M.I., 2018. Kernel feature selection via conditional covariance minimization. arXiv:1707.01164.

Cock, D.D., 2011. Ames, iowa: Alternative to the boston housing data as an end of semester regression project. Journal of Statistics Education URL: https://jse.amstat.org/v19n3/decock.pdf.

El Aboudi, N., Benhlima, L., 2016. Review on wrapper feature selection approaches, in: 2016 International Conference on Engineering & MIS (ICEMIS), pp. 1–5. doi:10.1109/ICEMIS.2016.7745366.

Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks, in: Teh, Y.W., Titterington, M. (Eds.), Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, PMLR, Chia Laguna Resort, Sardinia, Italy. pp. 249–256. URL: https://proceedings.mlr.press/v9/glorot10a.html.

Guyon, I., Elisseeff, A., 2003a. An introduction to variable and feature selection. CoRR URL: https://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf.

Guyon, I., Elisseeff, A., 2003b. An introduction to variable and feature selection. Journal of machine learning research 3, 1157–1182.

Guyon, I., Weston, J., Barnhill, S., Vapnik, V., 2002a. Gene selection for cancer classification using support vector machines. Machine learning 46, 389–422.

Guyon, I., Weston, J., Barnhill, S., Vapnik, V., 2002b. Gene selection for cancer classification using support vector machines. Machine learning 46, 389–422.

Ho, T.K., 1995. Random decision forests, in: Proceedings of 3rd international conference on document analysis and recognition, IEEE. pp. 278–282.

Horn, R.A., 1990. The hadamard product, in: Proc. Symp. Appl. Math, pp. 87–169.

J. Friedman, T. Hastie, R.T., . The Element of Statistical Learning. Springer, Berlin, 2001.

Kingma, D.P., Ba, J., 2017. Adam: A method for stochastic optimization. arXiv:1412.6980.

Kira, K., Rendell, L.A., 1992. A practical approach to feature selection, in: Sleeman, D., Edwards, P. (Eds.), Machine Learning Proceedings 1992. Morgan Kaufmann, San Francisco (CA), pp. 249–256. URL: https://www.sciencedirect.com/science/article/pii/B9781558602472500371, doi:https://doi.org/10.1016/B978-1-55860-247-2.50037-1.

Koller, D., Sahami, M., 1996. Toward optimal feature selection. Technical Report. Stanford InfoLab.

Kumar, V., Minz, S., 2014. Feature selection: a literature review. SmartCR 4, 211–229.

Kursa, M.B., Rudnicki, W.R., 2010. Feature selection with the boruta package. Journal of Statistical Software 36, 1–13. URL: https://www.jstatsoft.org/index.php/jss/article/view/v036i11, doi:10.18637/jss.v036.i11.

Leardi, R., 1994. Application of a genetic algorithm to feature selection under full validation conditions and to outlier detection. Journal of Chemometrics 8, 65–79.

Lecun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324. doi:10.1109/5.726791.

Li, A., Peng, H., Zhang, L., Huang, J., Guo, Q., Yu, H., Liu, Y., 2023. Fedsdg-fs: Efficient and secure feature selection for vertical federated learning, in: IEEE INFOCOM 2023-IEEE Conference on Computer Communications, IEEE. pp. 1–10.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R.P., Tang, J., Liu, H., 2017. Feature selection: A data perspective. ACM Comput. Surv. 50. URL: https://doi.org/10.1145/3136625, doi:10.1145/3136625.

Lundberg, S.M., Lee, S., 2017. A unified approach to interpreting model predictions. CoRR abs/1705.07874. URL: http://arxiv.org/abs/1705.07874, arXiv:1705.07874.

Marcílio, W.E., Eler, D.M., 2020. From explanations to feature selection: assessing shap values as feature selection mechanism, in: 2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), pp. 340–347. doi:10.1109/SIBGRAPI51738.2020.00053.

Mishra, P., 2021. Explainability for deep learning models, in: Practical Explainable AI Using Python: Artificial Intelligence Model Explanations Using Python-based Libraries, Extensions, and Frameworks. Springer, pp. 243–263.

Mlambo, N., Cheruiyot, W.K., Kimwele, M.W., 2016. A survey and comparative study of filter and wrapper feature selection techniques. International Journal of Engineering and Science (IJES) 5, 57–67.

Samek, W., Wiegand, T., Müller, K., 2017. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. CoRR abs/1708.08296. URL: http://arxiv.org/abs/1708.08296, arXiv:1708.08296.

Shimamoto, D., 2019. Multivariable Calculus. Don Shimamoto.

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

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

Wojtas, M., Chen, K., 2020. Feature importance ranking for deep learning. CoRR abs/2010.08973. URL: https://arxiv.org/abs/2010.08973, arXiv:2010.08973.

Xu, Z., Huang, G., Weinberger, K.Q., Zheng, A.X., 2014. Gradient boosted feature selection, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 522–531.

Yamada, M., Tang, J., Lugo-Martinez, J., Hodzic, E., Shrestha, R., Saha, A., Ouyang, H., Yin, D., Mamitsuka, H., Sahinalp, C., Radivojac, P., Menczer, F., Chang, Y., 2018. Ultra high-dimensional nonlinear feature selection for big biological data. IEEE Transactions on Knowledge and Data Engineering 30, 1352–1365. doi:10.1109/TKDE.2018.2789451.

Yamada, Y., Lindenbaum, O., Negahban, S., Kluger, Y., 2020. Feature selection using stochastic gates, in: III, H.D., Singh, A. (Eds.), Proceedings of the 37th International Conference on Machine Learning, PMLR. pp. 10648–10659. URL: https://proceedings.mlr.press/v119/yamada20a.html.

Ying, X., 2019. An overview of overfitting and its solutions. Journal of Physics: Conference Series 1168, 022022. URL: https://dx.doi.org/10.1088/1742-6596/1168/2/022022, doi:10.1088/1742-6596/1168/2/022022.

Zou, H., Hastie, T., 2005. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society Series B: Statistical Methodology 67, 301–320.

Appendix A. Results of synthetic Datasets 1

Table A.4: Averaged results of DNFS compared to FIDL, STG, CA, and SHAP on the synthetic datasets. Green highlights the method with the most optimal selections.

		Linear			Non-Linear		
	Method	Val.	N	Opt.	Val.	N	Opt.
Regression	**DNFS**	**100**	**4**	**100**	**100**	**4.132**	**95**
	FIDL	54	4	54	90	4	90
	STG	100	4	0	100	3	0
	CA	100	4	0	100	4	0
	SHAP	69	4	69	60	4	60
	PCC	100	8	0	100	7	0
Classification	**DNFS**	**100**	**4**	**100**	**100**	**4.325**	**93**
	FIDL	14	4	14	85	4	85
	STG	100	4	0	100	4	100
	CA	100	4	0	100	4	0
	SHAP	58	4	58	45	4	45
	PCC	0	0	0	0	0	0

Appendix B. Proof of proposition 1

Here we provide a proof of proposition 1, which in our case tells us that every possible penalization function can further be reduced to zero.

Proof. Since f is non-constant there exists $z \in \mathbb{R}^n$ such that $f(z) \neq 0$. Since f is continuous, we can use the intermediate value theorem for multivariate(Shimamoto, 2019) functions and deduce that for all

$$\xi \in [f(x), f(z)] = [0, f(z)] \subset \mathbb{R}$$

there exists an $y \in [x, y]$ such that $f(y) = \xi$. Expecially, for all $\varepsilon > 0$ we can find an y such that $|f(y)| < \varepsilon$.

Table A.5: Results of DNFS compared to FIDL, CA, STG, and SHAP on the benchmark datasets. Green highlights the method with the most optimal selections.

	MNIST		**Ames Housing**		**Basehock**	
Method	Val. Acc. [%]	N	Val. MAE [$\cdot 10^3$\$]	N	Val. Acc [%]	N
DNFS	**97**	**48**	**15**	20	95	53
FIDL	93	50	16	20	91	50
STG	96	47	15	23	93	51
CA	97	50	17	20	92	50
SHAP	92	50	16	20	92	50

Note that this is a more general result than necessary in our case. The result also covers z and (or) $f(z)$ being negative, which is impossible for a sensible penalization function p. Also we have the special case of $f(x) = f(0) = 0$ which is also covered by this result. □

Appendix C. Performance validation

The goal of the feature reduction task is to retain strong performance in the downstream supervised task, even after significantly reducing the number of input features. In Section 5, the performance of the task model is compared when trained on the full feature set and on the reduced set selected by DNFS, to ensure that the reduction does not negatively impact the task outcome. To further validate the effectiveness and generalizability of the selected features, we additionally trained an SVM or, in the case of regression tasks, a support vector regression (SVR) model with an RBF kernel, regularization parameter $C = 1.0$, and $\epsilon = 0.01$. These models were evaluated on the benchmark datasets using both the full and the reduced feature sets. The results, presented in Table C.6, are consistent with those in Table 2, confirming that DNFS reliably selects a compact yet effective subset of features without degrading task performance. For the Basehock dataset, only the classes "sci.space" and "rec.autos" were used in a binary classification setting, due to computational constraints that prevented successful training of the SVM on the full multiclass dataset. However, this does not affect the validity of the performance evaluation.

Table C.6: Validation performance of SVM/SVR models trained on the full and DNFS-reduced feature sets across benchmark datasets. *For the Basehock dataset, only the classes "sci.space" and "rec.autos" were used in the evaluation due to computational constraints.

Dataset	baseline	sel. perf.	N
MNIST	$95,2\%$	$93,5\% \pm 0,2\%$	50 ± 2
Ames H.	18.191	$15.000\$ \pm 500\$$	20 ± 3
Basehock*	99%	$98.8\% \pm 0.2\ \%$	53 ± 4

Appendix D. Stable statistical selection

Deciding when to stop the selection can be achieved either by training for a fixed number of epochs or by classic early stopping criteria like validation loss monitoring of the task model's performance. The latter has been implemented in the current version and leads to a severe decrease in runtime (cf. Appendix E.4) while maintaining high performance and a good selection.

Figure D.8 shows how often a given feature was classified as selected in the selection interval for different thresholds δ. Most of the features are selected either in nearly 100% or 0% of epochs, leaving just a small portion of features not decisively classified. The heatmap in Figure D.7 shows that changes in δ and p have minimal impact on the number of selected features, which is the desired behavior. Therefore, it suffices to use the default values (cf. Appendix E) when performing selection with DNFS. Convergence tends to get stabler for larger datasets with more features. The most unstable convergence was yielded on the synthetic classification. Here, changes in δ have the largest impact.

Appendix E. Experimental setup

Appendix E.1. Comparitive methods

The experimental setups are similar but vary slightly from task to task. For the regularization the built in TensorFlow[1] implementation was used. For SHAP[2] and FIDL[3] the authors' implementations were used. For comparison, the following

[1]TensorFlow: https://github.com/tensorflow/tensorflow
[2]SHAP: https://github.com/slundberg/shap
[3]FeatureImportanceDeepLearning: https://github.com/maksym33/FeatureImportanceDL

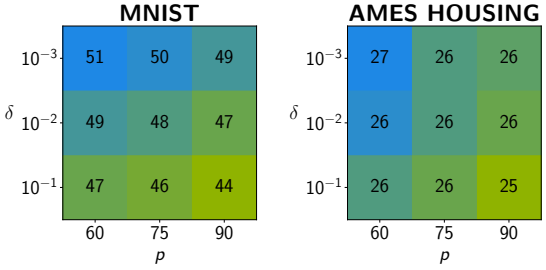


Figure D.7: The number of selected features for different p and δ on the benchmark datasets.

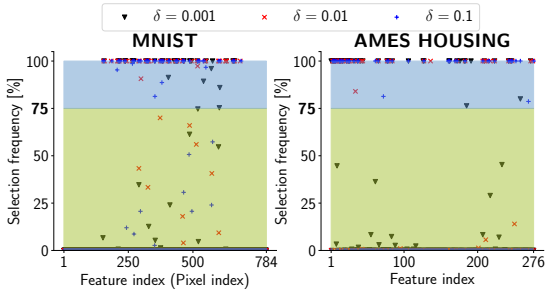


Figure D.8: Distribution of the statistically selected features for different thresholds δ on the benchmark datasets.

methods were chosen, all of which were described detailed in section 2. The required hyperparameters were set as follows:

The **CA** parameters were set according to the guidelines and examples given by the authors Balın et al. (2019). The model architectures were similar to the ones chosen for implementing DNFS.

For **FIDL** the same networks and setups as described in (Wojtas and Chen, 2020) were used for the synthetic datasets and the MNIST dataset. For the Ames housing task, the same task (operator) and selection (selector) networks as in our method (cf. Table E.8) were used.

For calculating **SHAP** values, there are no hyperparameters. To speed up the calculations for the MNIST dataset, a randomly generated subset of 10% the original size was used.

Appendix E.2. Setup

For all tasks, the penalization function p was employed as the arithmetic mean over all mask entries of $\mathbf{m}$ since this yielded stable results. The arithmetic mean has some unsuitable properties, which can be adverted by rethinking the convergence of the method as described in section 6.4.

Appendix E.3. Networks

Both networks are MLPs, with the selection network incorporating dropout layers between the fully connected ones. To mitigate the impact of the dropout layers on the final mask it's calculated not as the last output of the selection model, but rather as described in 6.5. To mitigate the vanishing gradient problem(Basodi et al., 2020) the rectified linear unit (ReLU) activation function was selected for the hidden

Table E.7: Hyperparameters and their values for the new method. λ denotes the value chosen for the implementation and λ_{calc} the value approximated as presented in section 6.3. For λ_{calc} to deliver the expected results, the target has to be normalized.

Dataset	Epochs	λ	$\approx \lambda_{calc}$	Batch size	p	δ
Synthetic Reg.	400	2	33	32	75%	0.01
Synthetic Class.	400	3	1,5	32	75%	0.01
Synthetic Reg. n.l.	400	2	33	32	75%	0.01
Synthetic Class. n.l.	400	3	1,5	32	75%	0.01
MNIST	300	10	0,5	64	75%	0.01
Ames housing	600	10^{-9}	20	32	75%	0.01
Basehock	200	0,5	0,3	64	75%	0.01

layers in the task and selection model, while a linear (ReLU if target was min-max-transfomed) activation function (resp. a softmax activation function) was chosen for the output layer in the case of a regression (resp. classification) task.

Appendix E.4. Runtime and hardware

Table E.9 shows the runtime requirements of DNFS on the various datasets as well as the network complexity. As described in section 6.6, DNFS is more computationally intensive than its competitors SHAP and L1 regularization but is also the only method that delivers reliably good performance. Further, it outperforms FIDL, the only approach comparable to DNFS concept-wise. It is highly probable that further code optimization could reduce the computing requirements significantly, for example, via swapping the order in which the task and selection model update. Greater utilization of built-in keras methods also has the potential of reductions in runtime since those are highly optimized. Furthermore, it is likely that for a successful selection just a fraction of the dataset is sufficient, thereby reducing time per epoch. First, tests for the MNIST dataset have delivered promising results. Extensive code optimization is beyond the scope of this work, and therefore, we focused on demonstrating the feasibility and effectiveness of DNFS, leaving performance tuning and advanced optimizations for future research.

The time measurements have been conducted on a computing node with 16 CPU cores and 32GB of memory. The CPUs were Intel Cascade Lake Xeon Gold 6230N

Table E.8: Task and selection networks that were used for the different datasets. The dropout layers in the selection network are not shown. In the case of the synthetic datasets the network architecture remained the same, whether the dependencies were linear or non-linear.

Dataset	Task network	Selection network
Synth. reg.	$\rightarrow 20 \rightarrow 100 \rightarrow 50 \rightarrow 50 \rightarrow 25 \rightarrow 1$	$\overset{d=0.5}{\rightarrow} 100 \overset{d=0.5}{\rightarrow} 75 \overset{d=0.5}{\rightarrow} 50 \overset{d=0.5}{\rightarrow} 10$
Synth. class.	$\rightarrow 20 \rightarrow 60 \rightarrow 30 \rightarrow 20 \rightarrow 10 \rightarrow 2$	$\overset{d=0.5}{\rightarrow} 100 \overset{d=0.5}{\rightarrow} 75 \overset{d=0.5}{\rightarrow} 50 \overset{d=0.5}{\rightarrow} 10$
MNIST	$\rightarrow 242 \rightarrow 100 \rightarrow 10$	$\overset{d=0.5}{\rightarrow} 256 \overset{d=0.5}{\rightarrow} 784$
Ames	$\rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 1$	$\overset{d=0.5}{\rightarrow} 256 \overset{d=0.5}{\rightarrow} 276$
Basehock	$\rightarrow 128 \rightarrow 512 \rightarrow 256 \rightarrow 2$	$\overset{d=0.5}{\rightarrow} 512 \overset{d=0.5}{\rightarrow}$ $256 \overset{d=0.5}{\rightarrow} 512 \overset{d=0.5}{\rightarrow} 7862$

(20-core, 2.3GHz, 30MB Cache, 125W)

Appendix F. Regarding code and implementation

The whole method is available in a GitHub repository that can be accessed via this link. There, too, is described what steps to take if one desires to reproduce the results. The concrete implementation can obtained from the code if of interest, but some key points will be discussed in the following.

DNFS is implemented using TensorFlow and utilizying the Keras API[4] for ease of use and implementation. We employed Adam(Kingma and Ba, 2017) as the optimizer solving the optimization problem. For simplicity, the generated metrics and masks are averaged over all batches of an epoch since the changes are minimal, and there would have been no point in storing every metric for every update.

Further, the implementation has just been tested on Linux systems. Theoretically, there should be nothing preventing you from testing the implementation on Windows, except that you perhaps have to change the optimizer to the non-legacy one. Moreover, we cannot make any statements about macOS and macOS on ARM, but due to the common Unix roots, the implementation will likely work seamlessly if all packages are installed.

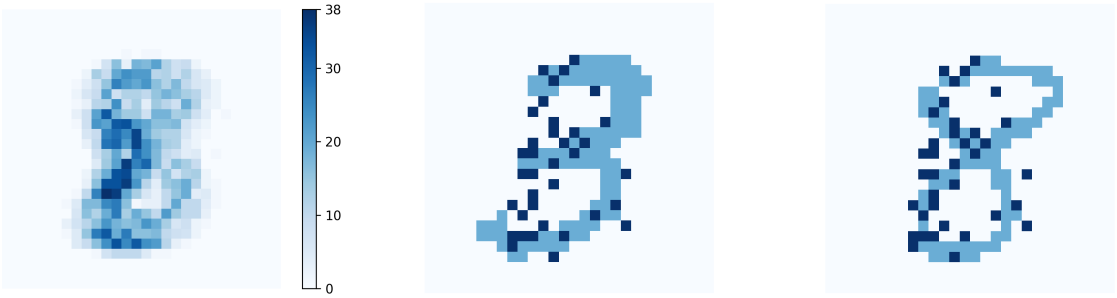
[4]Keras: https://keras.io/

Table E.9: Overview of training runtime (ms per epoch) and network size (number of learnable parameters) of DNFS across various datasets.

Dataset	$\approx t_{base}$	$\approx t_{DNFS}$	$\approx$ Increase	N_{task}	$N_{sel.}$
Synthetic Reg.	51 ms	71 ms	1,4x	11.221	12.985
Synthetic Class.	53 ms	69 ms	1,3x	3.972	12.985
MNIST	300 ms	800 ms	2,6x	20.6172	402.448
Basehock	530 ms	860 ms	1,6x	2.621.698	5.246.974
Ames housing	96 ms	136 ms	1,41x	45.825	141.844

Appendix G. More results on benchmark datasets

Figure G.9 visualizes the selections of DNFS on a subset of the MNIST dataset that contains just the digits three and eight. Figure G.9a shows that DNFS predominantly selects pixels at which the two digits are distinguishable.



(a) On average selected pixels over 100 trials.

(b) Selected mask on top of the digit 3.

(c) Selected mask on top of the digit 8.

Figure G.9: Visualization of the selected masks on the MNIST-Subset dataset with just the classes 3 and 8. The mask displayed in Figures G.9b and G.9c contains 36 pixels and yields a validation accuracy of 99%.

Appendix H. Dataset overview

Table H.10 offers a succinct summary of essential dataset metrics. The table below presents key statistics, including the number of classes, features, and overall dataset size.

Table H.10: Summary of key properties of synthetic and benchmark datasets.

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{D}\rvert$	Train-Val. Ratio	Classes	Data
Synth. Reg.	10	1500	66%	reg.	numeric
Synth. Class.	10	1500	66%	2	numeric
Synth. Reg. n.l.	10	1500	66%	reg.	numeric
Synth. Class. n.l.	10	1500	66%	2	numeric
MNIST	784	10.000	80%	10	image
Basehock	7862	1992	80%	20	text
Ames Housing	276	2930	80%	reg.	numeric